Streaming Time Series Anomaly Detection

Christina Karalka • AEM: 145 • EMAIL: kchristi@csd.auth.gr

1 INTRODUCTION

In the realm of time series anomaly detection (AD), adapting to continuously arriving data points in a streaming setting is crucial for timely and accurate detection of anomalies. This work examines the effectiveness of LSTM-based autoencoder architectures trained in an unsupervised manner. Specifically, two streaming adaptations of such a model originally designed for offline settings are proposed. The first approach operates independently on each arriving batch of data, while the second incorporates incremental learning and a concept drift detection mechanism to adapt to changes in the data stream. The models are evaluated on univariate time series from various domains and levels of normality, comparing their performance to two state-of-the-art (SOTA) online AD methods [1].

1.1 Preliminaries

A **Time Series** is a sequence $T = \{(x_t, y_t), t \in [0, L]\}$. Each data point $x_t \in \mathbb{R}^m$ is recorded at timestep t and is accompanied by labels $y_t \in \{0,1\}^m$. Here, x_t consists of m variables and $y_t[i]$ indicates whether $x_t[i]$ is a normal (0) or abnormal (1) observation. This work tackles univariate time series, meaning $m = 1$. The length of the series L can be infinite in the case of data streams.

Furthermore, a series might exhibit changes in normal behavior and anomalous patterns over time. In accordance with [1], such scenarios are simulated by concatenating multiple smaller series denoted as $T = T^1 \oplus \dots \oplus T^k$. Here, k is referred to as the **normality** level of the resulting time series. This approach is used to assess the model's capacity to adapt to concept drift.

Additionally, T can be segmented into **subsequences** of a fixed size W_{sub} through a sliding window. A subsequence of T is denoted as $T_{i,W_{sub}} = \{(x_t, y_t), t \in [i, i + W_{sub} - 1]\}$. Whether these subsequences are overlapping or not is determined by the step size S_{sub} used during their generation, with $S_{sub} < W_{sub}$ and $S_{sub} = W_{sub}$ defining overlapping and non-overlapping segmentations respectively.

2 METHODOLOGY

LSTM-based autoencoders are a suitable approach for time series AD [2], [3]. An autoencoder is a neural network designed to learn useful representations of data by learning to reconstruct its input during training. It comprises two parts: an encoder, which maps the input data to a latent space representation, and a decoder, which reconstructs the original data from this representation. The objective is to minimize the difference between the input and the reconstructed output [2].

LSTM (Long Short-Term Memory) models are a type of recurrent neural network (RNN) that excel at capturing long-term dependencies in sequential data [2]. They serve as the encoder and decoder in LSTM-based autoencoders, effectively capturing temporal patterns and reconstructing sequential data. This combination enhances the model's ability to learn normal behavior in time series data. Anomalies are identified when the reconstruction error exceeds a certain threshold, indicating that the input data deviates from the learned normal patterns [2]. This work utilizes such a network called EncDec-AD [4], proposed for offline AD, as a static baseline. Additionally, two variants are proposed, EncDec-AD-Batch and OnlineEncDec-AD, to train and evaluate this model in online settings.

2.1 EncDec-AD Baseline for Offline Anomaly Detection

The input series T is first segmented into overlapping subsequences with $S_{sub} = 1$. Firstly, a single-layer LSTM encoder generates vector representations for each input $T_{i,W_{sub}}$. Specifically, for each timestep $t \in [i, i + W_{sub} - 1]$, the encoder's hidden state $h_E(t)$ is computed using the input point x_t and the previous hidden state $h_E(t-1)$. The encoder's final hidden state $h_E(i + W_{sub} - 1)$ at the end of $T_{i,W_{sub}}$ is then used as the initial state $h_D(i + W_{sub} - 1)$ for the decoder.

[1] The entire codebase and results are available at https://github.com/ChristinaK97/StreamingTimeSeriesAnomalyDetection

The decoder, which is a single-layer LSTM followed by a linear layer, reconstructs $T_{i,W_{sub}}$ in reverse order. This means that the target sequence for reconstruction is $\{x_{i+W_{sub}-1}, \dots, x_{i+1}, x_i\}$. During training, at each timestep t, the decoder uses $h_D(t)$ and x_t to obtain its next hidden state $h_D(t-1)$ and predict $\widehat{x_t}$ for the target x_t. During inference, the prediction $\widehat{x_t}$ is used instead. Overall, the output predictions are generated by the linear layer from the decoder's hidden states.

The objective of this network is to minimize its reconstruction error. This is defined as the mean squared error between the original x_t and the reconstructed points $\widehat{x_t}$ across the input subsequences. During inference, the errors are used to compute an anomaly score a_t for each point x_t in the test series. Specifically, a_t is the mean of all reconstruction errors of x_t across all overlapping subsequences in which x_t appears.

The original paper describes EncDec-AD's training in a semi-supervised manner, where the model is trained exclusively on normal data points, and an anomaly threshold is determined using a labeled validation set. In contrast, considering that anomalies constitute the minority class [3], an unsupervised approach is adopted instead, using the initial region of T [1]. Specifically, 20% of the first subseries T^1 of T is withheld for training, in order to avoid using the entirety of T^i $(i = 1, \dots)$ for this purpose. Training lasts E_{TRAIN} epochs.

2.2 EncDec-AD-Batch

Similarly to [5], a dynamic baseline of EncDec-AD is established by independently processing batches of data points. This streaming setup is simulated by dividing T into $b = 10$ consecutive and non-overlapping batches. For each arriving batch, a new model is trained for E_{TRAIN} epochs using 20% of the initial points. Subsequently, it is employed to calculate anomaly scores for the remaining data in that batch. These outputs are stored, and the trained model is discarded upon the arrival of the next batch.

2.3 OnlineEncDec-AD

This work proposes an adaptation of EncDec-AD for online operation. Specifically, **incremental learning (IL)** is applied to adapt the model without complete retraining, and a **concept drift detection (CDD)** mechanism triggers model reset when necessary. Additionally, a **dynamic anomaly threshold** (DAT) is recalculated each time the model is updated. This framework draws inspiration from prior research on LSTM-based autoencoders for anomaly detection in streaming data, such as StrAEm++DD [6] and LSTM-AE [7].

We maintain the same LSTM encoder-decoder architecture and **initialize** an EncDec-AD model with 20% of points from the first input series T^1. The remaining points arrive one-by-one and are stored in a buffer with a capacity of W_{pred}. The model makes **predictions** either when the buffer reaches full capacity or when the CDD's conditions are met. A point $x_{t'}$ is deemed anomalous if its score $a_{t'}$ exceeds the current threshold θ_t (where $t' > t$).

DAT is calculated upon model initialization, reset and IL step on a training set X_{train} at some timestep t. Specifically, after training, the model outputs the reconstruction errors of X_{train} and θ_t is determined by the p −percentile of these losses calculated with linear interpolation. This statistical measure determines the value below which p% of the errors fall. For example, for $p = 92$, 92% of the error values are below the threshold. This ensures precise anomaly identification based on the evolving distribution of reconstruction errors.

To strike a balance between continuous model adaptation and computational efficiency, the execution of **IL** is regulated by specific activation conditions and a fixed-sized queue. The FIFO queue wnd_{IL} with capacity W_{IL} maintains the most recent points from the data stream. An IL step is triggered when either the queue reaches full capacity or when a specific percentage, q_{IL}, of its oldest points have been replaced by newer data. During an IL step, the model undergoes further training using the points in wnd_{IL} for a reduced number of epochs E_{IL} compared to the complete retraining epochs, E_{TRAIN}. Furthermore, DAT is recalculated using wnd_{IL}.

Timely identification and response to **concept drift** are crucial for maintaining model performance abrupt changes in data distribution. To that end, as illustrated in Figure 1, two non-overlapping windows wnd_{ref} and wnd_{drift} are maintained, each with a fixed size W_{drift}. wnd_{ref} is sequentially followed by wnd_{drift} so that the latter holds newer data points. Specifically, upon arrival, a new point x_t is stored in wnd_{ref} if capacity permits, otherwise it is directed to wnd_{drift}. Once both windows are full, the model calculates their reconstruction errors for statistical comparison.

It is anticipated that a concept drift will degrade model performance resulting in notable differences in reconstruction errors. We apply the Kolmogorov-Smirnov (KS) test with a predefined significance level s_{KS} to determine whether the errors from the two windows originate from different distributions. A significant result prompts the current model to predict the points stored in the buffer. Subsequently, wnd_{drift} is used to train a new model and update DAT. The windows wnd_{ref} and wnd_{drift} are cleared, along with wnd_{IL}, since its points are considered outdated. Conversely, in the absence of concept drift, points from wnd_{drift} are transferred to a new wnd_{ref} and the processing of the stream resumes.

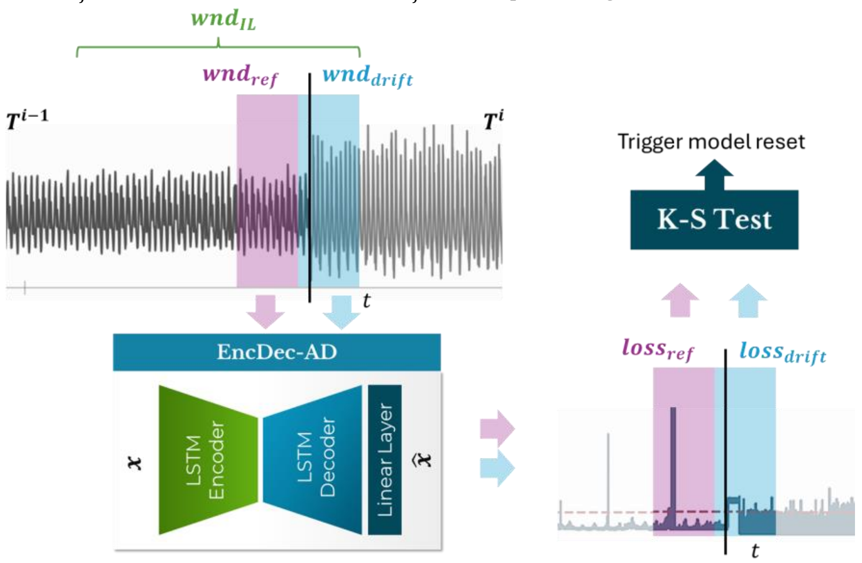


Figure 1: OnlineEncDec-AD Concept Drift Detection mechanism triggers model reset at timestep t

3 EXPERIMENTAL EVALUATION

3.1 Experimental Settings

Baselines: This study leverages an open-source implementation of the **EncDec-AD** from GitHub[2]. Table 1 summarizes the hyperparameter values for the three EncDec-AD variants. In addition to the **SAND** algorithm [5], the proposed models are also compared to one of the SOTA models for real-time AD that inspired our Online variant. **LSTM-AE** [7] employs 3-layered LSTMs for both the encoder and decoder. It initializes the model and performs incremental training per batch, while utilizing two estimators for predictions. Anomalies are detected if their reconstruction error exceeds thresholds derived from a rolling moving average and the 90-percentile value. Furthermore, a CDD mechanism triggers a model reset when the rate of predicted anomalies surpasses a predefined threshold (30%), leveraging the adaptive sliding window (ADWIN) technique [8]. For both SAND and LSTM-AE, we largely adopt the hyperparameter values documented in their respective papers [5], [7] and codebases [3] [4].

Table 1: EncDec-AD models hyperparameter values

Group	**Hyperparameter**	**Notation**	**Value**
Input	Subsequence length	W_{sub}	Dataset dependent
Model Initialization	Initial training %	–	20%
Model Architecture	EncDec-AD latent representation dim	–	32
Predictions	Predict buffer capacity	W_{pred}	2000
Model Training	Training epochs	E_{TRAIN}	40
	Learning Rate	–	10^{-3}
Dynamic Anomaly	Percentile %	p	92%

[2] EncDec-AD original code: https://github.com/KDD-OpenSource/DeepADoTS/blob/master/src/algorithms/lstm_enc_dec_axl.py

[3] TSB-UAD repository: https://github.com/TheDatumOrg/TSB-UAD/

[4] LSTM-AE repository: https://github.com/redsofa/streaming_anomaly_detection

Threshold (DAT)			
Incremental Learning (IL)	IL epochs	E_{IL}	10
	IL queue (wnd_{IL}) capacity	W_{IL}	1000
	IL queue replacement %	q_{IL}	50%
Concept Drift Detection (CDD)	Reference window (wnd_{ref}) size	W_{drift}	200
	Drift window (wnd_{drift}) size	W_{drift}	200
	KS test significance level	s_{KS}	0.001

Datasets[5]: The TSB-UAD benchmark suite [1] serves as the evaluation framework. It includes over 16 public datasets, from which one test series was randomly selected, with a preference for samples of larger lengths. For normality 1 experiments, a maximum of the initial 100k points were retained. For higher normality experiments, 50k points from each subseries T^i were kept. This approach was taken both due to resource constraints and to mitigate the discrepancy in lengths across the datasets.

Table 2 presents the results for all 16 single normality datasets (**k=1**). Due to the high number of combinations required for constructing higher normality series (e.g., 120 combinations for k=2), datasets where the models exhibited poor performance (e.g., AUC close to or lower than 50%) were excluded from the selection process if this trend was also observed in k>1 experiments, such as in the Daphnet and Dodgers datasets. The selected combinations for **k=2** are reported in Table 3. The selection was based on characteristics such as similar subseries lengths (C1, C2, C7), a similar number of anomaly points (C2, C5, C8, C9, C11) to ensure that results are not dominated by the performance in one subseries, and various trends in anomaly ratios between the two subseries.

For **k=3** (Table 4), triplets are generated by combining pairs of normality 2 series that share a common subseries. This ensures that the triplets maintain desired characteristics, such as similar lengths. Each triplet's subseries are then sorted once in ascending (↑) and once in descending (↓) order of their anomaly ratios (AR), and both resulting normality 3 series are used for experiments.

Evaluation Metrics: A discrepancy has been identified in the problem formulation among the compared models. The SAND algorithm, as well as the Offline and Batch variants of EncDec-AD, assign anomaly scores that establish a ranking of detected anomalies. In contrast, the Online variant and the LSTM-AE model classify points based on dynamic thresholds, resulting in binary outputs. Due to these differences, we adopt metrics such as AUC and F1 score, Recall and Precision **within the anomalous class**, which are conventional for classification scenarios. Metrics like Precision@k, typically used for ranking formulations, are not applicable in our context. To compute measures requiring binary predictions (F1, Recall etc.) for the first group of models, a point is deemed anomalous if its score exceeds three standard deviations from the mean value, following the setup of the TSB-UAD suite.

Computational Resources: The experiments were performed on a machine with a 12-core Intel CPU. The EncDec-AD models were developed using the PyTorch library, and CUDA acceleration was facilitated by a single RTX 3050 4 GB GPU.

3.2 Normality 1 Results

As shown in Table *2* and Figure 2, the Offline and Batch variants of EncDec-AD exhibit comparable performance for single normality series. Online outperforms Offline in 46% of cases for AUC and 56% for F1. Additionally, Online outperforms Batch in only 31% of cases for AUC and 56% for F1, indicating some confusion in the absence of concept drifts. As illustrated in Figure 2a, Online achieves higher recall for anomalies but also has a higher false positive rate.

Overall, the EncDec-AD models outperform other online baselines. When compared to the SAND algorithm, some exceptions are datasets such as Daphnet, Dodgers and MITDB that proved universally challenging for the autoencoder models with AUC scores close to or below 50%. Additionally, LSTM-AE, a model similar to Online, is outperformed by it in 62% of cases for AUC and 75% for F1.

[5] Statistics about the selected datasets are presented in the notebook “StreamingAnomalyDetectionNotebook.ipynd”

Table 2: Results for single normality (k=1) series. Dark blue (resp. green) denotes the highest AUC (resp. F1) score achieved for each dataset, while light blue denotes the second best score. AR denotes the anomaly ratio of the dataset

DATASET	SAND		ENCDEC-AD						LSTM-AE		
			Offline		Batch		Online				
	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AR
DAPHNET	79.00		40.78	1.16	27.25		48.27	5.45	48.97	3.87	5.37
DODGERS	75.90	11.60	49.67	1.21	55.74	1.10	46.01	4.75	49.88	1.47	11.1
ECG	98.78	49.47	77.40	14.97	67.87	14.69	72.17	33.17	61.45	21.68	4.6
GHL	50.94		65.66		99.02	16.77	78.44	1.32	49.01	0.00	0.14
GENESIS	0.26		96.86	18.30	92.34		91.40	6.08	91.55	10.50	0.31
IOPS	52.23		72.73	33.99	72.28	25.19	73.25	2.53	74.70	25.71	0.46
KDD21	88.52	12.54	70.77	8.69	67.65	8.92	62.72	5.16	49.41	1.22	0.78
MGAB	66.46		67.67	2.07	53.90	1.77	51.71	0.69	51.13	0.61	0.2
MITDB	98.13	56.02	37.83	1.21	39.91	1.51	46.36	1.51	49.10	1.81	1.4
NAB	42.79		47.12	2.26	56.79	1.20	61.60	26.60	56.87	21.16	9.84
OPPORTUNITY	86.41		99.85	72.23	98.04	70.73	76.26	8.00	48.42	0.47	2.23
OCCUPANCY	50.33		98.32	0.34	68.46		66.05	43.64	52.66	13.46	21.2
SMD	68.39		67.77	11.67	80.53	33.92	74.65	39.53	50.49	3.92	9.46
SVDB	49.03	4.71	66.36	13.93	74.61	5.09	54.47	24.35	51.69	12.44	28.9
SENSORSCOPE	55.79		43.15	4.32	85.68		79.23	48.57	54.01	16.18	12.7
YAHOO	51.48		100.00	50.00	100.00	50.00	95.99	4.21	99.04	33.33	0.14
AVERAGE	63.40	8.40	68.87	14.77	71.25	14.43	67.41	15.97	58.65	10.49	

A slight negative correlation is observed between the anomaly ratio (AR) and the AUC score for all models, with LSTM-AE being the most affected (-0.32). There is a more significant negative correlation between AR and the F1 score across all models. A notable exception is the Online-EncDec-AD model, that shows a strong positive correlation (+0.68) and consistently archives the highest F1 in high AR datasets (Figure 2c). This is due to its high true positive rates (Figure 2a). However, its performance is significantly diminished in low AR series (Figure 2b), highlighting its suitability and resilience for data with higher contamination.

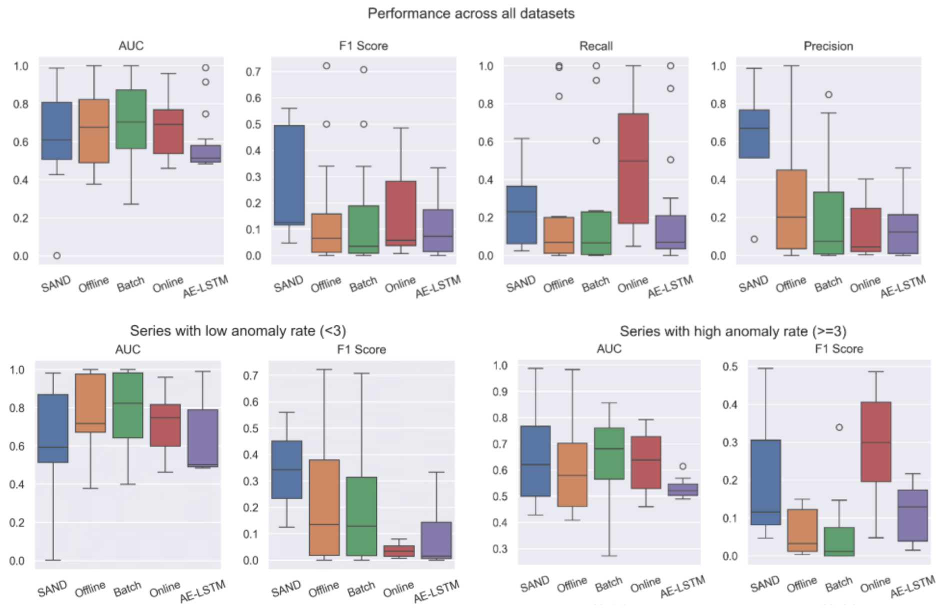


Figure 2: **(a)** Performance of models across **all** 16 single normality datasets - above. AUC and F1 scores in series with **(b) low AR** (bottom left) and **(c) high AR** (bottom right). Zero F1, Recall and Precision values for SAND were excluded from these diagrams.

3.3 Normality 2 Results

The selected combinations are grouped based on the pattern of anomaly ratios between the two series in Table 3. The **first group** consist of combinations where T^2 exhibits a decreased anomaly ratio compared to T^1, meaning T^1 has a higher anomaly ratio than T^2. In the **second group** T^1 has a lower ratio than T^2. Within each group, experiments are ordered by the absolute difference in anomaly ratios between T^1 and T^2. The first group ranges from 3.32% (C1) to 12.49% (C4) decrease, while the second from 0.17% (C5) to 30.9% (C13) increase.

In the **first group**, the streaming variants of EncDec-AD consistently outperform the Offline baseline. In contrast, the Offline model generally outperforms the streaming variants in the **second group**, with some exceptions where the Online model shows competitive F1 scores. This disparity might be attributed to the Offline model being exclusively trained solely on the less contaminated T^1 series in the second group. Notably, the pairwise correlations between Offline's AUC score and the AR of T^1 and T^2 are -0.62 and 0.01 respectively. This highlights the Offline model's diminished effectiveness when faced with higher contamination in the training set.

Interestingly, the **C13** combination in Table 3 has the highest possible contamination level with an AR of 48%. Despite autoencoders traditionally being trained exclusively on normal instances, the unsupervised EncDec-AD models outperform the non-neural SAND algorithm. The Online variant, in particular, achieves a satisfactory F1 score of 27%.

Focusing on the streaming solutions, we make the following observations:

- The **Batch** variant of EncDec-AD outperforms **SAND** in all but three series.
- While the **Batch** variant typically achieves higher AUC than the **Online** variant, especially in the first group, their relative performance is inconclusive. Specifically, the **Online** outperforms **Batch** in 46% of cases for AUC and 69% for F1 score.
- The **Online** variant surpasses **SAND** in 61% of cases in terms of AUC.
- **Online** achieves higher AUC scores than **LSTM-AE** in all but one dataset and higher F1 in 69% of cases

Table 3: Results for normality k=2 series. Each combination is denoted with the $T^1\_T^2$ format.

DATASET COMBINATION		SAND		ENCDEC-AD Offline		ENCDEC-AD Batch		ENCDEC-AD Online		LSTM-AE	
		AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1
1	SENSORSCOPE_SMD	45.64		63.63	2.32	65.76		69.61	29.62	54.11	16.10
2	ECG_MGAB	98.49	6.35	32.01		60.19	17.90	70.78	18.31	64.19	22.21
3	NAB_YAHOO	47.30		40.64	6.52	60.34	2.23	61.12	18.21	59.72	26.23
4	SENSORSCOPE_GHL	48.20		46.38	4.89	88.72	4.34	76.94	19.81	52.62	9.97
5	YAHOO_GENESIS	3.29		93.40	16.22	94.89		90.22	3.94	80.62	8.77
6	GENESIS_OPPORTUNITY	77.18		99.07	70.69	98.65	69.97	54.63	5.41	50.76	3.37
7	MGAB_OPPORTUNITY	88.34		96.27	80.20	96.14	88.33	58.23	5.61	48.54	0.54
8	ECG_SMD	60.69	2.90	73.90	14.12	78.40	9.53	64.44	20.30	55.98	17.66
9	ECG_SENSORSCOPE	90.15	1.84	74.41	2.81	67.43		71.02	37.85	54.82	16.77
10	IOPS_SMD	34.61		63.05	2.34	40.00	0.53	58.04	10.46	51.26	5.36
11	IOPS_NAB	53.30		86.83	10.96	83.75	10.25	53.47	2.41	63.26	20.33
12	MITDB_OCCUPANCY	87.25	0.80	99.87	97.14	84.77	54.20	75.36	21.54	49.14	3.67
13	OCCUPANCY_SVDB	31.63		56.59	2.65	45.31	3.37	55.45	26.92	52.67	12.68
AVERAGE		58.92	0.91	71.23	23.91	74.18	20.05	66.10	16.95	56.74	12.58

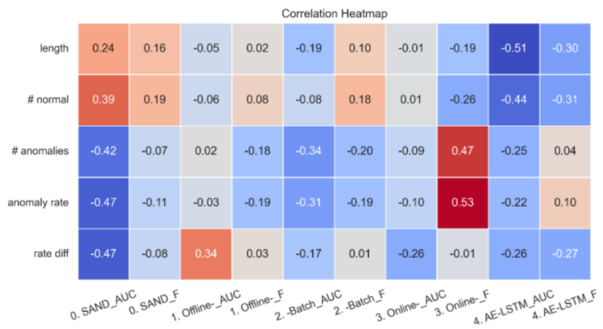


Figure 3: Correlation matrix showcasing the correlation between certain characteristics of **45** double normality datasets and the performance of each model in terms of AUC and F1 scores. "rate diff" denotes the difference between the ARs of T^2 and T^1

3.4 Normality 3 Results

Table 4 presents representative results for triple normality datasets sorted according to the absolute value of the difference between the ARs of T^1 and T^3. The subseries in some triplets are sorted in ascending order (↑) of their ARs, similar to the first group in Table 3, while others are sorted in descending order (↓), similar to the second group.

The three EncDec-AD models achieve comparable performance in terms of AUC, with the Online variant showing slightly lower scores. The Offline baseline is outperformed by the streaming variants in 67% of cases for AUC and in all but two datasets for F1. In all cases where Offline achieved the highest AUC or F1 score (C2, 3, 7, 13, 15) the model was trained on the least contaminated subseries of the triplet (↑). This behavior is confirmed by examining flipped pairs of triples, such as C1 and C2, where the Offline model consistently performs better in the ascending triplet. Meanwhile, no stable pattern is observed in the rest of the models related to the AR trends.

A general trend is that the OnlineEncDec-AD and LSTM-AE models outperform the ranking-based models in terms of F1 score, with the former achieving the highest score in all but three datasets. This superior performance is attributed to the dynamic threshold mechanism, which enables the classifier to adapt to the evolving distribution of anomaly scores. However, despite the Online model's increased recall, it also frequently yields a high number of false positives (Figure 4). This behavior might be acceptable if the AD task is evaluated in a cost-sensitive manner, with varying weights assigned to normal and abnormal classes [9]. Nonetheless, achieving a better balance between recall and precision would be preferable to optimize overall performance.

Finally, when comparing the Online variant with the other two intrinsically streaming frameworks:

- It achieved a higher AUC score than SAND in 67% of cases and higher F1 score in all series where such value was recorded for SAND.
- It outperformed LSTM-AE in all but one dataset for each metric.

Table 4: Results for normality k=3 series. Each combination is denoted with the $T^1\_T^2\_T^3$ format. ↓ denotes triplets sorted in descending order of the AR of T^i, while ↑ in ascending. Triple pairs highlighted with purple are pairs where T^1 and T^3 were flipped.

	DATASET COMBINATION	AR TREND	SAND		ENCDEC-AD Offline		ENCDEC-AD Batch		ENCDEC-AD Online		LSTM-AE	
			AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1
1	OPPORT_Genesis_YAHOO	↓	82.66		98.61	39.75	99.15	89.08	47.39	1.09	49.39	1.64
2	YAHOO_Genesis_OPPORT	↑	78.06		98.92	62.41	89.16		57.32	6.16	50.42	2.85
3	MGAB_OPPORT_ECG	↑	79.77	11.80	70.93	41.50	69.22	35.09	66.77	16.04	53.48	7.96
4	IOPS_ECG_SMD	↑	67.40	12.55	59.93	1.22	54.33	5.03	67.51	16.22	50.64	3.11
5	SMD_ECG_MGAB	↓	80.06	1.45	43.94	1.28	53.36	3.95	77.89	28.64	56.61	16.84
6	MGAB_ECG_SMD	↑	60.89	3.91	60.71	18.45	77.49	4.19	69.83	19.26	57.25	16.81
7	IOPS_SMD_NAB	↑	52.84		63.95	2.39	34.38	2.86	57.32	10.68	51.87	7.34
8	SensorScope_SMD_IOPS	↓	42.61		34.48	3.94	45.11		71.14	19.62	55.32	18.27
9	IOPS_SMD_SensorScope	↑	47.48		56.85	2.06	66.42		65.60	20.69	51.79	7.63
10	SensorScope_ECG_GHL	↓	47.67	0.06	41.16	2.83	64.15	2.64	69.40	20.09	55.46	14.82
11	SensorScope_SMD_GHL	↓	47.48		66.00	2.56	52.57		73.18	21.18	52.09	9.58
12	SensorScope_ECG_MGAB	↓	51.47	0.10	42.87	2.87	71.43	2.69	71.68	27.54	56.58	18.06
13	MGAB_ECG_SensorScope	↑	77.61	2.85	81.79	2.83	72.44		72.82	33.25	55.54	14.74
14	SVDB_Occupancy_MITDB	↓	50.91		76.95	14.20	78.01	9.36	53.13	23.71	51.24	11.39
15	MITDB_Occupancy_SVDB	↑	40.64	1.76	86.48	11.60	85.06	9.38	52.14	22.30	50.63	10.44
		AVERAGE	60.50	2.29	65.57	13.99	67.48	10.95	64.87	19.09	53.32	10.55

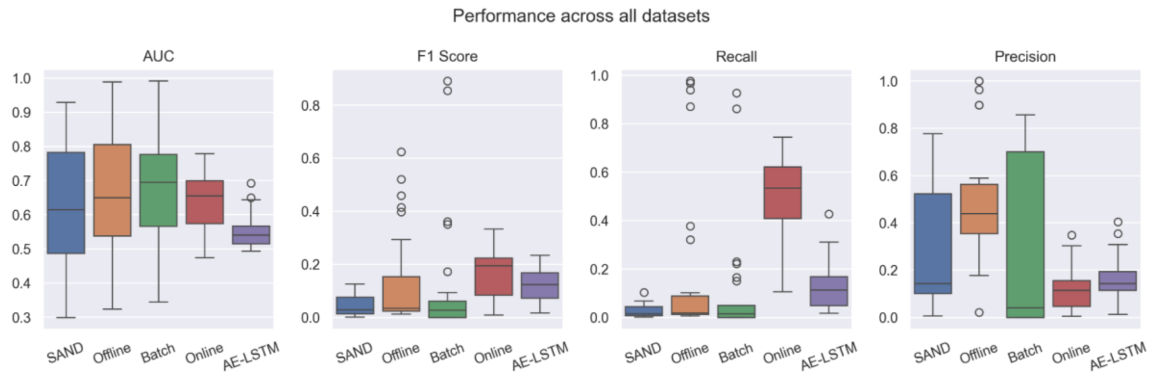


Figure 4: Performance of models across all **28** triple normality datasets generated as described in section 3.1. Zero F1, Recall and Precision values for SAND were excluded from these diagrams.

3.5 OnlineEncDec-AD Ablation Study

This section examines the significance of the key mechanisms in the OnlineEncDec-AD model, namely Concept Drift Detection (CDD), Incremental Learning (IL), and Dynamic Anomaly Threshold (DAT). The default model integrates all three functionalities. To evaluate the individual contributions of each mechanism, we conducted an ablation study by disabling one feature at a time and comparing the results. Given the high computational demands of this study, a representative subset of datasets across each level of normality (k=1, 2, and 3) was selected for experimentation.

The exclusion of **IL** has the least impact on model performance, with some datasets showing comparable or even slightly better results without it, as shown in Figure 7. For instance, datasets like Daphnet and ECG show marginal differences in performance with or without IL, indicating that the model can maintain reasonable performance levels even without continuous learning from new data. In contrast, the removal of **CDD** leads to a noticeable performance drop even in single normality series, underscoring its importance in adapting to changes in data distribution. Datasets exhibiting high variability benefit the most from CDD. Without it, the model's ability to detect anomalies diminishes significantly, as illustrated by its reduced Recall scores in Figure 5 .

Meanwhile, the removal of **DAT** shows mixed results, with its effectiveness varying depending on the dataset. In single normality series in particular, both significant improvements (e.g., Occupancy, SensorScope) and declines (e.g., OPPORTUNITY, SMD) were recorded. Overall, the inclusion of DAT negatively impacts both the detection of anomalies and the reduction of false positives (Figure 5c,d). However, this behavior is mitigated in higher normality datasets, where the contribution of DAT is largely positive (Figure 6b,c). This indicates that while threshold adaptation can sometimes hinder performance, it generally enhances performance in more complex scenarios.

Consequently, while the default model generally performs well across most datasets, it is not universally superior. For instance, in double normality series (Figure 6b), it consistently achieves the second highest F1 scores, as there are cases where removing one mechanism results in similar or better performance. However, other configurations might excel in individual datasets but fail to maintain their superiority reliably. This highlights the nuanced synergy between these mechanisms, indicating that they complement each other and help the model maintain satisfactory accuracy across diverse datasets. Nonetheless, the dataset-specific impacts suggest that customizing the inclusion of these functionalities based on the dataset's characteristics could further optimize performance.

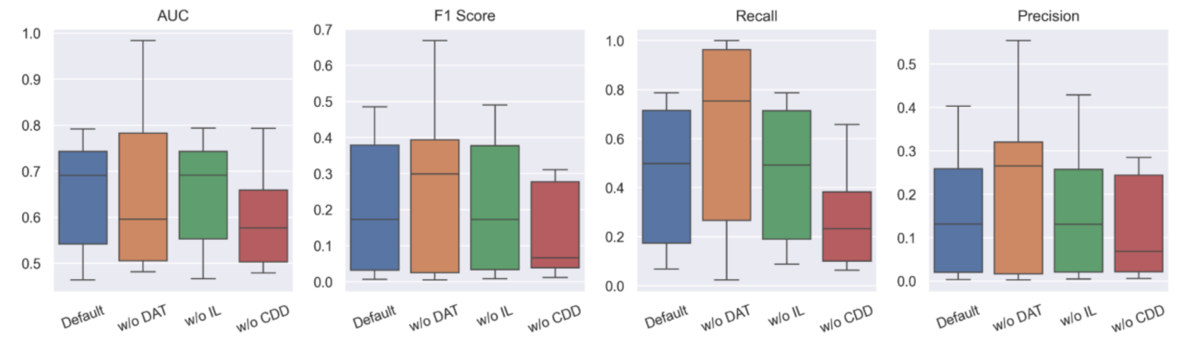
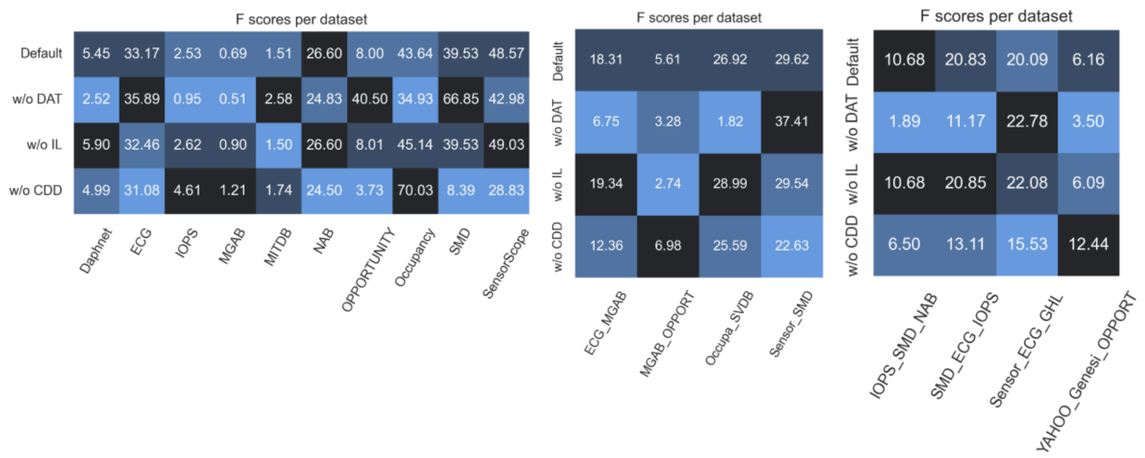


Figure 5: Ablation study results across 10 single normality datasets


F scores per dataset
F scores per dataset
F scores per dataset

	Daphnet	ECG	IOPS	MGAB	MITDB	NAB	OPPORTUNITY	Occupancy	SMD	SensorScope
Default	5.45	33.17	2.53	0.69	1.51	26.60	8.00	43.64	39.53	48.57
w/o DAT	2.52	35.89	0.95	0.51	2.58	24.83	40.50	34.93	66.85	42.98
w/o IL	5.90	32.46	2.62	0.90	1.50	26.60	8.01	45.14	39.53	49.03
w/o CDD	4.99	31.08	4.61	1.21	1.74	24.50	3.73	70.03	8.39	28.83

	ECG_MGAB	MGAB_OPPORT	Occupa_SVDB	Sensor_SMD
Default	18.31	5.61	26.92	29.62
w/o DAT	6.75	3.28	1.82	37.41
w/o IL	19.34	2.74	28.99	29.54
w/o CDD	12.36	6.98	25.59	22.63

	IOPS_SMD_NAB	SMD_ECG_IOPS	Sensor_ECG_GHL	YAHOO_Genesi_OPPORT
Default	10.68	20.83	20.09	6.16
w/o DAT	1.89	11.17	22.78	3.50
w/o IL	10.68	20.85	22.08	6.09
w/o CDD	6.50	13.11	15.53	12.44

Figure 6: F1 scores (%) in the anomaly class in single (a: left), double (b: middle) and triple (c: right) normality series

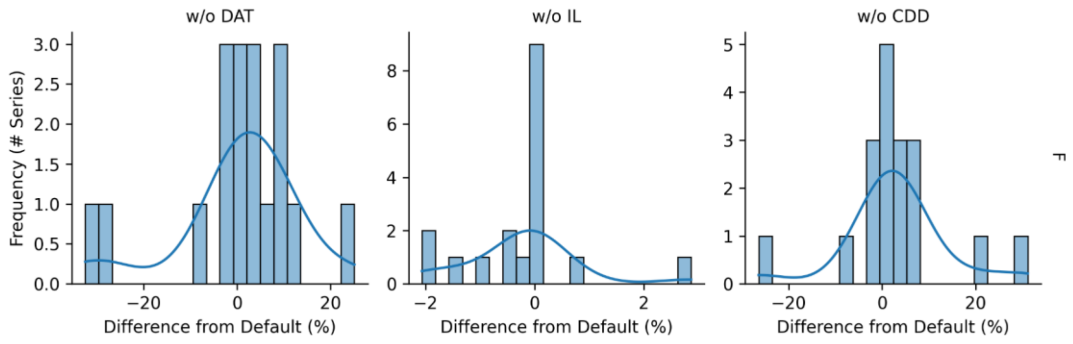


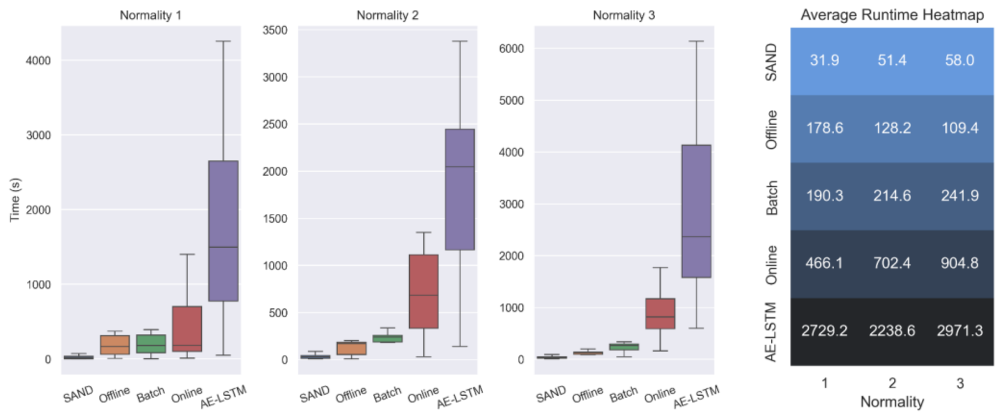
Figure 7: Distribution of the difference in F1 scores between the default OnlineEncDec-AD model and each ablation variant across all selected datasets. High positive values indicate that the default model outperforms the variant.

3.6 Runtime Performance Analysis

The performance of AD models can be significantly impacted by their runtime, particularly in a streaming setting where timely AD is crucial. This section analyzes the runtime of five different models across three normality levels. The SAND algorithm consistently demonstrates the lowest runtime across all normality levels. This makes SAND the most efficient model in terms of computational time, highlighting its suitability for real-time applications. In contrast, the Offline-EncDec-AD model's runtime is significantly higher than that of SAND, indicating a trade-off between computational efficiency and model accuracy. The EncDec-AD-Batch model is slower than both SAND and the Offline baseline as it retrains the model from scratch using a percentage of each arriving data batch.

The Online-EncDec-AD model exhibits high variability in runtime. The significant increase as normality levels rise indicates that this model's complexity grows substantially with the data's diversity and volume. Therefore, while its ability to adapt to evolving data distributions is beneficial, it comes at the cost of higher computational demands. The LSTM-AE model is significantly slower than Online, with runtimes often an order of magnitude higher. Despite its longer processing times, the LSTM-AE is frequently outperformed by the Online-EncDec-AD, which proves to be a more accurate and effective model for real-time AD.

In conclusion, the SAND algorithm stands out for its low runtime, making it ideal for applications where speed is paramount. Offline-EncDec-AD offers a balance between computational efficiency and potentially higher accuracy, especially in more stable data environments. EncDec-AD-Batch provides reliable performance across different normality levels but at a higher computational cost. Online-EncDec-AD and AE-LSTM, while possibly offering advanced detection capabilities at higher levels of normality, have high runtime requirements. The Online-EncDec-AD, however, achieves a better balance of performance and speed compared to LSTM-AE, making it a more practical choice for dynamic environments.



	1	2	3
SAND	31.9	51.4	58.0
Offline	178.6	128.2	109.4
Batch	190.3	214.6	241.9
Online	466.1	702.4	904.8
AE-LSTM	2729.2	2238.6	2971.3

Figure 8: Model execution times per normality (k) in seconds. It is worth noting that the max length of the series is 100k for single normality, while the max length of each subseries is 50k for higher levels of normality. Therefore, these diagrams aim to establish a comparison between models and not necessarily between normality levels.

REFERENCES

[1] J. Paparrizos, Y. Kang, P. Boniol, R. S. Tsay, T. Palpanas, and M. J. Franklin, “TSB-UAD,” *Proceedings of the VLDB Endowment*, vol. 15, no. 8, pp. 1697–1711, Apr. 2022, doi: 10.14778/3529337.3529354.

[2] Z. Z. Darban, G. I. Webb, S. Pan, C. C. Aggarwal, and M. Salehi, “Deep Learning for Time Series Anomaly Detection: A Survey,” Nov. 2022.

[3] R. Chalapathy and S. Chawla, “Deep Learning for Anomaly Detection: A Survey,” Jan. 2019.

[4] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, “LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection,” Jul. 2016.

[5] P. Boniol, J. Paparrizos, T. Palpanas, and M. J. Franklin, “SAND,” *Proceedings of the VLDB Endowment*, vol. 14, no. 10, pp. 1717–1729, Jun. 2021, doi: 10.14778/3467861.3467863.

[6] J. Li, K. Malialis, and M. M. Polycarpou, “Autoencoder-based Anomaly Detection in Streaming Data with Incremental Learning and Concept Drift Adaptation,” in *2023 International Joint Conference on Neural Networks (IJCNN)*, IEEE, Jun. 2023, pp. 1–8. doi: 10.1109/IJCNN54540.2023.10191328.

[7] N. Belacel, R. Richard, and Z. M. Xu, “An LSTM Encoder-Decoder Approach for Unsupervised Online Anomaly Detection in Machine Learning Packages for Streaming Data,” in *2022 IEEE International Conference on Big Data (Big Data)*, IEEE, Dec. 2022, pp. 3348–3357. doi: 10.1109/BigData55660.2022.10020872.

[8] A. Bifet and R. Gavaldà, “Learning from Time-Changing Data with Adaptive Windowing,” in *Proceedings of the 2007 SIAM International Conference on Data Mining*, Philadelphia, PA: Society for Industrial and Applied Mathematics, Apr. 2007, pp. 443–448. doi: 10.1137/1.9781611972771.42.

[9] K. G. Mehrotra, C. K. Mohan, and H. Huang, *Anomaly Detection Principles and Algorithms*. Cham: Springer International Publishing, 2017. doi: 10.1007/978-3-319-67526-8.