Breaking the Model Selection Barrier? THREE LEARNING PARADIGMS FOR SUPERVISED ENSEMBLING IN TIME SERIES ANOMALY DETECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Time series anomaly detection (TSAD) is a long-standing and extensively studied problem with applications across a large panel of domains. Despite the maturity of the field, recent benchmark studies have revealed that no single detection method consistently outperforms others across diverse datasets. While model selection approaches (i.e., choosing the best detector for a given scenario) have shown promising results, their effectiveness remains inherently limited by the performance ceiling of existing individual detectors. To address this limitation, supervised ensembling offers a promising path to surpass individual detectors by learning to combine their strengths. In this work, we unify and formalize the problem of supervised ensemble-based anomaly detection in time series, and introduce three principled strategies for learning such ensembles: (1) classical Machine Learning, (2) Reinforcement Learning, and (3) Genetic Programming. We perform a rigorous comparative evaluation across these strategies using identical model components, inputs, and experimental conditions to ensure fairness. Our findings not only highlight the strengths and trade-offs of each approach, but also identify Reinforcement Learning as a paradigm that can break the model selection barrier. Our experimental evaluation analyzes the robustness to *distribution shift*, *label budget*, and *detector pool*, pointing to promising directions for future research.

1 INTRODUCTION

Anomaly detection in time series data (Paparrizos et al., 2025) is a long-standing and critical task with broad applicability across various domains, including finance (Lütkepohl et al., 2004), industrial monitoring (Antoni & Borghesani, 2019), healthcare (Hadjem et al., 2016), environmental science (Goddard et al., 2003), and cybersecurity (Filonov et al., 2016). Over the years, a wide range of methods have been developed, including statistical techniques (Li et al., 2020), unsupervised learning, and more recently, deep learning-based approaches (Chauhan & Vig, 2015; Kim et al., 2018). In particular, unsupervised anomaly detection (Goldstein & Dengel, 2012; Audibert et al., 2020) remains the dominant paradigm due to the scarcity of labeled anomalies in real-world datasets.

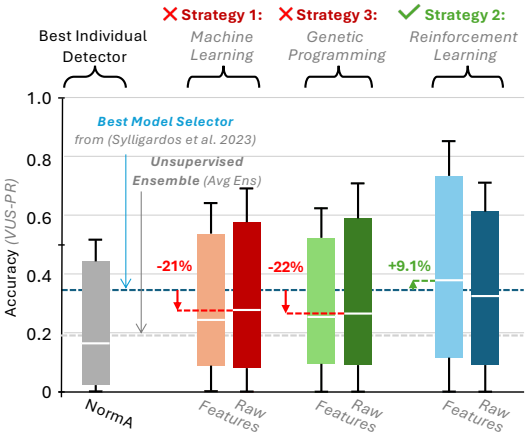


Figure 1: Comparison of the three strategies on the TSB-UAD benchmark (Paparrizos et al., 2022b).

However, recent benchmark studies (most notably the TimeEval (Schmidl et al., 2022), TSB-UAD (Paparrizos et al., 2022b), and TSB-AD (Liu & Paparrizos, 2025) benchmarks) have revealed a key insight: *no single detector consistently achieves top performance across diverse datasets and anomaly types*. This heterogeneity challenges the generalizability of individual methods and has motivated research into *model selection* approaches, where

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

the best-performing detector is chosen based on time series characteristics. Although promising, such methods are intrinsically limited by the performance ceiling of the best individual detector.

To overcome this ceiling, supervised ensemble learning offers an appealing alternative: rather than selecting a single detector, ensembles aim to combine the strengths of multiple detectors to achieve performance that exceeds any individual detector (Sylligardos et al., 2025; 2023; Vautier et al., 2025). While ensemble techniques are widely studied in classification and regression, their application in the domain of time series anomaly detection, particularly under a supervised setting, remains under-explored and lacks a unified formalization and evaluation. We emphasize here that we do not assume that abundant labeled anomalies are available in general but rather investigate whether limited supervision can be leveraged to surpass the performance ceiling of unsupervised methods.

In this paper, we conduct a comprehensive study of **supervised ensembling strategies for time series anomaly detection** and propose **three strong, reproducible baselines** designed to establish a clear and reliable performance bar for future research. We formalize the task as a supervised problem, where the objective is to learn a function that produces weights to combine individual detectors. We explore this problem through three principled and diverse learning paradigms:

- **Classical Machine Learning (ML):** A given model trained on either raw time series sequences or pre-computed features (Lubba et al., 2019).
- **Reinforcement Learning (RL):** An agent (i.e., a model) that learns to combine detector scores over time by maximizing a reward given by the anomaly detection accuracy.
- **Genetic Programming (GP):** An approach where individuals (i.e., models) are evolved via mutation and selection, optimizing the anomaly detection accuracy as a fitness function.

We compare *feature-based* and *raw-based* input representations within each strategy to assess their impact on ensemble quality. All methods are evaluated on the TSB-UAD benchmark, using consistent test splits, detectors, and experimental conditions to ensure fairness and comparability. **Crucially, our goal is not to compare architectures** (CNN vs LSTMs) **or different ensembling methods** (Stacking (Wolpert, 1992) vs Weighted voting (Khan et al., 2024)) **or hyperparameter sensitivity but to benchmark our three strong baselines upon a simple, reproducible baseline** for supervised ensembling grounded in existing research. To establish the value of supervised ensembling, we benchmark our strategies against: (i) the *best individual detector on TSB-UAD*, (ii) a naive *unsupervised average ensemble*, and (iii) the strongest *model selection* baselines from a recent experimental evaluation (Sylligardos et al., 2023; 2025).

Our findings, summarized in Figure 1, demonstrate that supervised ensembles can reliably outperform individual detectors and unsupervised ensembling, and in some cases, even surpass the best model selection methods (Sylligardos et al., 2023). These results highlight the potential of supervised learning in this context and open up promising directions for future work.

2 Background and Related Work

In this section, we review time series anomaly detection literature and discuss recent automated solutions while detailing their limitations. First, a time series $T = [T_1, T_2, ..., T_L] \in \mathbb{R}^L$ is a sequence of real-valued numbers $T_i \in \mathbb{R}$, where $L = |T|$ is the length of T, and T_i is the i^{th} point of T. Local regions of the time series, known as subsequences $T_{i,\ell} \in \mathbb{R}^\ell$ of a time series T, is a subset of successive values of T of length ℓ starting at position i, formally defined as $T_{i,\ell} = [T_i, T_{i+1}, ..., T_{i+\ell-1}]$. For a time series $T \in \mathbb{R}^L$, an anomaly detection method (or detector) D returns an anomaly score sequence $S_T \in \mathbb{R}^L$. We note N the number of detectors.

2.1 Time Series Anomaly Detection

Anomaly detection in time series is a crucial task for many relevant applications. Therefore, several methods (or detectors) have been proposed (Boniol et al., 2024). One type of anomaly detection method is *distance-based methods*, which analyze subsequences by utilizing distances to a given model to detect anomalies (Yeh et al., 2016; Breunig et al., 2000; Boniol et al., 2021a; Nakamura et al., 2023; Lu et al., 2022; Huang et al., 2026).

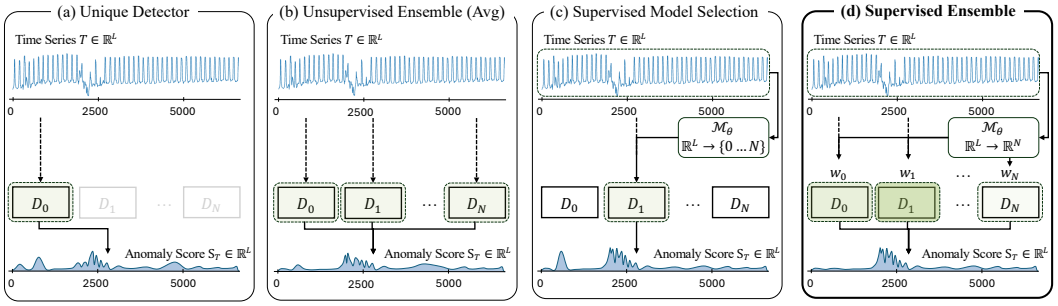


Figure 2: Time series anomaly detection: from unique detector (a) to supervised ensembling (d).

While methods in the previous category compute their anomaly score based on distances using raw time series elements (such as subsequences), *density-based methods* focus on detecting recurring or isolated behaviors by evaluating the density of points or subsequences within a specific representation space. This category can be divided into four sub-categories, namely *distribution-based* (Goldstein & Dengel, 2012), *graph-based* (Boniol & Palpanas, 2020; Wenig & Papenbrock, 2024), *tree-based* (Liu et al., 2008), and *encoding-based* (Gao et al., 2020).

Finally, *prediction-based* approaches aim to detect anomalies by predicting the expected normal behaviors based on a training set of time series or sub-sequences (containing anomalies or not). Methods in this category detect anomalies using prediction errors. More specifically, such category can be divided into *forecasting-based methods* (Malhotra et al., 2015; Su et al., 2019), and *reconstruction-based methods* (Sakurada & Yairi, 2014; Tuli et al., 2022; Goswami et al., 2024; Yang et al., 2023). A more detailed review is in the Appendix A.1.

2.2 One Unique Detector is Hopeless

Recently, several experimental evaluations for anomaly detection in time series have been proposed (Schmidl et al., 2022; Paparrizos et al., 2022b; Liu & Paparrizos, 2025). Such benchmarks provide a large collection of time series from various domains and evaluate multiple methods spanning all the categories mentioned above. However, these evaluations led to the same conclusion: no single method is universally best across all domains. The two main reasons are as follows:

Heterogeneity in anomaly types: Time series anomalies are either *point*, *contextual*, or *collective*. Such heterogeneity in anomaly types makes the anomaly detection task challenging. Moreover, even for time series with the same anomaly types, we observe that the most accurate models are all different (Sylligardos et al., 2023).

Heterogeneity in time series structures: On top of heterogeneity in anomaly types, we need to differentiate time series containing *single* or *multiple* anomalies (with different or similar anomalies). For instance, methods based on neighbor distance computation, such as LOF, are very accurate in detecting *single* or *multiple different* anomalies, but less accurate for *multiple similar* anomalies.

2.3 Towards Automated Solutions

A solution to the limitations mentioned above is to apply **model selection** (depicted in Figure 2(c)) based on time series characteristics (Sylligardos et al., 2023; 2025). The goal is to train a model to automatically select the best detectors for a given time series. A recent evaluation benchmarking different automatic methods highlighted model selection (i.e., meta-modeling) as the state-of-the-art approach for time series anomaly detection (Liu et al., 2025). However, model selection methods are inherently constrained by the detectors they select from. Since these approaches operate by choosing an existing detector, their performance is fundamentally bounded by the best detector within the candidate pool. As such, model selection cannot produce results that exceed the capabilities of its individual components (in this paper, we refer to the *Oracle* as the best theoretical model selector). This limitation arises when no single detector performs well across all datasets or anomaly types.

A solution to this limitation is to apply an **unsupervised ensemble** to the anomaly scores produced by all the detectors (Figure 2(b)). While several unsupervised ensembling techniques have been proposed (Aggarwal & Sathe, 2015), the *Averaging* strategy is the most robust and low-risk strat-

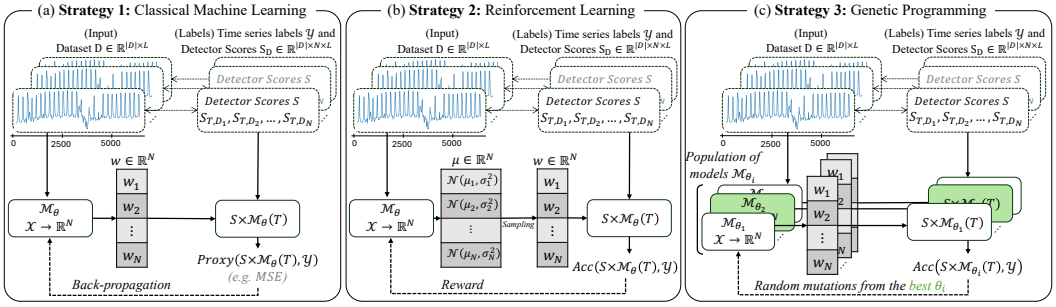


Figure 3: Overview of the different strategies considered for supervised ensembling.

egy (Aggarwal & Sathe, 2015). However, despite offering simplicity (Goswami et al., 2022; Schmidl et al., 2024) and robustness, this approach remains fundamentally agnostic to the time series and the contextual reliability of each detector (i.e., all treated equally). As a result, they are unable to emphasize the strengths of high-performing detectors or mitigate the weaknesses of less reliable ones. In contrast, a *supervised ensemble* (illustrated in Figure 2(d)) can learn to combine detector outputs based on actual anomaly patterns, potentially surpassing the performances of individual detectors, naive ensembles, and model selection.

2.4 Supervised Ensembling: *Problem Definition*

For a time series of length L, the supervised ensemble learning paradigm is defined as follows: the input data X is either a feature vector in $\mathbb{R}^F$, where F is the number of features, or a raw time series in $\mathbb{R}^L$. Given X, the weights W applied to each individual detector are sampled from a trained model $\mathcal{M}_{\theta_*} : X \to p_X \in \mathcal{P}(\mathbb{R}^N)$ which outputs a probability distribution over the set of weighting coefficients for the N detectors. The anomaly scores from the detectors are denoted by $S \in \mathbb{R}^{L \times N}$, and the corresponding ground truth label is represented by $Y \in \{0,1\}^L$. Finally, we note Acc, the anomaly detection metric (such as VUS-PR (Boniol et al., 2025)). The optimal model parameters, defined by $\theta_* \in \Theta := \mathbb{R}^{d_\mathcal{M}}$ where $d_\mathcal{M}$ is the number of parameters, are obtained as follows:

$$\theta_* = \arg\max_{\theta \in \Theta} \mathbb{E}_{(X,S,Y),\, W \sim \mathcal{M}_\theta(X)} \left[Acc\big(S \times \widetilde{W}, Y\big) \right]. \tag{1}$$

Since θ_* is unknown, we aim at maximizing the expected accuracy metric Acc through appropriate learning strategies. The term $S \times \widetilde{W} \in \mathbb{R}^L$ represents the output of the supervised ensemble model, i.e., the weighted anomaly scores, in which $\widetilde{W}$ is the weights vector W normalized by its norm. Therefore, the goal is to identify the most effective strategy for optimizing $\mathcal{M}_\theta$ as defined in Equation 1. The most effective strategy can be **Deterministic** (such as ML or GP) or **Stochastic** (such as RL). We provide a reformulated version of Equation 1 for the deterministic case in Appendix A.3.

3 TSAD Supervised Ensembling: *Three Strong Baselines*

In this section, we present our core contribution and introduce three strong baselines for supervised ensembling for time series anomaly detection. For fair comparison, we introduce a generic pipeline as a foundation for all strategies.

3.1 A Generic Pipeline

To consistently evaluate the three supervised ensembling strategies, we define a generic training and evaluation pipeline. Each time series in the dataset is either (i) converted into a set of F statistical features, resulting in one feature vector per time series, or (ii) reduced to the first window of fixed length ℓ if using the raw time series data directly. The model (as defined by the chosen strategy) outputs a weight vector $W \in \mathbb{R}^N$, where N is the number of detectors. We compute the final ensemble anomaly score for a given time series T as a weighted average of the detector scores using the normalised predicted weight noted $\widetilde{W}$. Contrary to Mixture of Experts (MoE) (Yuksel et al., 2012) in our framework, the experts are a diverse set of pre-existing, heterogeneous, and pre-trained algorithms. The pipeline is summarized in Algorithm 1 in Appendix A.4.

Table 1: Experimental setup for the evaluation of Ensemble Strategies

	Strategy 1 (ML)	Strategy 2 (RL)	Strategy 3 (GP)
Datasets and Detectors	TSB-UAD (Paparrizos et al., 2022b) (16 datasets, 12 detectors)		
Train/Test Split	*In-distribution:* 70% Train, 30% Test ; *Out-of-distribution:* Leave-One-Dataset-Out		
Input Representations	Raw windows of length 128 *or catch22* features (Lubba et al., 2019)		
Core Model *(Features)*	Best features-based model in (Sylligardos et al., 2023): Multi-Layer Perceptron (MLP)		
Core Model *(Raw)*	Best window-based model in (Sylligardos et al., 2023): Convolutional Neural Network (CNN)		
Optimization Strategy	Backpropagation	PPO (Schulman et al., 2017)	Genetic Algorithm
Loss / Objective	MSE	AUC-PR or VUS-PR as reward	AUC-PR or VUS-PR as fitness
Optimization Details	ADAM Optimizer	PPO with clipped surrogate loss	`pygad.GA` (Gad, 2023)

3.2 STRATEGY 1: *Classical Machine Learning*

We first consider the most straightforward path to solve the optimization problem: a gradient method minimizing an objective defined as closely as possible to the anomaly detection accuracy (illustrated in Figure 3(a)). The non-differentiability of the time series anomaly detection accuracy measure AUC-PR or VUS-PR (Paparrizos et al., 2022a) forces us to choose a differentiable proxy, such as the Mean Squared Error (MSE). However, selecting a differentiable proxy introduces inherent limitations, as we are no longer optimizing the true objective (i.e., the anomaly detection accuracy). Instead, we rely on a surrogate that may not faithfully reflect the nuances of the actual target. Formally, the problem initially formulated in Equation 1 is

$$\theta_* = \arg\max_{\theta \in \Theta} \mathbb{E}_{(X,S,Y)} \Big[\text{Proxy}\big(S \times \widetilde{M_\theta}(X), Y\big) \Big] \tag{2}$$

where $\widetilde{M_\theta}(X)$ are the normalized outputs (weights) of M_θ. In practice, we consider MSE as the Proxy. Moreover, the pipeline used in Strategy 1 closely resembles the model selection approach proposed in Sylligardos et al. (2023). The key difference lies in performing regression (i.e., predicting continuous weights for each detector) rather than classification. While framing the problem as a classification task may be more natural (considering the labels we are generating), adopting a regression-based approach ensures consistency with Strategies 2 and 3. This alignment allows for a fair comparison between strategies, focusing solely on the learning paradigm employed.

3.3 STRATEGY 2: *Reinforcement Learning*

In order directly optimize a non-differentiable TSAD metric (such as VUS-PR) without handcrafting a surrogate loss we consider a reinforcement learning (Figure 3(b)) strategy and use the metric as a reward for a policy gradient method (Sutton et al., 1998). In this framework, the agent (i) *observes a state* X (the features or raw window of a time series), (ii) *takes an action* W (the detector weights), and (iii) *collects the reward* $R(X, W) := Acc(S \times \widetilde{W}, Y)$ (notations of Equation 1). We note that in our implementation there is no spatial dependency between the states, each state-action-reward tuple is treated independently.

The objective is to learn the policy $\pi_\theta : X \mapsto \{\mathcal{N}(W_i; M_\theta(X)_i, (\sigma_\theta^i)^2)\}_{1 \leq i \leq N}$ (whose means are the deterministic outputs of the model M_θ of Equation 1) such that the expected reward is maximized. We chose the RL notations here but to stick with Equation 1 notations one can replace π_θ by $\mathcal{M}_\theta$. In practice, we use Proximal Policy Optimization (PPO) (Schulman et al., 2017) where a shared critic network estimates the value function $V_{\pi_\theta}(X) = \mathbb{E}_{W \sim \pi_\theta(X)}[R(X, W)]$ from which the advantage function $A_{\pi_\theta} = R(X, W) - V_{\pi_\theta}(X)$ measures how much better the action W is compared to the average action of the current policy π_θ.The objective function is:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_{\pi_\theta} \big[\min \left(r(\theta) A_{\pi_\theta}, \text{clip}(r(\theta), 1 - \epsilon, 1 + \epsilon) A_{\pi_\theta} \right) \big] \tag{3}$$

where $\epsilon > 0$, and $r(\theta) := \pi_\theta(X)(W) / \pi_{\theta_{\text{old}}}(X)(W)$ relates the evolution of the likelihood of the action W at the state X. The clip operator (Schulman et al., 2017) restricts $r(\theta)$ to $[1 - \epsilon, 1 + \epsilon]$.

3.4 STRATEGY 3: *Genetic Programming*

In contrast to Strategies 1 and 2, genetic programming offers a population-based, gradient-free optimization paradigm (Figure 3(c)). Like RL, it does not require a differentiable objective. Rather than optimizing a policy it complements RL and ML by directly exploring the solution space through

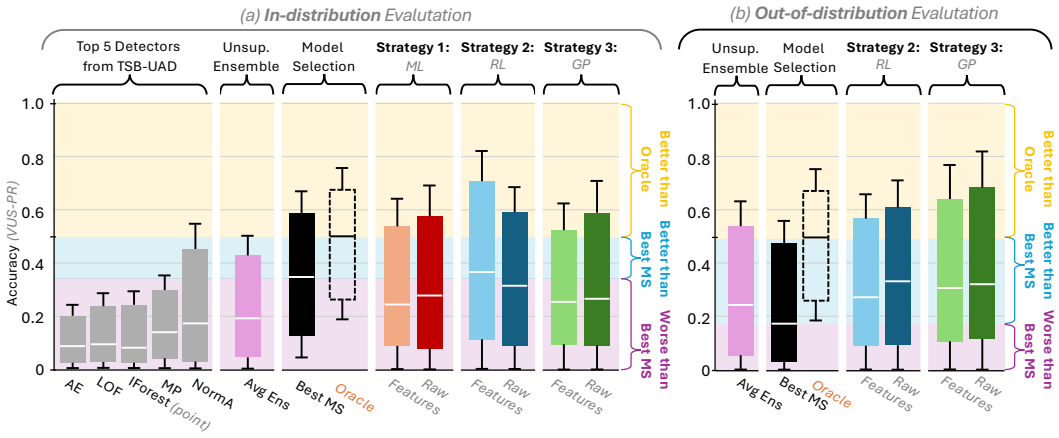


Figure 4: Accuracy averaged over 5 random seeds ((a) *in-distribution*, (b) *out-of-distribution*) of Strategy 1, 2, and 3 versus individual detectors, unsupervised ensemble, and the best model selector.

an evolutionary process (Mitchell, 1998), offering an almost model-free baseline. This difference is supported by the existence of hybrid frameworks such as Evolutionary Algorithm Reinforcement Learning (EARL) (Moriarty et al., 1999). The population at generation k is $P^k = \{\theta_1^k, \ldots, \theta_N^k\}$, where each individual $\theta_i^k \in \mathbb{R}^d$ is a different flattened weight vector of the same network $\mathcal{M}_\theta$ with a fixed architecture. The quality is evaluated by the fitness function using the detection metric Acc:

$$f(\theta) := \mathbb{E}_{(X,S,Y)}\big[Acc\big(S \times \widetilde{M}_\theta(X), Y\big)\big] \quad ; \quad \theta_* = \arg\max_{\theta \in \Theta} f(\theta) \tag{4}$$

The selection phase chooses the best individuals (based on fitness) to serve as parents. Among several strategies, we use the steady-state approach, in which only the least fit individuals are replaced each generation, while the best are retained. Formally, let $M \leq N$ and define $\{\theta_1^k, \ldots, \theta_M^k\} \subset P^k$ as the M best individuals at generation k. These parents produce offspring for the next generation through crossover (each child inherits half of its parameters from each parent) and mutation (10% of a child's parameters are perturbed by adding a random value between $[-1, 1]$).

4 Experimental Evaluation

We now evaluate the three strategies described above by answering the following questions:

(Q1) How accurate is supervised ensembles? We benchmark our strategies against traditional baselines, such as individual detectors, unsupervised ensembles, and model selection.
(Q2) Do supervised ensembles really ensemble? We assess whether the supervised ensemble mimics model selection (selecting only one detector) or combines meaningful detectors (leading to higher performances than the best detector on each time series).
(Q3) How do the strategies scale? We measure training and detection time for all strategies.
(Q4) How robust is the best strategy? We evaluate the accuracy evolution under different label budgets and after iteratively removing the three best detectors.

4.1 Experimental Setup

Table 1 summarizes our experimental setup. Overall, Strategy 1 relies on the ADAM optimizer with MSE loss. Strategy 2 employs the PPO algorithm (Schulman et al., 2017), where the core model acts as the policy network. Strategy 3 uses genetic programming via the `pygad.GA` library (Gad, 2023) to evolve model weights directly. We provide additional details in Appendix A.5.

We benchmark against four baselines: (i) 12 individual detectors from TSB-UAD (Paparrizos et al., 2022b), (ii) an unsupervised ensemble computed as the average of all detectors' outputs (called **Avg Ens**), (iii) the best model selection method identified in (Sylligardos et al., 2023) (called **Best MS**) and (iv) the best theoretical model selector, i.e., selecting the best detector for each time series (called **Oracle**). These comparisons enable a comprehensive assessment of the benefits and limitations of our proposed strategies relative to established methods. We provide more details in **our repository** [1].

[1] Our repository: link

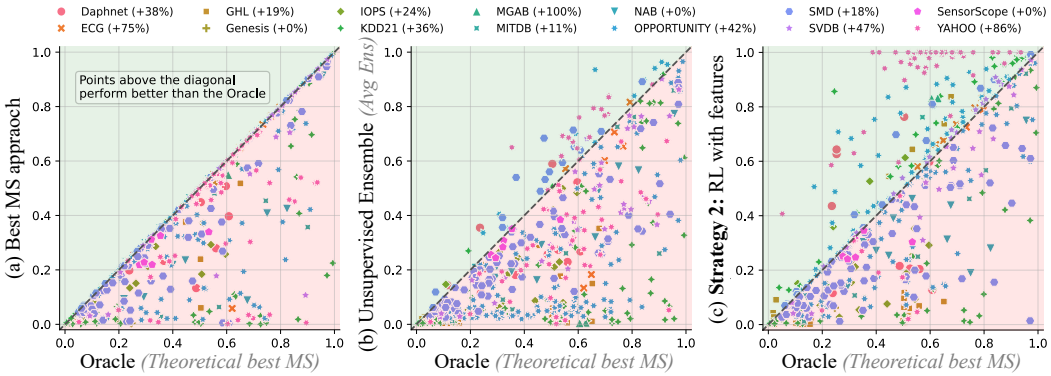


Figure 5: VUS-PR comparison of (a) *Best MS*, (b) *Avg Ens*, (c) Strategy 2 (with features) against the *Oracle*. % of time series per dataset on which RL (with features) outperforms the *Oracle*.

(Q1). How accurate is supervised ensembles?

In this section, we first compare the different strategies against the baselines (*Individual detectors*, *Avg Ens*, *Best MS*, and *Oracle*) and we identify which strategy is the most promising. In Figure 4, all strategies produce weights in $[0, 1]$. The scenario where ML and GP infer detector weights in $[-1, 1]$ is discussed in Appendix A.9. To achieve the comparison between supervised ensembling and the baselines we conduct two experiments: in the first (Figure 4(a)), we evaluate the anomaly detection accuracy (VUS-PR) of all strategies over the entire TSB-UAD benchmark (i.e., *in-distribution* setting). In such a setting, both the training and the test sets contain time series from all 16 datasets of TSB-UAD. In the second experiment (Figure 4(b)), we evaluate the performance of the models in an *out-of-distribution* setting (i.e., evaluated on a dataset not used in the training set) serving as a proxy for robustness assessment in label-scarce scenarios and demonstrating that supervision can be transferable and practically valuable. In this scenario, we leave one entire dataset out for the test and use the 15 remaining datasets for training. Based on the first experiment, we exclude Strategy 1 from the second experiment.

Table 2: Variability of the mean performance of each strategy across 5 random seeds in *in-distribution* setting (mean±std).

Method		VUS-PR
Strategy 1 (ML)	Features	0.336 ± 0.019
	Raw	0.347 ± 0.005
Strategy 2 (RL)	Features	$\mathbf{0.440 \pm 0.004}$
	Raw	$\underline{0.382} \pm \underline{0.008}$
Strategy 3 (GP)	Features	0.332 ± 0.020
	Raw	0.356 ± 0.010
Avg. Ens.		0.2734
Model Selection (MS)		0.3819
Oracle *(theoretical best MS)*		0.4885

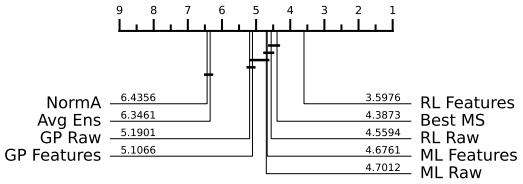


Table 3: Critical difference diagram using the Wilcoxon signed rank test ($\alpha = 0.05$).

In-distribution. Figure 4(a) shows that all supervised ensembling strategies significantly outperform individual detectors and *Avg Ens*, and that Strategy 2 (RL) on features outperforms *Best MS*: its median exceeds that of *Best MS*, and its accuracy distribution reaches higher values than the *Oracle*. Although its median remains below the *Oracle*'s, this shows that supervised ensembling can surpass the model selection *barrier* on specific datasets (see Q2). The choice of strategy (RL, GP or ML) also matters more than the input type: the gap between strategies is larger than the gap between features and raw subsequences within a given strategy. We attribute the strong performance of RL to its direct optimization of the true TSAD metric, VUS-PR. This hierarchy is stable across 5 seeds (Table 2), and the dominance of RL is statistically significant (Table 3).

Out-of-distribution. Figure 4(b) confirms the observation of Sylligardos et al. (2023) that ensembling outperforms model selection in this setting: both supervised strategies (RL and GP) outperform *Avg Ens* and *Best MS*. Surprisingly, unlike the in-distribution case, the data type matters more than the strategy: RL and GP reach similar accuracy but both perform significantly better on raw data than on features. We see two reasons for this, (i) in out of distribution ensembling many detectors is

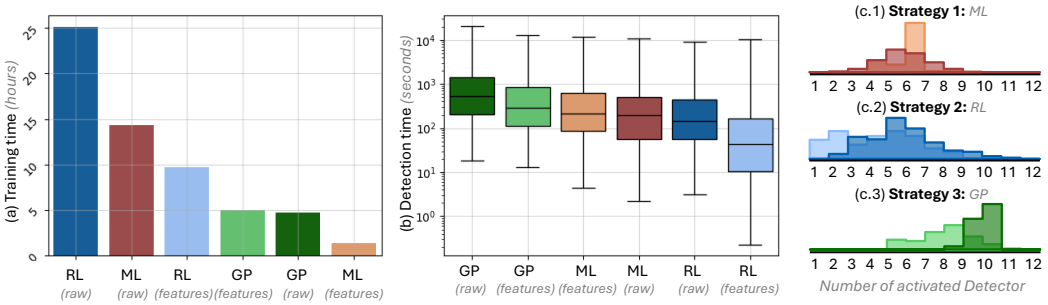


Figure 6: (a) Training time, (b) Detection time (i.e., both computing the weights and running the activated detectors), and (c) Number of activated detectors per strategy.

safer than selecting a few and we observe that feature based strategies select (i.e., with non-zero weights) fewer detectors than raw based ones. Additionally, (ii) the feature space is sparser than the raw subsequences, which may reduce generalisability. Additional results are given in Appendix A.9 (in-distribution) and Appendix A.10 (out-of-distribution).

Answer to Q1: *Supervised ensembling outperforms model selection in the out-of-distribution setting, and also does so in the in-distribution setting with Strategy 2 (RL) on features. Out of distribution, denser outputs (i.e., more detectors with non-zero weights as produced by raw-based strategies) are associated with better performance.*

(Q2). Do supervised ensembles really ensemble?

In the previous section, we observed that a supervised ensemble (Strategy 2 on features) can achieve higher performance than the *Oracle*. Thus, we conduct a per-time series comparison between Strategy 2 and the *Oracle*. Additionally, we include *Best MS* and *Avg Ens* in the comparison. Figure 5 depicts a pairwise comparison between them (i.e., each point is a time series).

First, Figure 5(a) illustrates the glass ceiling of model selection: by definition *Best MS* cannot outperform the *Oracle* (i.e., the best individual detector for each time series). As for *Avg Ens* (Figure 5(b)), it does not convincingly break this ceiling as only a few time series lie above the diagonal and unlike *Best MS* many of them (e.g., from OPPORTUNITY) are poorly detected (VUS-PR ≤ 0.1) while the *Oracle* remains accurate (VUS-PR ≥ 0.4). In contrast, supervised RL ensembling on features (Figure 5(c)) shows promising results, outperforming the *Oracle* on 42% of the OPPORTUNITY time series and on 86% of the YAHOO ones. It fails, however, to beat the *Oracle* when the latter performs poorly (no points in the top-left corner), and breaking the model selection *barrier* (i.e., lying above the diagonal) becomes more frequent as the *Oracle* improves. We discuss the impact of anomaly types and of accuracy variance among detectors in Appendix A.7.

Answer to Q2: *Supervised RL on features breaks the model selection barrier on MGAB (100%), YAHOO (86%) and ECG (75%) where it outperforms the Oracle on at least half of the time series, and is challenging on SVDB (47%) and OPPORTUNITY (42%).*

(Q3). How do the different strategies scale?

We now analyze the training and detection time of all three supervised ensembling strategies. The training times (Figure 6(a)) relate to the time needed to train and test the models entirely. The detection time (Figure 6(b)) depicts the time required to compute the weights and run the activated detectors (i.e., detectors associated with non-zero weights).

Figure 6(a) shows that raw based methods are slower than feature based ones with Strategies 1 and 2 (ML and RL), and equally slow with Strategy 3 (GP). Two factors are at play: (i) inferring the weights from raw subsequences is slower, as the CNN shared across all strategies has ~ 100 times more parameters than the MLP used on features; (ii) the experimental setup partly compensates for this by limiting the number of evaluations on raw data (Appendix A.5).

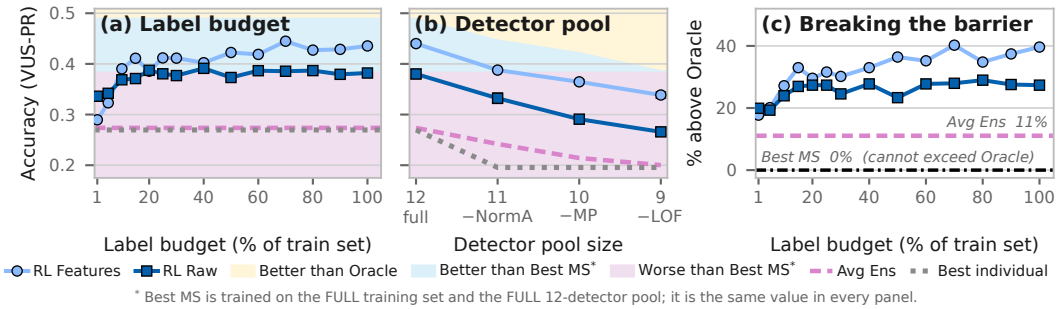


Figure 7: (a) Accuracy vs. label budget, (b) accuracy vs. detector pool size, and (c) percentage of test series above the *Oracle* accuracy. Panel (b) is averaged over 5 seeds; (a) and (c) use one seed.

In contrast, the detection time (Figure 6(b)) is not directly affected by the input type, but grows with the number of activated detectors (Figure 6(c), where a distribution shifted to the left indicates sparser weights). Sparse weights (i.e., few non-zero weights) are therefore key to reducing detection time. RL and GP activate fewer detectors on features than on raw data, and gradient based methods (ML, and RL with PPO) produce sparser activations overall. As a result, *RL on features* is not only the best supervised ensembling strategy in the in-distribution setting, but also the fastest.

Answer to Q3: *Supervised ensembling produces very different activated detector distributions, which have a significant impact on detection time and scalability (the sparser, the faster), as well as on the out-of-distribution scenario results (the denser, the better).*

(Q4). How robust to label budget and detector pool is the best strategy ?

Having identified RL as the best strategy in distribution and similar to GP out of distribution, we study its robustness to the **label budget** (Figure 7(a,c)) and to the **detector pool** (Figure 7(b)).

Label budget. The accuracy of RL on features increases steadily over the first 15% of the training set (+0.12 VUS-PR), and only slightly from 15% to 100% (+0.02). Since every fraction preserves the proportion of the different datasets this early increase cannot come from a specific training set composition. RL on both features and raw data almost reaches its final accuracy with 15% of the training set and its ability to break the model selection barrier depends only marginally on the budget: from 15% of the label budget RL on features outperforms *Best MS* in mean accuracy and beats the *Oracle* on nearly 30% of the time series. (Figure 7(c)).

Detector pool. We iteratively remove the most weighted detector and retrain the models from scratch after each removal. Consistently with Figure 5(c), RL exploits the strengths of the detectors so its accuracy decreases when strong detectors are removed. However, this decrease is linear, with a constant gap between RL and the *Oracle*. Thus, RL does not rely exclusively on a single detector.

Answer to Q4: *The RL approach is only marginally sensitive to the label budget as 93% of its final mean accuracy is reached with only 15% of the training set. Moreover, iteratively removing the best detectors does not cause a sharp drop either: RL on features still outperforms Best MS after the best detector is removed and its gap to the Oracle remains nearly constant.*

5 Conclusion and Key Takeaways

We conduct a comprehensive study of supervised ensembling strategies for time series anomaly detection. We explore three principled and diverse learning paradigms (ML, RL, and GP) and evaluate them in terms of accuracy and execution time. Overall, our key takeaways are: **T1:** Supervised ensembling outperforms model selection *out of distribution* and manages to do it *in distribution* with Strategy 2 (RL). **T2:** Strategy 2 (RL) with features breaks the model selection barrier on 40% of the time series. **T3:** The sparser the ensembling methods are, the faster. However, sparser outputs imply poor *out-of-distribution* performances. **T4:** Strategy 2 (RL) requires 15% of the training set to reach 93% of its final accuracy and still beats *Best MS* after the best detector is removed. These results highlight the potential of supervised learning in this context and open up promising directions for future work, particularly in the context of reinforcement learning, as a strong supervised ensembling baseline for time series anomaly detection.

AI USE STATEMENT

In this work, we used generative AI tools to assist with translation. We have not used generative AI tools to design or provide feedback on research methodology or experiments or to implement methods or to clean and reformat dataset and neither to interpret results. Concerning the tasks about how to generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, support qualitative and thematic data analysis they are not applicable to this work. Additionally, we used generative AI tools to modify scientific figures, edit software code and edit parts of the paper to improve readability. We have reviewed all AI-assisted work whether it is code, text or image edit to confirm their reliability. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

All three strategies share the generic pipeline described in Algorithm 1, and the complete experimental setup (datasets, splits, input representations, core models and optimization procedures) is summarized in Table 1. Hardware, hyperparameters for each strategy, evaluation metrics, feature-extraction and preprocessing details are given in Appendix A.5. All experiments rely on the publicly available TSB-UAD benchmark (Paparrizos et al., 2022b) and use the same 70/30 split as Sylligardos et al. (2023); when mentionned, results are averaged over 5 random seeds. Source code is provided as anonymized supplementary material.

REFERENCES

Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

Charu C. Aggarwal and Saket Sathe. Theoretical foundations and algorithms for outlier ensembles. *SIGKDD Explor. Newsl.*, 17(1):24–47, sep 2015. ISSN 1931-0145. doi: 10.1145/2830544.2830549. URL https://doi.org/10.1145/2830544.2830549.

Jérôme Antoni and Pietro Borghesani. A statistical methodology for the design of condition indicators. *Mechanical Systems and Signal Processing*, 114:290–327, 2019. ISSN 0888-3270. doi: https://doi.org/10.1016/j.ymssp.2018.05.012. URL https://www.sciencedirect.com/science/article/pii/S0888327018302632.

Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria A Zuluaga. Usad: Unsupervised anomaly detection on multivariate time series. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 3395–3404, 2020.

Paul Boniol and Themis Palpanas. Series2graph: Graph-based subsequence anomaly detection for time series. *Proc. VLDB Endow.*, 13(12):1821–1834, July 2020. ISSN 2150-8097. doi: 10.14778/3407790.3407792. URL https://doi.org/10.14778/3407790.3407792.

Paul Boniol, Michele Linardi, Federico Roncallo, Themis Palpanas, Mohammed Meftah, and Emmanuel Remy. Unsupervised and scalable subsequence anomaly detection in large data series. *The VLDB Journal*, March 2021a. ISSN 0949-877X. doi: 10.1007/s00778-021-00655-8. URL https://doi.org/10.1007/s00778-021-00655-8.

Paul Boniol, John Paparrizos, Themis Palpanas, and Michael J Franklin. Sand: streaming subsequence anomaly detection. *Proceedings of the VLDB Endowment*, 14(10):1717–1729, 2021b.

Paul Boniol, Qinghua Liu, Mingyi Huang, Themis Palpanas, and John Paparrizos. Dive into time-series anomaly detection: A decade review, 2024. URL https://arxiv.org/abs/2412.20512.

Paul Boniol, Ashwin K Krishna, Marine Bruel, Qinghua Liu, Mingyi Huang, Themis Palpanas, Ruey S Tsay, Aaron Elmore, Michael J Franklin, and John Paparrizos. Vus: effective and efficient accuracy measures for time-series anomaly detection. *The VLDB Journal*, 34(3):32, 2025.

Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. Lof: Identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, pp. 93–104, New York, NY, USA, 2000. ACM. ISBN 1-58113-217-4. doi: 10.1145/342009.335388. URL http://doi.acm.org/10.1145/342009.335388.

Sucheta Chauhan and Lovekesh Vig. Anomaly detection in ecg time signals via deep long short-term memory networks. In *2015 IEEE international conference on data science and advanced analytics (DSAA)*, pp. 1–7. IEEE, 2015.

Pavel Filonov, Andrey Lavrentyev, and Artem Vorontsov. Multivariate industrial time series with cyber-attack simulation: Fault detection using an lstm-based predictive data model, 2016.

Ahmed Fawzy Gad. Pygad: An intuitive genetic algorithm python library. *Multimedia Tools and Applications*, pp. 1–14, 2023.

Yifeng Gao, Jessica Lin, and Constantin Brif. Ensemble grammar induction for detecting anomalies in time series. In *Proceedings of the 23rd International Conference on Extending Database Technology, EDBT*, pp. 85–96, 2020.

Steve Goddard, Sherri K Harms, Stephen E Reichenbach, Tsegaye Tadesse, and William J Waltman. Geospatial decision support for drought risk management. *Communications of the ACM*, 46(1): 35–37, 2003.

Markus Goldstein and Andreas Dengel. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. *KI-2012: poster and demo track*, 1:59–63, 2012.

Mononito Goswami, Cristian Challu, Laurent Callot, Lenon Minorics, and Andrey Kan. Unsupervised model selection for time-series anomaly detection. *arXiv preprint arXiv:2210.01078*, 2022.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. Moment: A family of open time-series foundation models. In *International Conference on Machine Learning*, 2024.

Medina Hadjem, Farid Naït-Abdesselam, and Ashfaq Khokhar. St-segment and t-wave anomalies prediction in an ecg data using rusboost. In *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)*, pp. 1–6, 2016. doi: 10.1109/HealthCom.2016.7749493.

Mingyi Huang, Qinghua Liu, Paul Boniol, and John Paparrizos. Hydra: A multi-level hierarchy-driven approach for robust anomaly detection in time series. *Proceedings of the ACM on Management of Data*, 4(3):1–30, June 2026. doi: 10.1145/3802074.

Azal Ahmad Khan, Omkar Chaudhari, and Rohitash Chandra. A review of ensemble learning and data augmentation models for class imbalanced problems: Combination, implementation and evaluation. *Expert Systems with Applications*, 244:122778, 2024.

Chunggyeom Kim, Jinhyuk Lee, Raehyun Kim, Youngbin Park, and Jaewoo Kang. Deepnap: Deep neural anomaly pre-detection in a semiconductor fab. *Information Sciences*, 457:1–11, 2018.

Zheng Li, Yue Zhao, Nicola Botta, Cezar Ionescu, and Xiyang Hu. Copod: copula-based outlier detection. In *2020 IEEE international conference on data mining (ICDM)*, pp. 1118–1123. IEEE, 2020.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation Forest. In *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422, December 2008. doi: 10.1109/ICDM.2008.17. ISSN: 2374-8486.

Qinghua Liu and John Paparrizos. The elephant in the room: towards a reliable time-series anomaly detection benchmark. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2025. Curran Associates Inc. ISBN 9798331314385.

Qinghua Liu, Seunghak Lee, and John Paparrizos. Tsb-autoad: Towards automated solutions for time-series anomaly detection. *Proceedings of the VLDB Endowment*, 18(11):4364–4379, 2025.

Yue Lu, Renjie Wu, Abdullah Mueen, Maria A. Zuluaga, and Eamonn Keogh. Matrix profile xxiv: Scaling time series anomaly detection to trillions of datapoints and ultra-fast arriving data streams. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, pp. 1173–1182, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393850. doi: 10.1145/3534678.3539271. URL https://doi.org/10.1145/3534678.3539271.

Carl H. Lubba, Sarab S. Sethi, Philip Knaute, Simon R. Schultz, Ben D. Fulcher, and Nick S. Jones. catch22: CAnonical Time-series CHaracteristics. *Data Mining and Knowledge Discovery*, 33 (6):1821–1852, November 2019. ISSN 1573-756X. doi: 10.1007/s10618-019-00647-x. URL https://doi.org/10.1007/s10618-019-00647-x.

Helmut Lütkepohl, Markus Krätzig, and Peter CB Phillips. *Applied time series econometrics*. Cambridge university press, 2004.

Pankaj Malhotra, L. Vig, Gautam M. Shroff, and Puneet Agarwal. Long Short Term Memory Networks for Anomaly Detection in Time Series. In *ESANN*, 2015.

Melanie Mitchell. *An introduction to genetic algorithms*. MIT press, 1998.

David E Moriarty, Alan C Schultz, and John J Grefenstette. Evolutionary algorithms for reinforcement learning. *Journal of Artificial Intelligence Research*, 11:241–276, 1999.

M. Munir, S. A. Siddiqui, A. Dengel, and S. Ahmed. Deepant: A deep learning approach for unsupervised anomaly detection in time series. *IEEE Access*, 7:1991–2005, 2019. doi: 10.1109/ACCESS.2018.2886457.

Takaaki Nakamura, Ryan Mercer, Makoto Imamura, and Eamonn Keogh. Merlin++: parameter-free discovery of time series anomalies. *Data Min. Knowl. Discov.*, 37(2):670–709, January 2023. ISSN 1384-5810. doi: 10.1007/s10618-022-00876-7.

John Paparrizos, Paul Boniol, Themis Palpanas, Ruey S Tsay, Aaron Elmore, and Michael J Franklin. Volume under the surface: a new accuracy evaluation measure for time-series anomaly detection. *Proceedings of the VLDB Endowment*, 15(11):2774–2787, 2022a.

John Paparrizos, Yuhao Kang, Paul Boniol, Ruey S. Tsay, Themis Palpanas, and Michael J. Franklin. Tsb-uad: an end-to-end benchmark suite for univariate time-series anomaly detection. *Proc. VLDB Endow.*, 15(8):1697–1711, April 2022b. ISSN 2150-8097. doi: 10.14778/3529337.3529354. URL https://doi.org/10.14778/3529337.3529354.

John Paparrizos, Paul Boniol, Qinghua Liu, and Themis Palpanas. Advances in time-series anomaly detection: Algorithms, benchmarks, and evaluation measures. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '25, pp. 6151–6161, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400714542. doi: 10.1145/3711896.3736565. URL https://doi.org/10.1145/3711896.3736565.

Magali Parrino, Antoine Ajenjo, Emmanuel Remy, Pierre Stephan, and Paul Boniol. What Streaming Anomaly Detection Finds (and Misses) in Industrial Time Series. In *ECML PKDD 2026 - European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases : Industrial Track*, Naples, Italy, September 2026. URL https://inria.hal.science/hal-05659049.

Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, MLSDA'14, pp. 4–11, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450331593. doi: 10.1145/2689746.2689747. URL https://doi.org/10.1145/2689746.2689747.

Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. Anomaly detection in time series: a comprehensive evaluation. *Proc. VLDB Endow.*, 15(9):1779–1797, May 2022. ISSN 2150-8097. doi: 10.14778/3538598.3538602. URL https://doi.org/10.14778/3538598.3538602.

Sebastian Schmidl, Felix Naumann, and Thorsten Papenbrock. Autotsad: Unsupervised holistic anomaly detection for time series data. *Proceedings of the VLDB Endowment*, 17(11):2987–3002, 2024.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '19, pp. 2828–2837, 2019. ISBN 9781450362016. doi: 10.1145/3292500.3330672.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Emmanouil Sylligardos, Paul Boniol, John Paparrizos, Panos Trahanias, and Themis Palpanas. Choose wisely: An extensive evaluation of model selection for anomaly detection in time series. *Proc. VLDB Endow.*, 16(11):3418–3432, July 2023. ISSN 2150-8097. doi: 10.14778/3611479.3611536. URL https://doi.org/10.14778/3611479.3611536.

Emmanouil Sylligardos, John Paparrizos, Themis Palpanas, Pierre Senellart, and Paul Boniol. Msad: A deep dive into model selection for time series anomaly detection. *The VLDB Journal*, 34(6): 1–25, 2025.

Shreshth Tuli, Giuliano Casale, and Nicholas R. Jennings. Tranad: deep transformer networks for anomaly detection in multivariate time series data. *Proc. VLDB Endow.*, 15(6):1201–1214, February 2022. ISSN 2150-8097. doi: 10.14778/3514061.3514067.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL http://jmlr.org/papers/v9/vandermaaten08a.html.

Nicolas Vautier, Paul Caron, Nardi Xhepi, Félicie Bizeul, Manel Boumghar, Christophe Degouy, and Paul Boniol. From benchmarks to production: Transferring time series anomaly detection methods for electricity production monitoring. In *IEEE International Conference on Data Engineering (ICDE)*, 2025.

Phillip Wenig and Thorsten Papenbrock. Series2graph++: Distributed detection of correlation anomalies in multivariate time series. In *Big Data Analytics and Knowledge Discovery: 26th International Conference, DaWaK 2024, Naples, Italy, August 26–28, 2024, Proceedings*, pp. 225–230, Berlin, Heidelberg, 2024. Springer-Verlag. ISBN 978-3-031-68322-0. doi: 10.1007/978-3-031-68323-7_17. URL https://doi.org/10.1007/978-3-031-68323-7_17.

David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.

Yiyuan Yang, Chaoli Zhang, Tian Zhou, Qingsong Wen, and Liang Sun. Dcdetector: Dual attention contrastive representation learning for time series anomaly detection. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, pp. 3033–3045, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701030. doi: 10.1145/3580305.3599295.

Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh. Matrix profile i: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pp. 1317–1322, 2016. doi: 10.1109/ICDM.2016.0179.

Seniha Esen Yuksel, Joseph N Wilson, and Paul D Gader. Twenty years of mixture of experts. *IEEE transactions on neural networks and learning systems*, 23(8):1177–1193, 2012.

A APPENDIX

This section contains additional resources on the literature review (Section A.1 and Section A.2), the problem formulation (Section A.3), the experimental setup (Section A.5), and additional experimental analysis (from Section A.7).

A.1 TIME SERIES ANOMALY DETECTION LITERATURE REVIEW

Anomaly detection in time series is a crucial task for many relevant applications (Parrino et al., 2026). Therefore, several methods have been proposed in the literature (Boniol et al., 2024). One type of anomaly detection method is *distance-based methods*, which analyse subsequences by utilizing distances to a given model to detect anomalies. In this category, we can identify three sub-categories. The first is *discord-based*. These methods focus on analysing subsequences for the purpose of detecting anomalies in time series, primarily by utilizing nearest neighbour distances among subsequences (Yeh et al., 2016). The second sub-category is *proximity-based*. These methods focus on estimating the density of specific types of subsequences to either extract normal behaviour or isolate anomalies. Since a subsequence can be seen as a multidimensional point (with the number of dimensions corresponding to the subsequence length), general outlier detection methods can be applied for time series anomaly detection (Breunig et al., 2000). The last category is *clustering-based*, which comprises methods that utilize the distance to a given clustering partition to detect anomalies. In this sub-category, NormA, which first clusters data to obtain the normal behaviour (Boniol et al., 2021a;b), has demonstrated strong performance.

While the previously mentioned methods compute their anomaly score based on distances using raw time series elements (such as subsequences), *density-based methods* focus on detecting recurring or isolated behaviours by evaluating the density of points or subsequences in a specific representation space. This category can be divided into four sub-categories, namely *distribution-based*, *graph-based*, *tree-based*, and *encoding-based*. Among them, Isolation Forest (Liu et al., 2008), a *tree-based* method grouping points or subsequences into different trees, and Series2Graph, a *graph-based* method that converts the time series into a graph to facilitate the detection of anomalies (Boniol & Palpanas, 2020), have been shown to work particularly well for the time series anomaly detection task (Boniol & Palpanas, 2020).

Furthermore, *forecasting-based methods*, such as recurrent neural network-based (Malhotra et al., 2015) or convolutional network-based (Munir et al., 2019), have been proposed for this task. These methods use the past values as input, predict the following one, and use the forecasting error as an anomaly score. Such methods are usually trained on time series without anomalies, or make the assumption that the anomalies are significantly less frequent than normal behaviours.

Finally, *reconstruction-based methods*, such as auto-encoder approaches (Sakurada & Yairi, 2014), are trained to reconstruct the time series and use the reconstruction error as an anomaly score. As both forecasting and reconstruction-based categories detect anomalies using prediction errors (either forecasting or reconstruction error), we can group them into *prediction-based methods*.

A.2 ANOMALY TYPES

Point anomalies refer to data points that deviate remarkably from the rest of the data. Similarly, *contextual* anomalies refer to data points within the distribution's expected range (in contrast to point anomalies) but that deviate from the distribution's expected behavior given a specific context (e.g., a window). *Collective* anomalies refer to sequences of points that do not repeat a typical (previously observed) pattern. The first two categories, namely, point and contextual anomalies, are referred to as *point-based* anomalies, whereas *collective* anomalies are referred to as *subsequence* anomalies.

A.3 DETERMINISTIC VERSUS STOCHASTIC LEARNING STRATEGIES

Deterministic models: ML and GP approaches. The deterministic case appears when a function embodies the model, we will note $M_\theta : X \mapsto M_\theta(X) \in \mathbb{R}^N$. In practice, M_θ is implemented as a network, either an MLP or a CNN, depending on whether the data are features or a raw time-series window. More formally, the model generating the weights can be defined with $\mathcal{M}_\theta(X) := \delta_{M_\theta(X)}$ where $\delta_{M_\theta(X)}$ is the Dirac distribution at the network output $M_\theta(X)$. The optimization problem can

Algorithm 1 Generic Pipeline for Supervised Ensembling

Require: Dataset of time series $\mathcal{D}$, window size ℓ, number of detectors N, model $\mathcal{M}_\theta$, training strategy
Ensure: Trained model parameters θ_*, anomaly scores for test time series

```
Split D into D_train, D_val, D_test
for each time series T in D_train ∪ D_val do
    if using raw time series then
        Collect the first window of size ℓ of T
    else if using features then
        Extract F features from T as input vector
    end if
end for
Train model M_θ on D_train and early stop on D_val
Obtain final parameters θ_*
for each test time series T in D_test do
    if using raw time series then
        Select the first window of size ℓ of T and predict the weight vector W
    else if using features then
        Extract features and predict the weight vector W
    end if
    Retrieve detector scores matrix S ∈ R^{L×N}
    Compute ensemble score: Ŝ = S × W̃
end for
```

thus be reformulated with

$$\theta_* = \arg\max_{\theta \in \Theta} \mathbb{E}_{(X,S,Y)} \Big[Acc\big(S \times \widetilde{M_\theta}(X), Y\big) \Big]. \tag{5}$$

To solve this optimization task, we want to use gradient methods because they are simple to implement and effective in most cases. However, Acc (AUC-PR or VUS-PR) is not differentiable, and facing this issue, we consider two approaches: Machine Learning and Genetic Algorithms. The former chooses to change the metric to a differentiable proxy, such as the *Mean Squared Error* (MSE). The latter instead decides to keep the metric of interest and forget about gradient methods for keeping the best random mutation of θ over generations to target θ_*.

Stochastic models: RL approach. The main problem described in Section 2.4 is exactly the stochastic case (the deterministic one is simply a particular case). More specifically, the RL strategy outputs a normal distribution specific to each individual detector, from which the weights are later sampled. We will see in the section dedicated to RL how we can, surprisingly, take the gradient of the main objective and how RL introduces not only a stochastic approach, but an entire paradigm.

A.4 Generic Pipeline

This section contains additional resources on the generic pipeline considered in our experimental evaluation.

A.5 Experimental Setup

Technical setup. All experiments were conducted on a server with Cascade Lake Intel Xeon 5217 8 cores, 3-3.7GHz CPU, and Nvidia V100 32GB GPU.

Global Parameter settings. We use the same 70/30 split of the TSB-UAD benchmark as in Sylligardos et al. (2023) for comparison fairness. All methods were early-stopped if there was no evaluation improvement for half the total evaluation number.

ML parameter settings. On the features, the MLP was trained with a learning rate of 10^{-2} and a batch size of 32. As for the raw data, the CNN is made of 2 blocks, a GAP layer, and its kernel size is 3. It was trained with a learning rate of 10^{-4} and a batch size of 64. Both models were trained for 50 epochs.

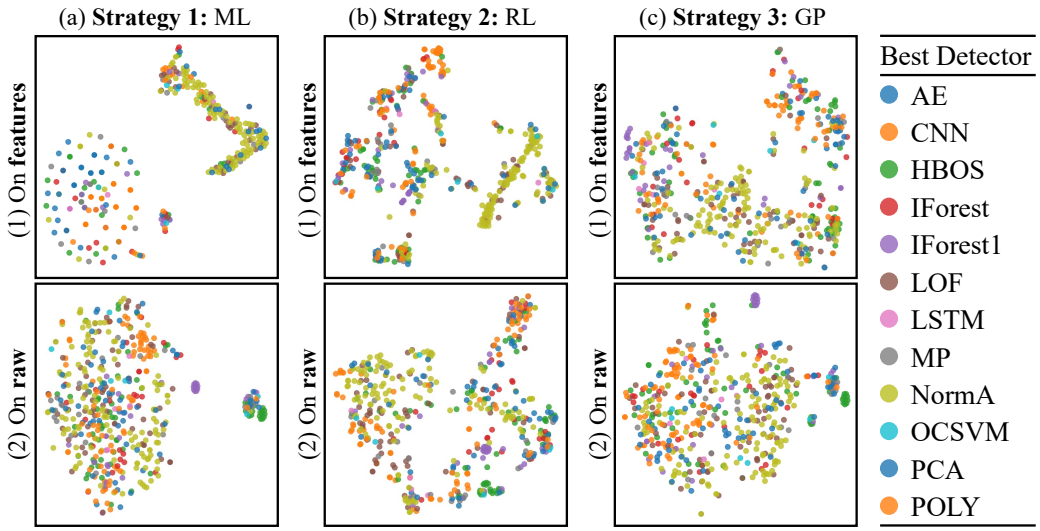


Figure 8: Per strategies 2D-t-SNE (van der Maaten & Hinton, 2008) of the weights for each time series labelled according to the best individual detector.

RL parameter settings. Considering the parameters naming of Stable-Baselines3, for both features and raw data, we fix the learning rate to 3.10^{-3}, run the methods with 720000 total time steps (total amount of states visited, windows/features in our formulation), 20 epochs every 1024 steps, and a batch size of 128. The experiments on the raw data are evaluated 10 times along the way, and this number is raised to 100 for the features (since they were less time-consuming). Weights W are clipped to 0-1. This decision is motivated empirically to favour convergence stability. We provide additional experiments with unclipped weights.

GP parameter settings. Both MLP and CNN were trained for 20 generations, 20 solutions, 5 parents mating, and a random sample of 256 time series to evaluate the fitness of each individual (for time efficiency purposes).

Evaluation metrics. For anomaly detection accuracy, we consider the measure VUS-PR (Paparrizos et al., 2022a). As detailed in Boniol et al. (2025), this metric is specifically designed to account for imprecise, noisy, misaligned, or time-delayed labels, ensuring that our reported performance is robust to the inherent ambiguity of anomaly timestamps. Concerning execution time evaluation, we mean by "execution time" the time required to perform all the training/testing phases (*i.e.*, the entire training pipeline). Finally, we will look at the inference time of each strategy, *i.e.*, the time needed for a model (MLP or CNN) to output a vector for a given input and a given paradigm (ML, RL, GP).

Feature extraction and preprocessing. Feature extraction is performed using the official pycatch22 library (Lubba et al., 2019), ensuring a standardized, reproducible, and computationally efficient implementation. While feature-based representations naturally map variable-length inputs to a fixed dimensionality, raw time series require specific segmentation to ensure consistent input sizes. To address this without introducing zero-padding or truncation artifacts, we segment series into windows of fixed length ℓ. In standard cases, these windows are non-overlapping. However, when the time series length L is not a multiple of ℓ, we introduce a partial overlap between the first two windows only. This adjustment shifts the segmentation grid of the subsequent windows such that the final window aligns exactly with the last time point of the series, preserving data integrity at the boundaries.

A.6 How Accurate is Supervised Ensembles? (Part 2)

We continue our analysis with Figure 8, which examines through t-SNE (van der Maaten & Hinton, 2008) plots of the weights whether time series best detected by the same detector (one color per detector) receive similar weights across supervised ensembling strategies. From this perspective, the feature-based RL strategy yields the most effective separation, producing a compact representation in which time series best analyzed by the same detector tend to cluster together. Moreover, we

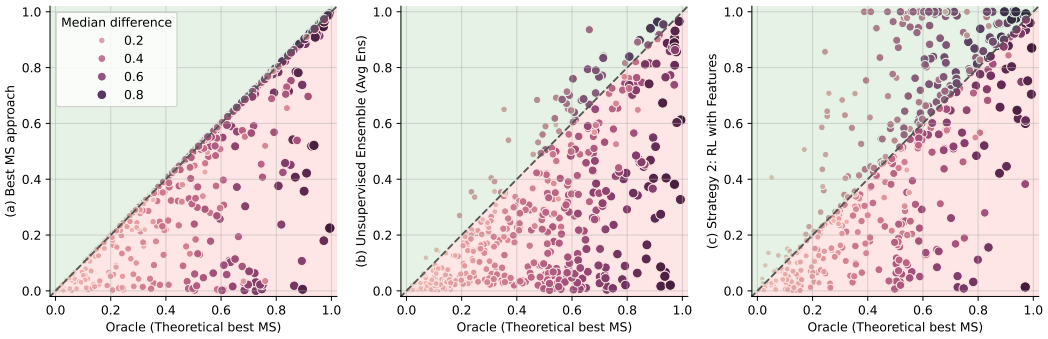


Figure 9: (left) Best model selection (Sylligardos et al., 2023), (middle) Unsupervised Ensemble, (right) Strategy 2 (with features) against the *Oracle*. For every time series, we represent the difference between the first and the median of all other detectors with the disk size and color.

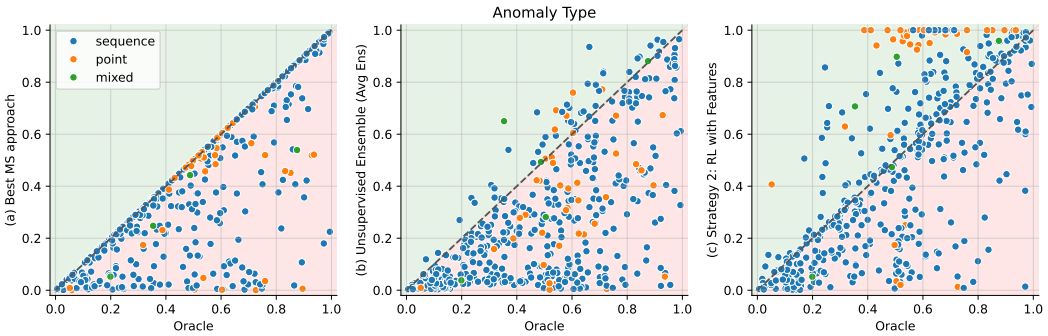


Figure 10: (left) Best model selection (Sylligardos et al., 2023), (middle) Unsupervised Ensemble, (right) Strategy 2 (with features) against the *Oracle*. For every time series, we represent the type of anomalies it contains with a specific colour.

observe that features as input representation for all strategies leads to more compact representations. These two observations suggest that RL on features captures detector-specific characteristics more consistently than the other supervised approaches. All t-SNE are computed with a perplexity of 30 and without prior PCA (Abdi & Williams, 2010).

A.7 Do Supervised Ensembles Really Ensemble? (Part 2)

In Figure 9, we focus specifically on time series on which the ensemble outperforms the *Oracle*. First, when considering the three plots of Figure 9, we observe that small, clear disks are more often on the left. Big, dark disks are almost always on the right, because it is rare for the first two detectors to perform very well (small disk on the right), and impossible for very well-performing detectors to coexist with a low value of *Oracle*.

Overall, Figure 9 shows that time series with no single best detector are leading to poor ensemble performance (both unsupervised and supervised). Most of the time series for which the supervised ensembling approach outperforms the *Oracle* are associated with a strong single best detector.

A.8 Point Anomalies versus Sequence Anomalies

Figure 10 depicts a per-time series comparison between (left) Best model selection (Sylligardos et al., 2023), (middle) Unsupervised Ensemble, (right) Strategy 2 (with features) against the *Oracle*. Figure 10 focuses explicitly on the impact of anomaly types on the performance of the supervised ensembling approach (Strategy 2).

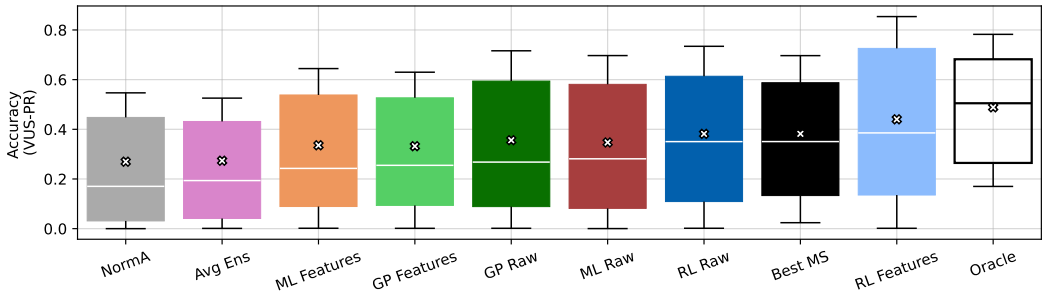


Figure 11: Overall accuracy averaged across 5 random seeds of all strategies, *Best MS*, *Avg Ens*, and *Oracle* in the *in-distribution* case with weights in $[0, 1]$ for all strategies.

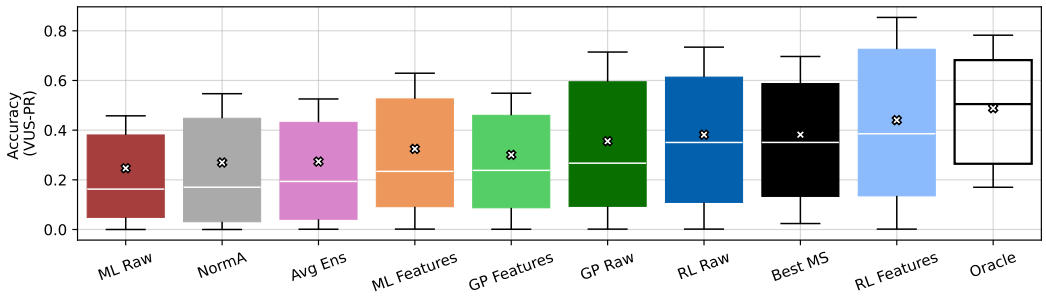


Figure 12: Overall accuracy averaged across 5 random seeds of all strategies, *Best MS*, *Avg Ens*, and *Oracle* in the *in-distribution* case with weights clipped in $[-1, 1]$ for ML and GP and clipped in $[0, 1]$ for RL for stability purposes.

A.9 In-Distribution Accuracy Evaluation

Figure 11 displays an alternative form of the results illustrated in Figure 4. In addition to the median and distribution of performance, this Figure also shows the mean performance of the strategies. Figure 12 displays the same type of information, but with the ML and GP strategies producing weights in $[-1, 1]$ while still in $[0, 1]$ for RL, as it is unstable otherwise. This instability is likely due to the variance introduced in the reward signal when subtracting detector outputs is allowed, destabilizing the policy gradient updates. These two plots in the *in-distribution* setting show that clipping yields better results. The clipped weights can sometimes be beneficial for the ensembling, but overall, they remove the poor-performing detectors, leading to better performances as illustrated in Figure 13.

A.10 Out-of-Distribution Accuracy Evaluation

Figure 13 displays an alternative form of the results illustrated in Figure 4. In addition to the median and distribution of the performances, this plot also shows the percentage by which a given strategy outperforms the *Avg Ens* method. Figure 14 displays the same type of information, but with the GP approach producing weights in $[-1, 1]$. These two plots in the *out-of-distribution* setting demonstrate the benefits of clipping. We might have thought that the sparsity induced by the clipping would lead to weak performance in the *out-of-distribution* scenario, as we saw in Q3 that denser activations perform better. However, clipping is moderated enough not to introduce to much sparsity in this case and improve the results as illustrated by Figure 13.

Figure 15 depicts the detailed results of the experiment on the *out-of-distribution* case, whose aggregated results are displayed in Figure 4. For a given dataset, the number of points per box plot is the number of series in the dataset. We observe that the *Oracle* is almost always more accurate than the ensembling and selecting methods, except for ECG and Occupancy, where the RL on features

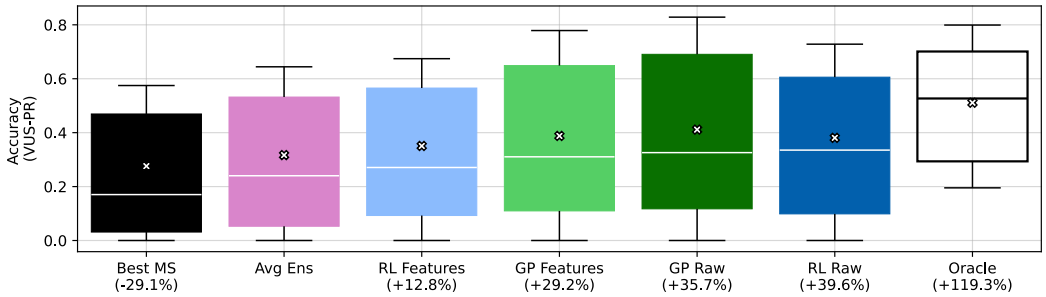


Figure 13: Overall accuracy of all strategies, *Best MS*, *Avg Ens*, and *Oracle* in the *out-of-distribution* case with weights clipped in $[0, 1]$ for all strategies.

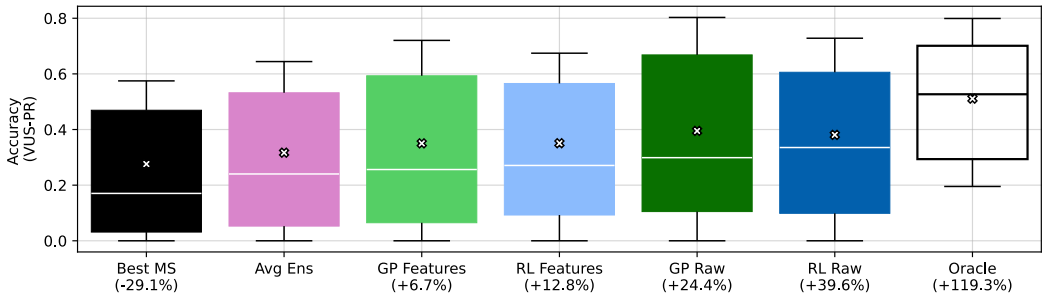


Figure 14: Overall accuracy of all strategies, *Best MS*, *Avg Ens*, and *Oracle* in the *out-of-distribution* case with weights in $[-1, 1]$ for GP and clipped in $[0, 1]$ for RL for stability purpose.

supervised ensembling is better. This is also the case for the YAHOO dataset, where the GP strategy applied to both features and raw data achieves higher accuracy.

For the sake of transparency, we also represent in Figure 16 the analogue of Figure 15 with the weights clipped in $[0, 1]$ for all strategies.

A.11 Weight Distribution per Model

Figure 17 shows the distribution of detector weights assigned by each model in the *in-distribution* setting; the clipped version of this plot sets all negative weights to zero. The ML- and GP-based strategies assign both positive and negative weights, whereas the RL-based models exclusively produce positive weights.

We also observe that some strategies consistently exclude specific detectors. For example, in the ML Features model, the LSTM and POLY detectors consistently receive zero weight, while the RL Raw model excludes four detectors entirely.

Finally, the spread of the weight distributions differs across strategies. The GP Raw model exhibits very compact distributions, indicating more stable weight assignments, while the ML Raw model shows the widest variability, suggesting greater sensitivity to different time series.

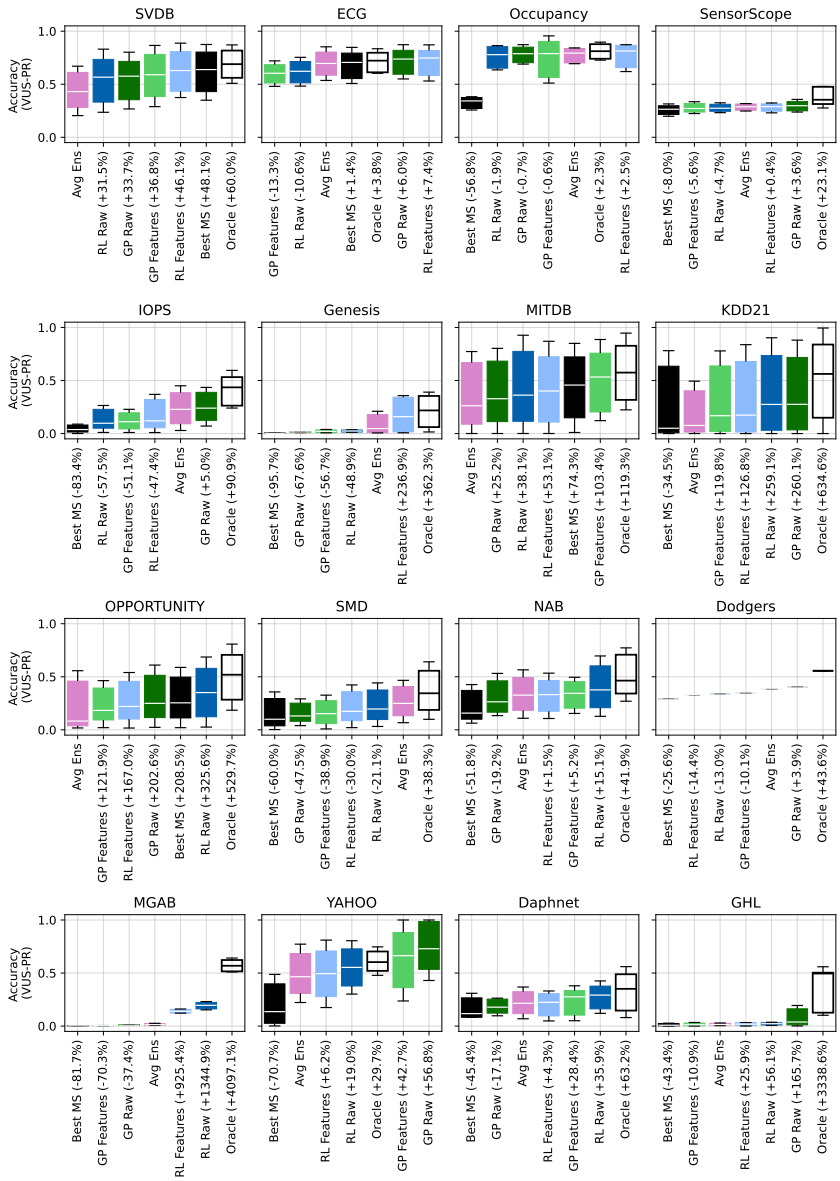


Figure 15: Per-dataset accuracy of all strategies, *Best MS*, *Avg Ens*, and *Oracle* in the *out-of-distribution* case with weights in $[-1, 1]$ for GP and clipped in $[0, 1]$ for RL for stability purpose.

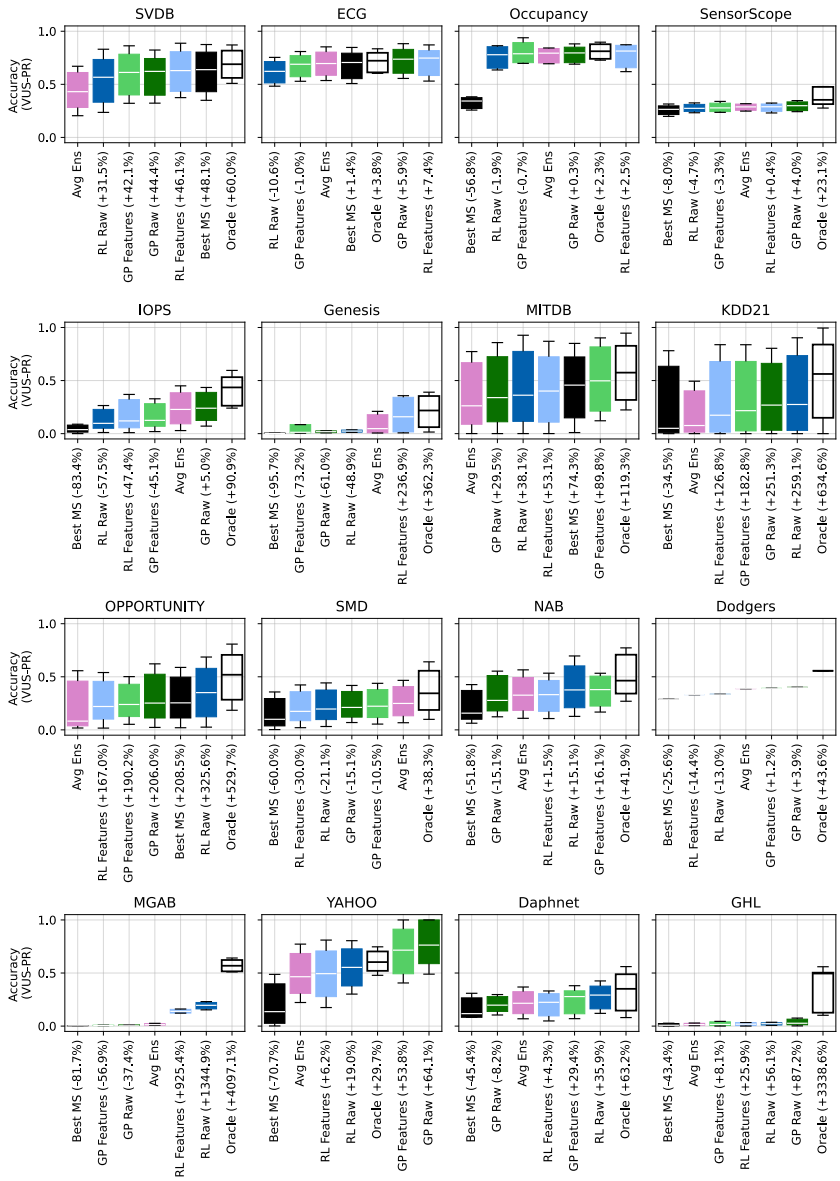


Figure 16: Per-dataset accuracy of all strategies, *Best MS*, *Avg Ens*, and *Oracle* in the *out-of-distribution* case with all strategies weights clipped in $[0, 1]$.

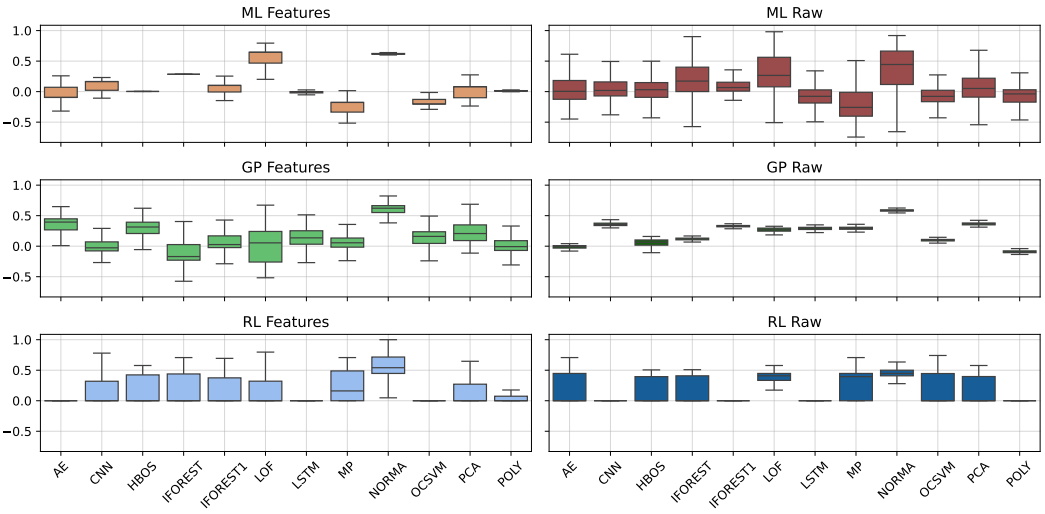


Figure 17: Weight distribution of detectors across different models. Each boxplot represents the variability of weights assigned to detectors by a given model in the *in-distribution* setting.