

Article

ReLATE+: Unified Framework for Adversarial Attack Detection, Classification, and Resilient Model Selection in Time-Series Classification

Cagla Ipek Kocal ¹, Onat Gungor ^{2,*}, Tajana Rosing ² and Baris Aksanli ¹

¹ Department of Electrical and Computer Engineering, San Diego State University, San Diego, CA 92182, USA; ckocal0169@sdsu.edu (C.I.K.); baksanli@sdsu.edu (B.A.)

² Department of Computer Science and Engineering, University of California, San Diego, CA 92093, USA; tajana@ucsd.edu

* Correspondence: ogungor@ucsd.edu

Abstract

Minimizing computational overhead in healthcare time-series classification remains a critical challenge, particularly for deep learning models operating on high-dimensional sequential data under resource and latency constraints. This challenge is further exacerbated by adversarial attacks, which introduce evolving threats and necessitate robust yet efficient mechanisms for maintaining reliable model performance—a requirement of paramount importance in healthcare, where wearable and smart devices have brought continuous monitoring outside clinical settings and model failures can directly compromise patient safety. In this paper, we propose ReLATE+, a unified framework for adversarially robust and computationally efficient time-series classification. ReLATE+ integrates three key capabilities: (i) detection and classification of adversarial inputs, (ii) dataset-level similarity analysis, and (iii) adaptive model selection. Upon receiving new data, the framework first determines whether the input is adversarial and identifies the attack type. It then leverages this information to retrieve a similar dataset from a repository and identify the corresponding high-performing models and trains only a small set of selected candidates instead of exhaustively retraining all models. This approach ensures strong performance while reducing the need for retraining, and it generalizes well across different domains with varying data distributions and feature spaces. Experiments show that ReLATE+ reduces computational overhead by an average of 77.68%, enhancing adversarial resilience and streamlining robust model selection, all without sacrificing performance, within 2.02% of Oracle.



Academic Editors: Irfan Awan, Amna Qureshi and Muhammad Shahwaiz Afaqui

Received: 4 May 2026

Revised: 20 July 2026

Accepted: 25 July 2026

Published: 27 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

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

Keywords: time-series classification; adversarial robustness; attack detection and classification; efficient inference; adaptive model selection; edge and distributed intelligence

1. Introduction

Various tasks rely on time-series data, i.e., sequences of observations collected over intervals, including anomaly detection [1], clustering [2], and classification [3]. Among these tasks, time-series classification with machine learning (ML) spans a wide range of domains, including network intrusion detection [4], financial time-series classification [5], and speech emotion recognition [6]. While impactful across many fields, healthcare stands out as a domain of particular urgency, with applications such as epileptic activity classification using EEG signals [7], wearable activity recognition [8], patient monitoring [9],

and clinical decision support [10] all requiring robust, resilient, secure, and accurate ML-based solutions that can adapt in real time. These requirements are especially critical when deployment targets resource-constrained edge devices, where computational overhead must be minimized without sacrificing robustness against evolving adversarial conditions that can directly compromise patient safety.

Time-series ML applications face significant challenges due to the dynamic nature of streaming data, which is often limited or incomplete in real-time settings, making it impractical to wait for sufficient data accumulation before retraining models [11]. Furthermore, retraining models on newly observed data is computationally expensive and time-consuming, posing additional challenges for deployment in resource- and latency-constrained environments encountered in networked systems [12]. In this context, deep learning (DL) models are often favored for multivariate time-series classification tasks due to their ability to automatically extract relevant features. However, these DL models could show significant variability in classification performance, as shown in Figure 1. These results highlight the substantial impact of model choice on classification outcomes, underscoring the critical need for careful and informed model selection in the context of DL-based time-series analysis. This motivates the need for efficient DL model selection methods that can adapt to new incoming data without requiring extensive retraining.



Figure 1. Prediction performance of deep learning models on multivariate time-series classification.

In addition to the challenges posed by streaming data and retraining overhead, deep learning (DL) models are vulnerable to adversarial attacks, as they can be manipulated by small, imperceptible changes in the input data, especially when the data is limited or incomplete. These attacks introduce perturbations that obscure essential patterns, potentially leading to misclassifications in high-stakes applications where accuracy is critical. This is particularly concerning in real-time applications where timely and accurate predictions are critical, such as healthcare. For example, small perturbations in medical sensor data could lead to incorrect diagnoses, potentially endangering lives [13] or manipulated wearable sensor readings could lead to misdiagnosis in remote patient monitoring systems, where clinicians rely entirely on automated model outputs for decision making [14]. Unlike domains where errors carry financial or operational costs, healthcare misclassifications can be irreversible, making adversarial robustness not merely desirable but clinically essential. Mitigating these risks requires methods that not only adapt efficiently to new incoming data but also demonstrate resilience against adversarial attacks.

Building on the need for resilience in streaming environments, the ability to effectively detect adversarial manipulations and perform robust, attack-aware classification forms a foundational component of secure time-series systems [15]. Rather than relying solely on costly model retraining, next-generation frameworks must be capable of flagging suspicious perturbations as data arrive, determining the nature of any tampering, and then seamlessly invoking classifiers that can both label and be resilient against adversarial inputs. Unifying lightweight attack detection with adaptive model selection maintains real-time performance without heavy computational demands. This allows it to stay accurate and reliable even against adaptive adversarial strategies in critical time-series applications.

In light of these challenges, this work focuses specifically on healthcare time-series classification, motivated by three complementary factors: (i) domain criticality—healthcare is among the highest-stakes environments for ML deployment, where adversarial vulnerabilities can translate directly into patient harm, making robust and reliable classification a non-negotiable requirement—this urgency is amplified by the ubiquity of consumer health devices, which move monitoring outside clinical supervision and broaden the attack surface; (ii) data availability—open-source healthcare and human activity recognition benchmarks provide standardized, reproducible evaluation environments that enable rigorous and fair comparison across methods; and (iii) computational constraints—healthcare monitoring systems are increasingly deployed on resource-limited edge devices such as wearables and bedside monitors, where reducing inference overhead is not a convenience but a prerequisite for real-world deployment.

We propose ReLATE+ to tackle the computational and retraining challenges in DL-based time-series classification with streaming data, particularly in the presence of adversarial attacks. ReLATE+ begins by flagging adversarially attacked inputs, introducing minimal computational overhead compared to standard inference and avoiding the costly retraining cycles required by conventional model-centric defenses. It then infers the attack type and leverages this information to identify a similar dataset from a repository so that only the model architectures that previously demonstrated robustness under comparable data and attack conditions are trained on the incoming data. This three-step pipeline substantially reduces the need for exhaustive testing and retraining across all models and datasets. ReLATE+ achieves substantial computational savings, averaging 77.68% reduction in overhead, with strong performance, within 2.02% of Oracle, providing a fast and efficient solution for time-series classification under both adversarial and non-adversarial conditions.

2. Background and Related Work

2.1. Adversarial Attacks

Adversarial attacks pose a significant threat to the reliability of machine learning models by introducing subtle, often imperceptible perturbations to input data that lead to incorrect predictions [16]. These attacks are typically categorized into white-box and black-box settings. White-box attacks assume full access to the model's architecture, parameters, and gradients, enabling attackers to craft highly effective perturbations by directly exploiting internal model information. In contrast, black-box attacks operate with no knowledge of the model's internal workings and rely solely on observable outputs to guide the adversarial manipulation [17].

Costa et al. [18] offer a taxonomy of adversarial attacks. Among the white-box methods, Fast Gradient Sign Method (FGSM) generates adversarial examples in a single step by applying perturbations in the direction of the gradient's sign to maximize the model's loss. Its iterative variant, the Basic Iterative Method (BIM), applies smaller perturbations across multiple steps, resulting in stronger adversarial examples. Momentum Iterative Method

(MIM) further improves BIM by adding momentum in gradient updates to avoid poor local minima, enhancing attack stability and success. DeepFool operates by approximating the classifier's decision boundaries as linear hyperplanes and iteratively adjusting inputs to find the minimal perturbation needed for misclassification. Auto-Projected Gradient Descent (AutoPGD) dynamically adapts step sizes in optimization, making it a robust and efficient iterative white-box attack. ElasticNet, a white-box method, integrates both L1 and L2 regularization, enabling the generation of sparse and interpretable perturbations, valuable in domains requiring minimally invasive changes. In the black-box setting, Boundary Attack initiates from a misclassified input and gradually reduces the perturbation while maintaining misclassification. This is especially effective when model gradients are unavailable, suitable for real-world systems with restricted access.

2.2. Multivariate Time-Series Classification (MTSC)

MTSC assigns predefined labels to time-series data by analyzing the temporal patterns across multiple variables simultaneously [19]. Traditional ML methods often struggle with raw time-series data due to its temporal structure and high dimensionality, needing specialized approaches [20]. Numerous ML algorithms are designed to enhance the scalability and predictive capabilities of models for time-series classification. Zheng et al. developed a framework with multi-channel deep convolutional neural networks in combination with Multilayer Perceptrons [21]. Grabocka et al. introduced a shapelet-based method with supervised selection and online clustering [22]. Ruiz et al. proposed a method combining DL, shapelets, bag-of-words approaches, and independent ensembles [23]. Baldán et al. employed feature-based methods with traditional classifiers [24]. Fawaz et al. reviewed various DL models for time-series classification, including CNN-based architectures such as ResNet and FCN, and evaluated them also on multivariate benchmarks [3]. Cong et al. uses labeled Petri nets to find fault patterns in discrete time systems [25]. Despite significant advancements in MTSC, the computational overhead remains a critical limitation. Most existing approaches rely on exhaustive training procedures that require significant computational resources.

2.3. Similarity-Based Selection Approaches

Similarity analysis has a crucial role in ML and time-series analysis, offering a foundation for dataset comparison, feature selection, and model evaluation. Marks examined three measures of similarity for comparing two sets of time-series vectors, including the Kullback–Leibler divergence, the State Similarity Measure, and the Generalized Hartley Metric [26]. Bounliphone et al. introduced a statistical test of relative similarity to address challenges in model selection for probabilistic generative frameworks [27]. Assegie et al. proposed a feature selection method using dataset similarity to improve the classification performance [28]. Cazelles et al. introduced Wasserstein-Fourier distance to measure the dissimilarity between stationary time series [29]. Xu et al. further explored Dynamic Time Warping as a robust method for time-series curve similarity, and it is highly effective in handling variations in temporal alignment [30]. Existing methods are tailored for static datasets and do not account for the dynamic nature of time-series data, lacking the adaptability required for evolving data and resilience against adversarial attacks.

2.4. Adversarial Attacks in MTSC

The streaming, high-dimensional, and time-sensitive nature of MTSC makes it especially susceptible to even minor adversarial perturbations, motivating a dedicated review of attacks in this domain. Several studies analyze the vulnerabilities of MTSC models to adversarial attacks and propose solutions to enhance resilience. Harford et al. adapt Adversarial Transformation Network on a distilled model and show that 1-NN Dynamic

Time Warping and Fully Convolutional Networks are highly vulnerable [31]. Galib et al. analyze time-series regression and classification performance under adversarial attacks, finding that Recurrent Neural Network models were highly susceptible [32]. Siddiqui et al. proposed a regularization-based defense [33]. Gungor et al. developed a resilient stacking ensemble learning-based framework against various adversarial attacks [34]. However, these approaches overlook significant retraining or exhaustive experimentation whenever new attack scenarios or datasets emerge.

2.5. Adversarial Attack Detection

To address the retraining overhead of model-centric defenses, many works shifted toward attack detection, identifying manipulated inputs so that classifiers can apply specialized handling or fallbacks to improve robustness. [35]. Feature Squeezing reduces input complexity, such as bit-depth or resolution, and flags samples based on inconsistent predictions across transformations [36]. Mahalanobis-distance-based detection models the distribution of deep features per class and classifies inputs as adversarial if they fall outside high-density regions [37]. In time-series domains, RADAR applies a bi-directional LSTM autoencoder to detect perturbations through reconstruction error and uncertainty estimation [38]. TRAIT takes a spectral approach by analyzing the loss landscape during training to detect adversarial patterns with high generalizability [39]. ReLATE+ builds on these approaches by integrating Fourier and Wavelet-based anomaly detection with similarity-driven model selection, enabling resilient performance with substantially reduced retraining.

ReLATE+ addresses both adversarial robustness and retraining overhead by combining a targeted attack detection pipeline with similarity-driven model selection. It begins by screening incoming time-series data for adversarial perturbations, and when detected, classifies them into specific adversarial attack types. Using a pre-evaluated model repository, ReLATE+ computes dataset level similarity and dynamically selects the most effective classifier, reducing retraining to a small set of DL models rather than the full model pool. This not only adapts to the dynamic nature of new time-series data but also enhances robustness against adversarial attacks by prioritizing resilient model selection.

3. Proposed Framework: ReLATE+

ReLATE+ provides a robust approach for model selection in time-series classification, balancing resilience with computational efficiency. Its design is driven by the need to (1) reduce model retraining through strategic reuse of existing models, (2) detect and adapt to adversarial threats in a lightweight manner, and (3) match incoming data to historically similar conditions to inform model selection. The architecture of ReLATE+, comprising four key modules, is illustrated in Figure 2. Module 0 constructs a Performance Benchmark Database (PBD) that stores pre-evaluated metrics for various DL models across multiple datasets and attack scenarios. Once new data are received, Module 1 classifies them as clean, partially attacked, or fully attacked. If an attack is detected, Module 2 further identifies the attack group. Module 3 matches the incoming data to the most similar dataset in the repository using a similarity metric, and selects the top performing models accordingly, eliminating exhaustive model evaluations.

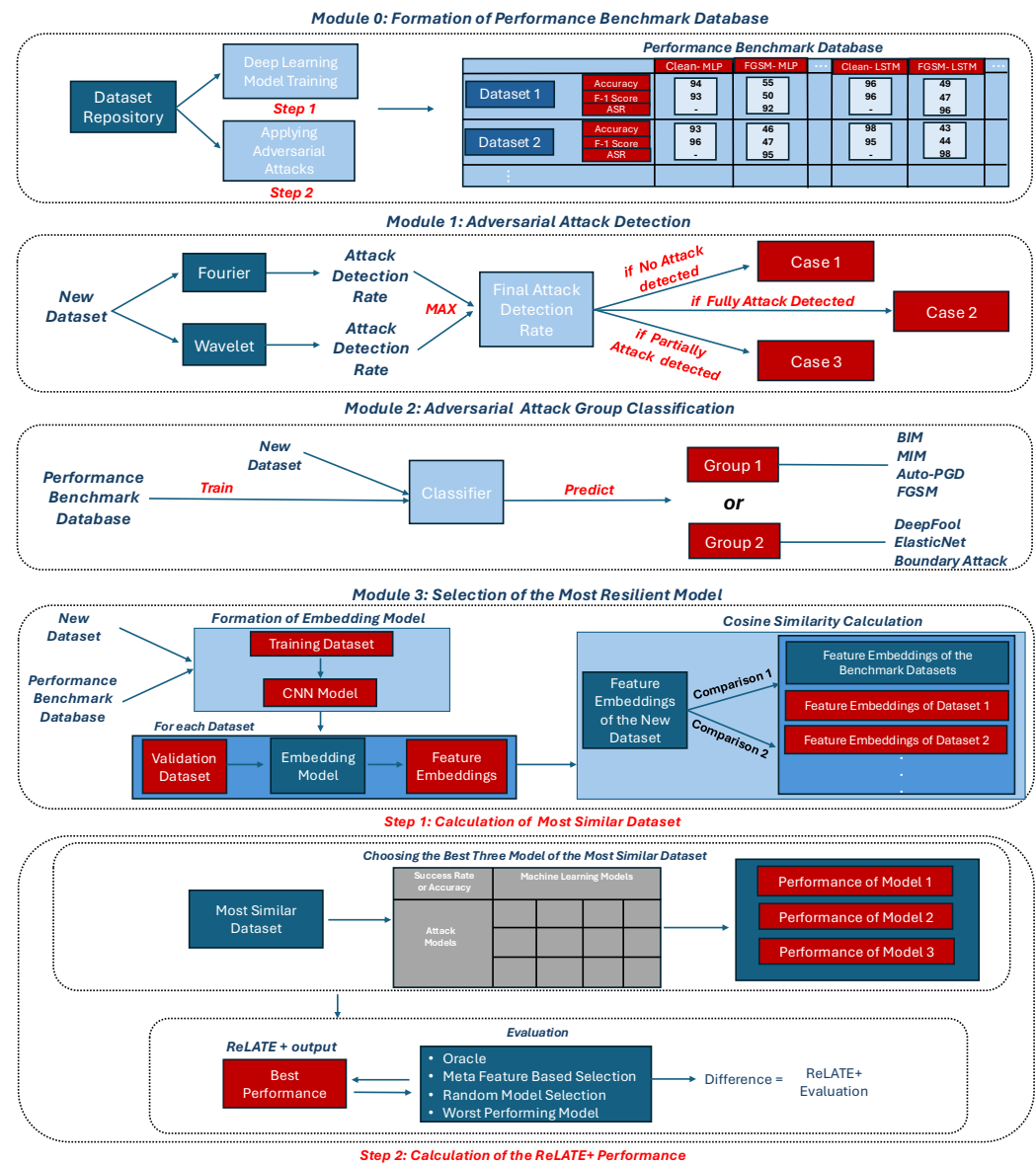


Figure 2. ReLATE+ framework.

3.1. Module 0: Formation of Performance Benchmark Database

This module constructs a PBD using previously analyzed datasets. It is conducted offline prior to receiving any incoming data and consists of two main steps: deep learning model training and adversarial attack generation.

3.1.1. Step 1: Deep Learning Model Training

This step trains our DL models on various datasets. We begin with hyperparameter tuning to identify the optimal settings for each model and dataset. We then train the models and record their performance (accuracy and F1-score) in the PBD. We use the training dataset for model training, the validation dataset for hyperparameter tuning, and reserve the test dataset to evaluate the final performance. This module processes the input datasets and generates the clean data (no attack) performance. The DL models span a diverse range, each addressing specific challenges of time-series data, including capturing long-range temporal dependencies, identifying cyclical patterns within sequential data, and managing high dimensionality in multivariate settings. We select 14 state-of-the-art DL models: MLP [40], FCN [40], ResNet [40], LSTM [41], GRU [41], LSTM-FCN [41],

GRU-FCN [41], MSWDN [42], TCN [43], MLSTM-FCN [44], InceptionTime [45], Residual CNN [46], OmniScaleCNN [47], and Explainable Convolutional Model [48].

3.1.2. Step 2: Applying Adversarial Attacks

Threat Model: We consider an adversary whose goal is to induce untargeted misclassification, i.e., to cause the model to produce any incorrect label rather than steer predictions toward a specific target class. Regarding the adversary's knowledge, we evaluate both white-box and black-box settings: the white-box attacks (FGSM, BIM, MIM, AutoPGD, DeepFool, and ElasticNet) represent a worst-case, maximally informed adversary, while the black-box Boundary Attack reflects a more constrained and practical one. In terms of capability, the adversary is restricted to norm-bounded perturbations δ satisfying $\|\delta\| \leq \epsilon$, ensuring that the manipulation remains subtle and difficult to detect; unless otherwise stated, we set $\epsilon = 0.1$, and we further examine $\epsilon \in \{0.001, 0.2\}$ in our ablation study (Section 5.4). Finally, regarding the attack surface, we assume the adversary can inject perturbations at any point after the data has been digitized but before it reaches inference. In a representative deployment—for example, the motion and activity based datasets used in this work (e.g., wearable gesture and movement recordings) would, in a real healthcare deployment, originate from body-worn sensors whose readings traverse multiple hardware stages. An adversary could inject perturbations at any of these: the sensing device itself, the wireless network during transmission, an edge aggregation gateway, local on-device storage, or the communication channel with a cloud-based inference service.

Applying Adversarial Attacks: In this step, each dataset-model pairing undergoes evaluation through nine state-of-the-art adversarial attacks, both white-box and black-box. White-box attacks have full access to the model's architecture and parameters to generate adversarial examples, while black-box attacks rely solely on the model's outputs, without knowledge of its internal structure. This module generates adversarial attack versions of each dataset, testing the robustness of DL models against various attack types. The results (accuracy, F1-score, and attack success rate) are recorded in the PBD. Adversarial attacks are applied to the test portion of the data using models trained in Step 1 with optimized hyper-parameters. Each attack is chosen for its unique approach to disrupting data and exposing model weaknesses, from simple gradient-based methods to complex iterative strategies. We select seven white-box and black-box attacks [18]: Fast Gradient Sign Method (FGSM), DeepFool, Basic Iterative Method (BIM), Momentum Iterative Method (MIM), ElasticNet, Auto Projected Gradient Descent (AutoPGD) and Boundary Attack.

3.2. Module 1: Adversarial Attack Detection

In real-world scenarios, adversarial inputs often impact only portions of the data, rendering simple binary classification (attack vs. no attack) inadequate for capturing varying degrees of compromise. To better reflect this complexity, we classify incoming data as clean (Case 1), fully attacked (Case 2), or partially attacked (Case 3). For this classification task, we apply both Wavelet Transform and Fourier Transform, selected for their complementary strengths in detecting different types of adversarial perturbations. The structure of incoming time-series data, whether perturbations are localized or spread throughout, is typically unknown. The Wavelet Transform excels at identifying short-term, high-frequency anomalies common in gradient-based attacks by analyzing both time and frequency domains [49]. In contrast, the Fourier Transform is suited for capturing global frequency shifts or consistent periodic distortions introduced by systematic or optimization-based attacks [50]. Together, these transforms enable robust detection across a broad range of perturbation patterns. To support classification into the three cases, we compute the

Attack Detection Rate, quantifying the proportion of adversarial perturbation present in the input.

After obtaining detection scores from both the Wavelet and Fourier Transforms, the higher of the two is selected to represent the final attack detection ratio. This approach ensures that even localized or subtle adversarial signals, possibly detected by only one method, are not overlooked. Given the unpredictable nature of adversarial perturbations, favoring the stronger signal helps safeguard against misclassifying compromised data as clean, which could otherwise lead to inappropriate model selection or reduced robustness. In cases where the two methods disagree significantly, selecting the higher value prioritizes caution, acknowledging that an undetected attack can be more harmful than responding to a false alarm. This asymmetry is especially pronounced in healthcare applications: an undetected perturbation in a patient-monitoring signal could mask an abnormal pattern and lead to a missed alert, whereas a false alarm merely triggers a redundant check. This deliberately trades a small increase in false positives for a substantially lower risk of missing attacks—a balance we consider worthwhile, since an undetected adversarial input is more costly than an occasional false alarm.

The selected attack detection ratio is then used to classify the data into one of three categories. This three-way classification is necessary to capture the full spectrum of real-world scenarios: some data may be entirely clean (Case 1), others consistently attacked (Case 2), while many fall in between where only a portion is adversarially perturbed (Case 3). Let T be a predefined threshold: if the ratio is below T , the data are labeled clean; if it exceeds $1 - T$, they are labeled fully attacked; otherwise, they are considered partially attacked.

3.3. Module 2: Adversarial Attack Group Classification

When an adversarial attack is detected in Module 1, the next step is to determine the nature of the attack. Accurately identifying the type of perturbation enables more targeted and effective model selection in later stages, as different attacks can degrade model performance in distinct ways. Rather than pinpointing the exact attack, this module categorizes the input into one of two broader adversarial attack groups based on shared structural characteristics. Group 1: Iteration-Based Attacks consists of gradient-based attacks (including FGSM, BIM, MIM, and AutoPGD) which typically generate small, directionally consistent perturbations by directly leveraging model gradients. Group 2: Optimization- and Decision-Based Attacks comprises optimization-based and decision-based attacks such as DeepFool, Boundary Attack, and ElasticNet, which rely on iterative optimization, decision boundary exploration, or sparse modifications. Categorizing at this group level is sufficient because attacks within a group share structural characteristics that induce similar perturbations, and therefore similar model vulnerabilities; a model resilient to one attack in a group tends to be resilient to the others, which makes identifying the exact attack both unnecessary and more error-prone. Since these groupings reflect each attack's intrinsic mechanism rather than dataset-specific properties, the approach generalizes across datasets within a domain, which tend to share similar temporal characteristics and respond consistently to the same attack groups. Accordingly, a classifier trained on the PBD is used to distinguish between these groups, enabling the system to make context-aware decisions based on the likely nature of the attack.

3.4. Module 3: Most Resilient Model Selection

3.4.1. Step 1: Most Similar Dataset Calculation

When a new dataset arrives, mimicking real-time conditions, ReLATE+ initiates similarity comparison for model selection. The process begins by training a lightweight

DNN-based similarity function, built on a simple CNN architecture, using the training portion of each dataset in the PBD. Intuitively, this CNN compresses each dataset's temporal and statistical structure—length, dimensionality, class count, and dynamics—into an embedding that serves as its structural signature. Since structurally similar datasets respond similarly to models—one accurate and resilient on one tends to perform alike on another—a nearby embedding lets the same high-performing model be reused rather than learned anew. This is especially useful within a domain such as healthcare, where new datasets frequently arise—from different sensor placements, subjects, or recording conditions—yet share underlying temporal structure, allowing a model proven on one to transfer effectively to another. Concretely, once trained, we use these CNNs to extract feature embeddings from the validation portions of their respective datasets. For the incoming dataset, we train a separate similarity function using the training portion of its data and extract embeddings from the validation portion. We then normalize these embeddings to ensure efficient quantification of average similarity between datasets based on their feature distributions. We compare the embeddings extracted from the incoming dataset to those in the PBD with an Embedding Similarity Metric, e.g., Cosine similarity.

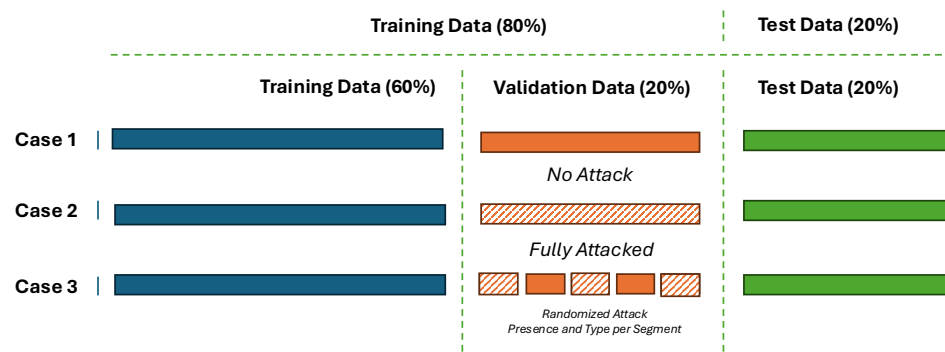
3.4.2. Step 2: ReLATE+ Performance Assessment

Here, we find the dataset with the highest similarity score and select its top three performing DL models (ranked by test performance recorded in Module 0: Step 1 and 2). This approach minimizes computational overhead by eliminating the need to retrain or test all models on the incoming dataset, training only the selected top three rather than avoiding training altogether. Next, we assess the performance of the selected models on the new dataset by training them on its training portion and evaluating their accuracy and resilience on the test portion. For clean data, we measure performance with accuracy. For the adversarial data, we assess resilience with attack success rate (ASR). This selection is directly informed by the similarity analysis, ensuring that the chosen model is well suited to handle the unique characteristics and potential adversarial challenges of the new data. The best model is then deployed on the new dataset for real-time use. We compare the performance of the best model with oracle, random model selection, meta-feature-based selection and the worst-performing model to evaluate ReLATE+ output.

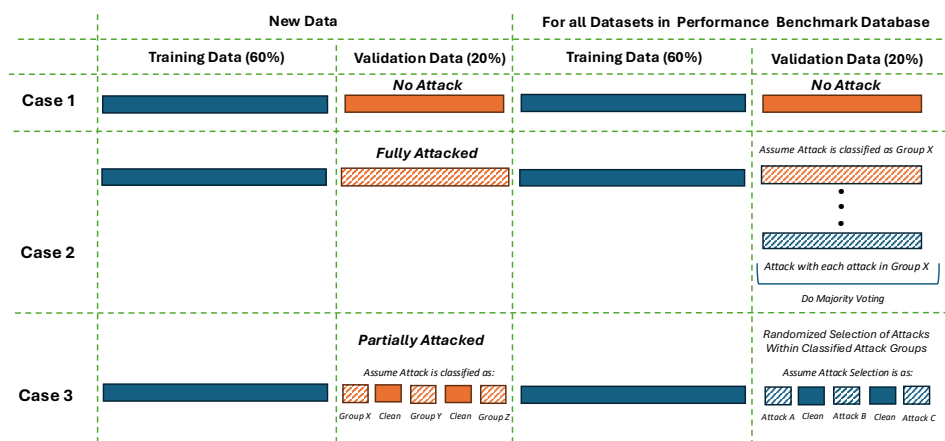
3.5. Using ReLATE+ on a New Dataset and Design Choices

3.5.1. Incoming Data Setup

We conduct a series of experiments where each of the datasets in our PBD is treated as a new, unseen arrival in rotation. In each round, one dataset is designated as the “new arrival”, while the remaining datasets are treated as the pre-existing “drive” datasets. For each drive dataset, we assess key performance metrics, including accuracy, F1-score, and ASR, across all models, encompassing both the clean and adversarially attacked variants. Figure 3 part a illustrates the training–validation splits for each case. In all cases, the training dataset is split into an 80% training set and a 20% validation set to facilitate model training and validation for comparisons. This split is determined after several trials to find the most effective training-validation ratio. The custom CNN is trained using the training portion of the dataset, while the validation portion is utilized for similarity measurement during case implementation. Overall, we design three cases to assess model selection under different adversarial conditions:



(a) Incoming data setup



(b) Case evaluation explanation

Figure 3. Overall case explanations.

Case 1: No Adversarial Attack—When the incoming dataset is classified as clean (Figure 3a), it is compared with the clean versions of each dataset in the drive (Figure 3b). We find the most similar dataset in the drive and select its three best-performing models. We evaluate the selected models on the incoming dataset to identify the best among them.

Case 2: Full Attacks—When the incoming dataset is identified as being subject to a single type of adversarial attack (Figure 3a), a dataset similarity analysis is performed using a majority voting approach to find the most similar dataset. If the dataset is classified as fully attacked by Module 1 (Case 2), its corresponding attack group is identified by Module 2. With this attack group information, each dataset in the PBD is attacked using all individual attacks within the predicted group. For each attack, a similarity analysis is conducted between the resulting dataset and the incoming dataset, and the results are recorded. After all comparisons are completed, majority voting is applied to determine the most similar dataset overall (Figure 3b). This strategy addresses the uncertainty regarding the exact attack affecting the incoming dataset by evaluating all plausible candidates within the predicted group. Once the most similar attacked dataset is identified, its top three models—ranked by attack success rate (ASR), where a lower ASR indicates higher model resilience—are selected and evaluated on the incoming dataset.

Case 3: Partial Attacks—We design a randomized attack scenario to simulate diverse and unpredictable adversarial conditions. We divide each dataset's validation portion into five equal segments. For each segment, a binary decision is made at random to determine whether the segment will be subjected to an adversarial attack. If an attack is

to be applied, the attack type is randomly selected from the predefined set of adversarial methods described previously (Figure 3a).

Once the attack pattern across the five segments is established and treated as the incoming dataset, Module 1 classifies it as partially attacked (Case 3). Then, each segment is isolated and re-evaluated individually by Module 1 to distinguish between clean and attacked portions. Segments identified as attacked are then passed to Module 2, where their attack group is classified according to the framework. The sequence of clean and attacked segments—along with the corresponding attack groups for the attacked segments—is recorded to mirror the structure of the incoming dataset across the datasets in the PBD. The validation portion of each benchmark dataset is similarly divided into five segments. Segments identified as clean remain unchanged, while those classified as attacked are perturbed using a randomly selected attack from the corresponding predicted attack group. This approach preserves the structural integrity of the datasets while introducing controlled attack variation aligned with the incoming pattern (Figure 3b). After generating these patterned datasets, the most similar dataset is identified based on the established segment-level structure, and the subsequent steps follow the procedure outlined in the previous cases.

The complete workflow in this section is concisely captured in Algorithm 1, where each step is expressed in high-level pseudo-code to highlight the sequence of operations.

3.5.2. Design Choices

All experiments in this paper were conducted with $\epsilon = 0.1$ for attack methods that allow adjustable perturbation strength, as this value offers a balanced trade-off between attack effectiveness and subtlety, as validated by our ablation study (Section 5.4).

To implement the adversarial attack detection strategy in Module 1, the following design choices are introduced to implement the detection strategy. An attack detection method is trained on transformed clean training data by modeling normalized deviation scores, and it flags anomalies when these scores exceed a threshold based on a high-percentile value. To determine the threshold for classifying input data as Clean (Case 1), Partially Attacked (Case 3), Fully Attacked (Case 2), a series of experiments is conducted on the validation sets within the PBD. The Fourier and Wavelet-based detection methods are trained using the corresponding training partitions of each dataset. Based on the threshold sweep on the validation sets (Table 1), the combined false-positive/false-negative error is minimized and stays stable across $T \in \{13, 14, 15\}\%$, where both rates are at their optimal balance. Below this range the false-positive rate rises, and above it the false-negative rate increases. We adopt the lower end, $T = 13\%$, as a smaller threshold since it is more conservative against missed attacks; the upper bound then follows by symmetry as $1 - T = 87\%$. Under this rule, if the detection rate is below 13%, the data is considered clean (Case 1); if it falls between 13% and 87%, it is classified as partially attacked (Case 3); and if it exceeds 87%, the dataset is labeled as fully attacked (Case 2). For Case 3, since the incoming data arrives in five equally sized chunks, the possible degrees of partial attack correspond to chunk-level perturbations, i.e., 20%, 40%, 60%, and 80%. The reason behind five segments is to balance granularity against cost: too few cannot localize partial perturbations, while each additional segment is processed independently by Modules 1 and 2, increasing cost. More generally, this is a tunable parameter k : finer localization can increase k at added cost, and latency-critical settings can decrease it, without changing the framework. Given this segmentation, if the overall detection rate falls within the 13–87% range, indicating a partial attack, the system selects the closest predefined intensity level (within a $\pm 10\%$ tolerance) to estimate the proportion of attacked chunks. After this, as each

segment is independently processed through Module 1 to determine whether it is attacked or clean, a threshold of 50% is applied to each, reflecting the two possible outcomes.

Algorithm 1 Similarity-driven dataset selection.

Require: Input D_{in} , Performance Benchmark Database $\mathcal{B}$

Require: Modules M1 (detector), M2 (attack-group classifier), M3-1 (similarity), M3-2 (performance)

```

1:  $c \leftarrow M1(D_{in})$ 
2:  $val_{in} \leftarrow ValidationSplit(D_{in})$ 
3:  $test_{in} \leftarrow TestSplit(D_{in})$ 
4: if  $c = \text{CASE1}$  then ▷ Dataset fully clean
5:   for all  $\mathcal{D} \in \mathcal{B}$  do
6:      $val_{\mathcal{D}} \leftarrow ValidationSplit(\mathcal{D})$ 
7:   end for
8:    $sim \leftarrow M3-1(val_{in}, val_{\mathcal{D}})$ 
9:    $perf \leftarrow M3-2(sim, test_{in})$ 
10:  return  $(sim, perf)$ 
11: end if
12: if  $c = \text{CASE2}$  then ▷ single unknown attack
13:   $\mathcal{G} \leftarrow M2(D_{in})$ 
14:  for all  $a \in \mathcal{G}$  do
15:    for all  $\mathcal{D} \in \mathcal{B}$  do
16:       $val_{\mathcal{D}} \leftarrow ApplyAttack(a, ValidationSplit(\mathcal{D}))$ 
17:    end for
18:     $sim[a] \leftarrow M3-1(val_{in}, val_{\mathcal{D}})$ 
19:  end for
20:   $sim_{best} \leftarrow MajorityVoting(sim)$ 
21:   $perf \leftarrow M3-2(sim_{best}, test_{in})$ 
22:  return  $(sim, perf)$ 
23: end if
24: if  $c = \text{CASE3}$  then ▷ mixed clean/attacked
25:    $k \leftarrow 5$  ▷ number of segments
26:   for  $i \leftarrow 1$  to  $k$  do ▷ detect attacked segments
27:      $s_i \leftarrow i\text{-th segment of } val_{in}$ 
28:      $status[i] \leftarrow M1(s_i)$ 
29:     if  $status[i] = \text{ATTACKED}$  then
30:        $\mathcal{G}[i] \leftarrow M2(s_i)$ 
31:     else
32:        $\mathcal{G}[i] \leftarrow \emptyset$ 
33:     end if
34:   end for
35:   for all  $\mathcal{D} \in \mathcal{B}$  do ▷ create attacked replicas
36:      $val_{\mathcal{D}} \leftarrow ValidationSplit(\mathcal{D})$ 
37:     Partition  $val_{\mathcal{D}}$  into  $t_1, \dots, t_k$ 
38:     for  $i \leftarrow 1$  to  $k$  do
39:       if  $status[i] = \text{ATTACKED}$  then
40:          $a \leftarrow RandomSelect(\mathcal{G}[i])$ 
41:          $t_i \leftarrow ApplyAttack(a, t_i)$ 
42:       end if
43:     end for
44:      $val_{\mathcal{D}} \leftarrow \bigcup_{i=1}^k t_i$ 
45:    $sim \leftarrow M3-1(val_{in}, val_{\mathcal{D}})$ 
46:    $perf \leftarrow M3-2(sim, test_{in})$ 
47:   return  $(sim, perf)$ 
48: end if

```

Table 1. Threshold sweep for clean-vs-attacked classification on the validation sets.

Threshold T (%)	FP Rate (%)	FN Rate (%)	Accuracy (%)
5	83.3	0.0	89.6
10	50.0	0.0	93.8
11	50.0	0.0	93.8
12	50.0	0.0	93.8
13	16.7	0.0	97.9
14	16.7	0.0	97.9
15	16.7	0.0	97.9
16	16.7	2.4	95.8
17	16.7	2.4	95.8
18	16.7	2.4	95.8
19	16.7	2.4	95.8
20	16.7	2.4	95.8

The following design choices are introduced for Module 2: Adversarial Attack Group Classification, which is responsible for categorizing detected attacks into predefined groups based on their characteristics. For adversarial attack classification, XGBoost [51] is selected as the classifier due to its strong predictive performance compared to other supervised learning methods. The model is trained using datasets from the PBD, where each dataset is subjected to all available adversarial attack types. Samples are then labeled as either Group 1: Iteration-Based Attacks or Group 2: Optimization- and Decision-Based Attacks, depending on which attack group the applied attack belongs to, forming the labeled training data for the classification task. When an incoming dataset is classified as fully attacked (Case 2), the trained classifier model is used to directly predict the corresponding attack group. For partially attacked datasets (Case 3), the specific attacked segments identified in Module 1 are isolated, and the classifier is then applied to those segments to determine the most likely attack group.

4. Experimental Setup

Dataset Description: We utilize six datasets from the UEA Multivariate Time-Series Classification (MTSC) archive [52]. The selected datasets cover Human Activity Recognition (HAR) and motion data recordings, which are treated as proxies for healthcare monitoring rather than as clinical data. Because wearable devices have shifted health observation outside clinical settings, body-worn motion and gesture signals have themselves become the input to health-relevant inference, making these benchmarks representative of such deployments while remaining standardized and reproducible. This selection is intended to evaluate the robustness and generalizability of the proposed method across different datasets within a single application domain. Multivariate datasets are prioritized since they reflect the complexity of real-world time-series problems, where multiple interrelated variables must be considered together to accurately capture underlying temporal patterns [23]. As summarized in Table 2, the datasets vary in training size, test size, number of dimensions, sequence length, and class count, allowing for comprehensive evaluation under varying data characteristics.

Hardware Setup: We use a PC equipped with an Intel Core i7-9700K CPU (8 cores), 32 GB of RAM, and a 16 GB NVIDIA GeForce RTX 2080 dedicated GPU.

Evaluation Metrics: We use three metrics: accuracy, F1-score, and attack success rate (ASR). Accuracy measures the proportion of correctly classified instances out of the total number of samples. F1-score evaluates the balance between precision and recall, making it well suited for datasets with class imbalance. ASR measures the effectiveness of adversarial

attacks by calculating the percentage of instances where model predictions are successfully altered.

Table 2. Selected datasets from the UEA repository.

Dataset	Train	Test	Dim.	Len.	Classes
NATOPS	180	180	24	51	6
UWaveGestureLibrary	120	320	3	315	8
Cricket	108	72	6	1197	12
ERing	30	270	4	65	6
BasicMotions	40	40	6	100	4
ArticulatoryWordRecognition	275	300	9	144	25

Dataset Similarity Calculation: We quantify dataset similarity using a custom CNN with two 1D convolutional layers, adaptive max-pooling, and dropout to extract features from both clean and attacked data. The final fully connected layer maps these features to class predictions. The resulting embeddings are then normalized using L2 normalization to ensure consistency and scale invariance. We use cosine similarity between the embeddings, which measures the angular similarity between two vectors in a multi-dimensional space:

$$\text{Cosine Similarity} = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} \quad (1)$$

where $\mathbf{A}$ and $\mathbf{B}$ are the embedding vectors. Cosine similarity ranges from 0 (orthogonal) to 1 (identical). To identify the most appropriate similarity metric for ReLATE+, we also evaluate performance using DTW [53], which aligns sequences by minimizing temporal distortions, and Wasserstein Distance [54], which quantifies distributional differences by computing the optimal transport cost between probability distributions. Based on the superior performance of the custom CNN combined with cosine similarity, we select it as the primary similarity metric.

Baselines: We compare ReLATE+ against four baseline approaches: random model selection, Oracle (best model), the worst-performing model and meta-feature-based selection [55]. All baseline comparisons are conducted using the same set of 14 deep learning models (see Module 0: Formation of PBD Step 1), ensuring consistency and fairness across evaluations. Random model selection uses a Monte Carlo approach, where an ML model is randomly chosen for each dataset. This process is repeated 1000 times, with the average performance calculated as the baseline score for random selection. Oracle represents the maximum accuracy recorded on the test data for each dataset among all models. Oracle reflects the upper performance bound achievable by the optimal model with exhaustive search and is usually computationally impractical. The worst-performing model represents the least effective model among all the possible DL models in the database, reflecting the lower performance bound. Meta-feature-based selection constitutes a classical meta-learning baseline that describes each dataset with hand-crafted meta-features (sequence length, number of classes, number of channels, the channel-averaged mean, standard deviation, and skewness). After z-score normalization, each incoming dataset is matched to its nearest repository dataset via Euclidean distance, and that matched dataset's top models are reused, isolating the benefit of ReLATE+'s learned CNN embeddings over hand-crafted similarity.

For the state-of-the-art comparison, we conceptualize it as a spectrum bounded by Oracle and Random Model Selection. The appropriate choice within this spectrum depends on the specific priorities and the available model space. If the application demands strict high performance—such as maximizing accuracy or ensuring strong adversarial

resilience—regardless of computational cost, Oracle represents the ideal, though computationally expensive, option. In contrast, if computational efficiency is the primary concern and performance is of lesser importance, Random Model Selection offers a lightweight alternative. However, for scenarios where both strong performance and computational efficiency are critical, ReLATE+ provides the most balanced and practical solution.

5. Results

5.1. ReLATE+ Performance

This section presents the outcomes of Module 1: Adversarial Attack Detection, Module 2: Adversarial Attack Group Classification, and Module 3: Most Resilient Model Selection, followed by a detailed overhead analysis, and an ablation study. For Case 1 and Case 2, all possible scenarios were evaluated. However, as Case 3 involves a stochastic process in which each data segment is randomly assigned to be either clean or attacked—and if attacked, an attack type is also randomly selected—five representative random scenarios were generated for evaluation due to the inherent randomness and combinatorial explosion of possible configurations.

5.1.1. Module 1: Adversarial Attack Detection Results

The objective of Module 1 is to classify incoming data into one of three categories: Case 1 (no attack), Case 2 (fully attacked), or Case 3 (partially attacked). The results from Module 1 demonstrate that our technique effectively distinguishes between clean, partially attacked, and fully attacked datasets.

Table 3 presents the results for Case 1: No Adversarial Attack. In the clean data setting, a lower attack detection rate is desirable, as it indicates the absence of false positives. These findings show that all but one dataset in the PBD were correctly classified as clean, with Fourier and Wavelet complementing each other, Fourier capturing global patterns, and Wavelet detecting localized variations, to ensure robustness. An exception occurred with the BasicMotions dataset. While the Wavelet Transform correctly identified the data as clean, the Fourier Transform produced a high attack detection rate. Since the final decision is based on the maximum detection score, this led to a misclassification. This single false positive is the practical cost of the maximum rule's trade-off: in exchange for one clean misclassification, the rule reliably detects attacks that only one of the two transforms can flag while the other misses them.

Table 3. Attack detection results for Case 1.

Dataset	Fourier (%)	Wavelet (%)
UWave	7.50	5.60
ERing	12.22	12.20
BasicMotions	100.00	0.00
Cricket	6.90	4.17
NATOPS	12.22	9.44
ArticularyWordRecognition	3.66	3.33

For Case 2, 40 out of 42 dataset–attack combinations (derived from 6 datasets and 7 attacks) were correctly identified as fully attacked using the combined approach, as shown in Table 4. In the table, the first value in each cell corresponds to the detection rate from the Fourier Transform, the second from the Wavelet Transform, and the bolded value represents the maximum of the two, which is used for final classification. The results reflect strong overall performance, with only two misclassifications across all evaluations. For the NATOPS dataset under FGSM, BasicIterative, MIM, and AutoPGD attacks, Wavelet

achieved moderate detection rates (50–60%), which resulted in misclassification as Case 3, since this range falls within the 13–87% detection threshold defined for partial attacks. In contrast, Fourier consistently produced higher detection rates (95–100%), correcting Wavelet’s output and underscoring the benefit of their combined use. Also in the case of DeepFool attacks on UWave and ERing, Wavelet demonstrated limited performance (41.25% and 57.03%, respectively), but Fourier compensated with significantly higher detection rates (87.71% and 100%). Similarly, Fourier failed to detect any attacks on the BasicMotions dataset, whereas Wavelet successfully corrected the classification. Since final decision is based on the maximum score between the two methods, these results further highlight their complementary strengths. This complementary behavior arises from the distinct characteristics of the two transforms: Fourier is more sensitive to global frequency-domain perturbations, making it effective for attacks that induce consistent spectral distortions, while Wavelet excels at capturing localized, transient anomalies in both time and frequency domains. However, for DeepFool attacks on NATOPS and Cricket, both Fourier and Wavelet achieved only around 50% detection accuracy, leading to misclassification as Case 3 instead of Case 2.

Table 4. Comparison of Fourier (F) and Wavelet (W) attack detection rates for Case 2. Format: F/W, all values in %. Bold indicates the higher of the two methods.

Attack	UWave	ERing	BasicMotions	Cricket	NATOPS	AWR
FGSM	100.0/100.0	100.0/97.8	98.0/97.5	0.0/ 100.0	100.0/61.1	100.0/100.0
BIM	100.0/100.0	100.0/92.2	0.0/ 100.0	100.0/100.0	96.7/50.6	100.0/100.0
DeepFool	87.7/41.3	100.0/57.0	0.0/ 100.0	54.2/30.6	15.6/15.6	98.7/ 100.0
MIM	100.0/100.0	100.0/92.6	0.0/ 97.5	100.0/100.0	100.0/61.1	98.7/ 100.0
AutoPGD	100.0/100.0	100.0/91.5	0.0/ 100.0	100.0/100.0	90.6/48.3	100.0/100.0
ElasticNet	100.0/100.0	100.0/100.0	0.0/ 100.0	100.0/100.0	100.0/100.0	100.0/96.93
Boundary	100.0/100.0	100.0/100.0	100.0/100.0	100.0/100.0	100.0/100.0	100.0/99.67

For Case 3, five random scenarios were evaluated for each dataset in the PBD. The detection behavior in Case 3 largely mirrored the outcomes of Cases 1 and 2. As shown in Table 5, detection results are reported in the format Fourier/Wavelet (Final Decision), where the final classification is based on the higher of the two scores. The Final Decision, which represents the attack intensity level—20%, 40%, 60%, or 80%—is estimated by selecting the closest matching level to this maximum score, allowing for a $\pm 10\%$ tolerance around each target. In BasicMotions, Fourier failed to detect both clean and attacked segments that were successfully identified by Wavelet, leading to correct final decisions. In contrast, DeepFool attacks on UWave and ERing were missed by Wavelet but detected by Fourier. Overall, only DeepFool attacks on NATOPS and Articulatory Word Recognition were not detected by either method, resulting in the only misclassifications in Case 3. This is due to DeepFool’s minimal and fine-grained perturbations, which are designed to closely follow the decision boundary and often leave little detectable signal distortion in either the time or frequency domain. All other segments were correctly classified.

After classifying the incoming data as partially attacked and estimating its attack intensity level as 20%, 40%, 60%, or 80%, each of the five segments was independently processed through Module 1 to determine whether it was attacked or clean. The attack detection performance at the segment level was consistent with the results observed in Case 2, showing the reliability of the underlying detection mechanism. These segment level predictions were then aggregated into a pattern (such as attacked/clean/attacked/clean/-clean) and passed to Module 2 to classify the attack group associated with the attacked segments of the data.

We examine the performance of Module 1 across different domains with two datasets: ArticulatoryWordRecognition (AWR) and UWaveGestureLibrary (UWave), which originate from distinct domains— Motion and HAR, respectively. Both datasets demonstrated reliable performance in Module 1, achieving high attack detection rates across clean, partially attacked, and fully attacked settings, with minimal false positives or negatives. Notably, AWR remained robust even against subtle perturbations, while UWave exhibited near-perfect detection across all adversarial methods, including DeepFool, which was particularly challenging for other datasets. This also indicates that the 13–87% thresholds hold beyond a single dataset, classifying inputs correctly across all six datasets, which span both the HAR and motion domains. Because they operate on normalized detection rates rather than dataset-specific signal magnitudes, the thresholds are not tied to the scale or domain of any particular dataset.

Table 5. Comparison of Fourier (F) and Wavelet (W) detection rates relative to Decision (D) baseline for each sequence. Format: F/W (D), all values in %. Bold indicates the higher value between Fourier and Wavelet.

Dataset	Sequence 1	Sequence 2	Sequence 3	Sequence 4	Sequence 5
	Format: F/W (D), all in %				
NATOPS	63 /44 (60)	28 / 28 (20)	47 /37 (40)	33/ 44 (40)	64 /45 (60)
UWave	63 / 63 (60)	27 /23 (20)	80 /77 (80)	44 /43 (40)	82 /71 (80)
Cricket	61/ 63 (60)	24 / 24 (20)	60 /50 (60)	42/ 43 (40)	72 /68 (80)
ERing	65 /63 (60)	30/ 34 (40)	82 /57 (80)	48 /46 (40)	83 /81 (80)
BasicMotions	0/ 63 (60)	0/ 30 (20)	0/ 80 (80)	0/ 45 (40)	0/ 85 (80)
AWR	62 /45 (60)	23 / 23 (20)	80 / 80 (80)	42 / 42 (40)	81 / 81 (80)
Ground Truth	60	20	80	40	80

5.1.2. Module 2: Adversarial Attack Group Classification Results

The aim of Module 2 is to classify the data flagged as attacked in Module 1 into one of two attack groups: Group 1: Iteration-Based Attacks or Group 2: Optimization- and Decision-Based Attacks. Since Case 1 corresponds to clean (unattacked) data, no experiments are performed for that scenario. For Case 2, all attack scenarios are evaluated. In Case 3, only the segments identified as attacked by Module 1 are isolated and evaluated. The results of these experiments are summarized in Table 6 for Case 2 and Table 7 for Case 3. According to the Table 6, the mean classification accuracy is 92.21% for Group 1 and 84.93% for Group 2, yielding an overall average accuracy of 88.69% for Case 2. The classification performance on isolated segments in Case 3 closely mirrors the results observed in Case 2, though with slightly lower overall accuracy as shown in Table 7. This consistency suggests strong alignment between the two cases, as the segments analyzed in Case 3 are essentially smaller portions of the same datasets used in Case 2, and the detected attacked segments retain similar characteristics to the fully attacked data.

To assess Module 2 across domains, we evaluate AWR (speech) and UWave (gesture). Both datasets demonstrated high classification accuracy when distinguishing between Group 1 and Group 2, effectively capturing the underlying attack group structures. Their performance remained consistent even when isolating and classifying individual attacked segments, indicating robust generalization at the segment level. While both performed well, AWR slightly outperformed UWave overall, particularly excelling in Group 2 scenarios, showcasing stability across varied segment compositions and reliability under more complex, mixed-attack conditions. That this group-level distinction holds across two datasets from distinct domains supports our claim that the grouping reflects the intrinsic mechanism

of each attack rather than dataset-specific properties, and is therefore not an artifact of our particular dataset selection.

Table 6. Classification accuracy and attack-group performance for each dataset for Case 2.

Dataset	Average Accuracy	Group 1 Accuracy	Group 2 Accuracy
NATOPS	90.26	97.41	84.90
UWaveGestureLibrary	85.67	88.71	81.61
Cricket	87.28	90.08	83.55
ERing	90.57	95.80	83.60
BasicMotions	88.43	89.17	87.88
ArticulatoryWordRecognition	89.90	92.11	88.06
Average	88.69	92.21	84.93

Table 7. Classification accuracy and attack-group performance for each dataset for Case 3.

Dataset	Average Accuracy	Group 1 Accuracy	Group 2 Accuracy
NATOPS	92.06	97.51	84.80
UWaveGestureLibrary	85.54	88.53	81.55
Cricket	87.18	89.83	83.65
ERing	90.40	95.49	83.62
BasicMotions	88.55	89.15	87.75
ArticulatoryWordRecognition	89.22	92.02	88.15
Average	88.83	92.08	84.92

5.1.3. Module 3: Most Resilient Model Selection Results

The goal of this module is to select the most robust model for the incoming data by identifying the most similar dataset in the PBD and training the best-performing resilient algorithms associated with that dataset on the incoming data. Table 8, Figures 4 and 5 present the results for Module 3. All results are averaged across datasets. The performance of ReLATE+ is compared against Oracle, random model selection, meta-feature-based selection and the worst-performing model. Table 8 shows ReLATE+'s performance in Case 1 (i.e., under no adversarial attack). Oracle refers to the highest accuracy achieved for each dataset, while random selection reflects the average accuracy across multiple randomized model selections. The worst-performing model corresponds to the lowest accuracy per dataset. Lastly, meta-feature-based selection reports the accuracy obtained by matching each dataset to its most similar counterpart via hand-crafted meta-features, then evaluating that counterpart's top three models and selecting the best. ReLATE+ achieves accuracy close to Oracle, with only a 1.55% gap, edges out meta-feature-based selection by 1.97% and outperforms random model selection by 12.45%. These results demonstrate ReLATE+'s ability to match datasets with appropriate models based on similarity, without the need for exhaustive testing.

Table 8. ReLATE+ results for Case 1 using accuracy.

Case	Oracle	ReLATE+	Meta-Learning	Random Model Selection	Worst Model Performance
Case 1	95.00	93.45	91.48	81.00	44.38

Figure 4 shows results for Case 2 across all attack types, along with their average performance across datasets, providing a comprehensive assessment of ReLATE+'s robustness

under different adversarial conditions. Figure 5 presents detailed results for each of the five randomized attack sequences in Case 3. The repetition of Case 3 five times accounts for the variability introduced by randomized attacks and evaluates ReLATE+’s consistency across diverse scenarios. In both cases, Oracle (red) is defined as the model with the lowest ASR for each dataset, while ReLATE+ (green) represents the proposed approach, and meta-feature-based selection (light blue) reports the result of hand-crafted meta-feature matching. Random selection (blue) performance is computed using a Monte Carlo approach, while the worst model (purple) corresponds to the highest average ASR. In Case 2, ReLATE+’s ASR is, on average, only 1.9% higher than Oracle’s, yet 16.3% lower than that of random model selection, and 17.8% lower than the worst-performing model, as well as 7.1% lower than meta-feature-based selection. Similarly, in Case 3, ReLATE+ exceeds Oracle’s ASR by just 2.6%, while outperforming random selection and the worst model by 11.1% and 15.5%, respectively, and meta-feature-based selection by 6.5%. This underscores ReLATE+’s ability to utilize dataset-specific similarities to recommend models with enhanced adversarial robustness. By aligning model recommendations with the feature distributions of datasets, ReLATE+ effectively mitigates the impact of adversarial attacks.

Overall, ReLATE+ performs consistently close to the Oracle in all cases, even when the performance gap between the worst and best-performing models is substantial. On average across Case 1, Case 2, and Case 3, ReLATE+ is within 2.02% of the Oracle, achieves a 13.28% improvement over random model selection, and outperforms the worst-performing model by 27.32%. ReLATE+ also consistently exceeds meta-feature-based selection across all cases, by an average margin of 5.19%, confirming that while hand-crafted meta-features provide a reasonable baseline, learned CNN embeddings yield more effective dataset similarity for resilient model selection. This highlights ReLATE+’s ability to achieve near-optimal performance and robustness across diverse adversarial scenarios, all while significantly reducing computational cost by avoiding exhaustive model testing.

Analysis on AWR and UWave consistently showed strong model selection results with ReLATE+, performing close to the oracle and outperforming all random, meta-feature-based and worst-case baselines. In Case 1, AWR nearly matched the oracle (99.00% vs. 99.33%), and UWave also performed competitively (89.44% vs. 93.89%). In Case 2, both achieved significantly lower ASRs than the baselines, with AWR at 23.6% and UWave at 42.9%, close to their respective oracle-level performances. In Case 3, averaged over five randomized sequences, ReLATE+ maintained low ASRs for both datasets—23.6% for AWR and 29.7% for UWave—remaining well below random, meta-feature-based and worst-case outcomes. These results highlight ReLATE+’s ability to generalize under varying attack patterns and reliably select robust models across domains.

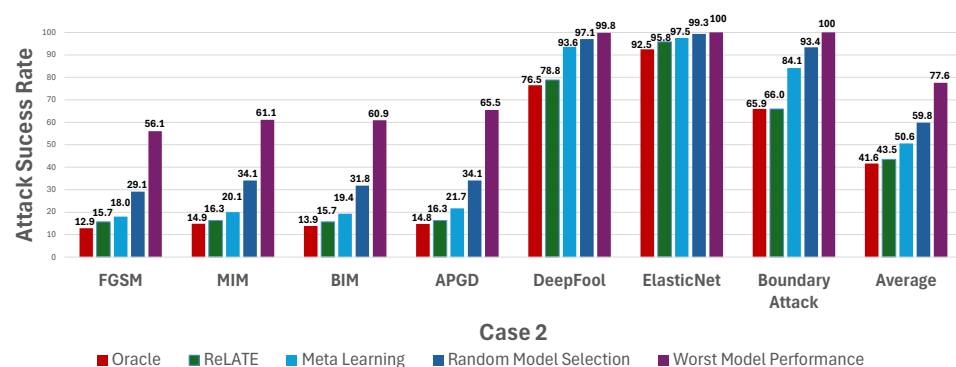


Figure 4. ReLATE+ Case 2 ASR results.

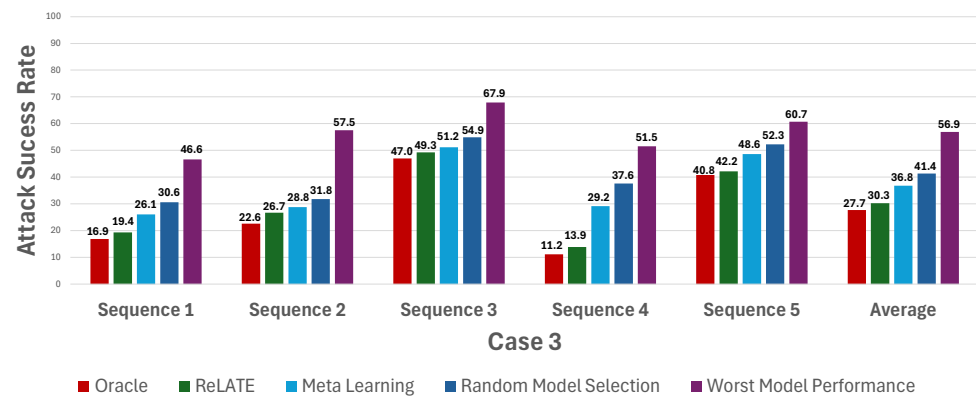


Figure 5. ReLATE+ Case 3 ASR results.

5.2. Overhead Analysis

In the absence of ReLATE+, to achieve optimal model performance (Oracle), all DL models have to be trained. Thus, Oracle overhead is determined by evaluating all models for each case. This process involves training each model and applying all adversarial attacks, ensuring a comprehensive assessment of their performance. ReLATE+ avoids this exhaustive per-dataset retraining by relying on a PBD, whose construction—training the 14 DL models on each dataset in PBD, generating their adversarial variants, and training the XGBoost classifier and the benchmark CNN similarity functions—is a one-time offline cost, taking approximately 14 h and producing a PBD under 1 MB. Unlike Oracle, which re-trains all models for every incoming dataset, this cost is paid once and amortized across arrivals, leaving the per-arrival savings below unaffected. At runtime, since ReLATE+ focuses on choosing the most similar dataset and evaluating only the top three models, it significantly reduces computational overhead. All overhead values reported are based on execution time measurements. In Module 1, its overhead is negligible compared to the Oracle. In Module 2, ReLATE+ requires only 3% of the Oracle’s cost to prepare the training data, train an XGBoost model, and perform prediction. In Module 3, it reduces model training and evaluation overhead by 85.12% in Case 1, 77.26% in Case 2, and 79.25% in Case 3, resulting in an average reduction of 80.54% across all cases. The additional cost of computing the similarity metric is minimal, contributing just 1.02%, 1.05%, and 0.53% of Oracle overhead in Cases 1, 2, and 3, respectively. When comparing total overhead, including all three modules, ReLATE+ achieves a reduction of 81.07% in Case 1, 73.21% in Case 2 and 75.73% in Case 3. Even when including ReLATE+’s internal framework costs, the overall reduction remains high at 77.68% on average. On the other hand, when we evaluate meta-feature-based selection from an overhead perspective, it shares the same top-three model selection and evaluation cost as ReLATE+, differing only in the similarity step. Since it relies on simple hand-crafted statistics rather than CNN embeddings, its similarity computation is lighter, reducing the similarity overhead by approximately 0.30% across the three cases. However, this cost saving is negligible next to the consistent performance gap in favor of ReLATE+, confirming that the learned similarity is well worth its minimal additional cost. These results highlight how ReLATE+ significantly reduces computational costs by efficiently selecting resilient models, ensuring both optimal performance and resource efficiency, even under adversarial conditions.

5.3. Performance Analysis of Similarity Metrics

We compare three well-known methods: our approach (CNN + Cosine Similarity), Dynamic Time Warping (DTW) [53], and Wasserstein Distance [54] as dataset similarity metrics, and evaluate their performance across Cases 1, 2 and 3. Figure 6 shows the average

percentage difference between Oracle and ReLATE+ when using each similarity metric. In Case 1, this difference is calculated as the accuracy difference percentage. In all other cases, it represents the ASR percentage difference with respect to the Oracle. While CNN combined with Cosine Similarity does not achieve the highest performance in every case, it exhibits the smallest average performance deviation and consistently performs near the optimal metric when not the best. It also achieves the highest computational efficiency, reducing computation time by 20% in Case 1, 15% in Case 2 and 18% in Case 3, averaging a 17.7% improvement over DTW and Wasserstein Distance, which exhibit nearly identical execution times.

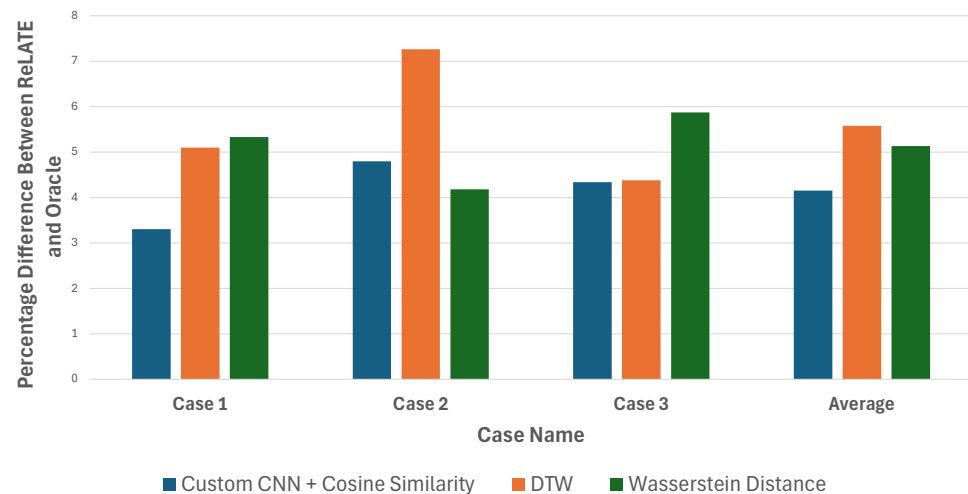


Figure 6. Similarity metric performance comparison.

5.4. Ablation Study on Epsilon in Adversarial Attacks

To evaluate the sensitivity of ReLATE+ to adversarial perturbation strength, we conducted an ablation study by varying the attack parameter ϵ , which controls the magnitude of perturbation in gradient-based attacks such as FGSM, BIM, AutoPGD, and MIM. While our primary experiments were conducted with $\epsilon = 0.1$, we repeated the analysis using $\epsilon = 0.001$ and $\epsilon = 0.2$ to assess performance under weaker and stronger attack settings respectively. Since Case 1 involves no attacks and Case 3 follows a randomized attack sequence that includes methods not governed by ϵ (such as DeepFool, ElasticNet, and Boundary Attack), changes in ϵ cannot be directly isolated in those scenarios. Accordingly, the ablation study was conducted solely on Case 2, which consists of fully attacked scenarios by a variety of adversarial methods; however, only the ϵ -based gradient attacks were considered for this evaluation.

In Module 1, as shown in Table 9, the detection performance varied notably with changes in ϵ . These results reflect average detection accuracy across all datasets. When $\epsilon = 0.01$, both Fourier- and Wavelet-based detection accuracy dropped significantly, indicating that subtle adversarial perturbations are more difficult to detect. In contrast, using $\epsilon = 0.1$ resulted in a sharp increase in detection rates across all attack types. When the attack strength was further increased to $\epsilon = 0.2$, Wavelet-based detection improved even further, reaching approximately 100% accuracy under several attacks. However, Fourier-based performance remained largely stable between $\epsilon = 0.1$ and $\epsilon = 0.2$. This stability is primarily attributed to the BasicMotions dataset, where Fourier-based detection fails to capture perturbations effectively at both levels of attack strength, thus limiting the average gain in overall performance.

Table 9. Adversarial attack detection accuracy (%) for Fourier and Wavelet methods under different ϵ values in Case 2, using attack implementations described in [18].

Attack	Fourier ($\epsilon = 0.01/0.1/0.2$)	Wavelet ($\epsilon = 0.01/0.1/0.2$)
FGSM	57.4/83.3/83.3	44.0/92.7/99.3
BIM	57.1/82.8/83.3	44.9/90.6/100.0
MIM	58.4/83.9/83.3	44.7/91.9/99.6
AutoPGD	53.5/81.8/83.3	42.7/90.3/100.0

In Module 2, we analyze the impact of perturbation strength on Group 1 classification accuracy across different datasets. Table 10 shows that model performance improves consistently as the attack strength increases from $\epsilon = 0.01$ to $\epsilon = 0.2$. These improvements were observed across all datasets. These findings confirm that ReLATE+ not only detects adversarial perturbations effectively but also maintains strong performance in classifying attack types as the perturbation magnitude increases. Lastly, in Module 3, we examine the effect of ϵ on resilient model selection using dataset similarity. ReLATE+ performance decreased by 0.42% for $\epsilon = 0.001$ and increased by 0.35% for $\epsilon = 0.2$ compared to the baseline $\epsilon = 0.1$. Taken together, these results justify our choice of $\epsilon = 0.1$ since at $\epsilon = 0.01$ the perturbations are too subtle to be reliably detected and exert little effect, whereas $\epsilon = 0.2$ improves detection only marginally while making the perturbation more visible. The value $\epsilon = 0.1$ thus represents the point at which the attack is both effective and detectable while remaining imperceptible, confirming it as a balanced parameter choice.

Table 10. Group 1 classification accuracy (%) across datasets under different perturbation strengths ϵ .

Dataset	Group 1 Accuracy ($\epsilon = 0.01$)	Group 1 Accuracy ($\epsilon = 0.1$)	Group 1 Accuracy ($\epsilon = 0.2$)
UWave	87.68	88.71	95.80
ERing	91.67	95.80	95.80
BasicMotions	84.37	89.17	90.22
Cricket	89.39	90.08	93.10
NATOPS	88.20	97.41	98.10
AWR	82.27	92.11	93.05
Average	87.26	92.21	94.34

5.5. Limitations and Potential Solutions

Our experiments reveal several conditions under which ReLATE+ is less effective. Detection degrades when perturbations are minimal and boundary-aligned: DeepFool was the only attack our Fourier–Wavelet rule failed to flag, causing all misclassifications in Case 2 (NATOPS, Cricket) and Case 3 (NATOPS, AWR), and the ablation confirms this is a perturbation-strength effect, with accuracy falling to 42.7–58.4% at $\epsilon = 0.01$. A second trade-off arises from the fusion rule itself, which deliberately favors caution and therefore occasionally flags a clean input, as observed once on BasicMotions in Case 1. Each such false positive triggers the downstream attack-group classification and similarity search unnecessarily, and the underlying assumption that a missed attack is more costly than a false alarm may not hold in lower-risk domains, where a symmetric fusion rule would be preferable. The threshold sweep in Table 1 further shows that the false-positive rate is stable across $T \in \{13, 14, 15\}\%$, but that the false-positive rate rises sharply below this range.

These observations point to natural extensions of the framework. A complementary behavior-based detector, such as prediction-margin consistency under input transformations, would help capture low- ϵ attacks that leave little trace in the time or frequency domain and could act as a tie-breaker when the two transforms disagree sharply. Similarly, a confidence check on the top cosine similarity score could trigger evaluation of a larger candidate pool whenever no close match is found in the PBD.

Beyond these extensions, we plan to evaluate ReLATE+ on domains with different temporal characteristics, such as industrial sensing and network traffic, to test whether the detection thresholds and the two-group attack taxonomy transfer without recalibration. We also intend to make the PBD incremental so that newly arriving datasets and newly published attacks are absorbed online rather than requiring a full offline rebuild.

6. Conclusions

The dynamic nature of time-series data makes it challenging to determine whether inputs are clean, adversarial, or incomplete. Deep learning models are well-suited to address this complexity, as they can learn temporal patterns directly from raw sequences. However, frequent retraining can be impractical due to high computational costs, especially in real-time settings with limited data. These challenges are further compounded by vulnerability to adversarial attacks, where even small perturbations can cause significant misclassifications. Such risks are especially critical in healthcare-oriented applications, where wearable and smart devices increasingly drive continuous monitoring outside clinical supervision, and misclassifying an adversarially perturbed signal could delay or misguide a clinical response. To address these issues, we propose ReLATE+, a resilient learner selection mechanism that integrates attack detection, attack group classification, and adaptive defense for time-series classification. It uses a three-stage detect–classify–select pipeline to reduce reliance on costly model retraining by dataset-level similarity. ReLATE+ flags adversarial inputs using Fourier and Wavelet features, classifies attacks with XGBoost, and selects top-performing models from similar datasets for resilient model selection. Experiments show it reduces computation by 77.68%, maintains accuracy within 2.02% of an oracle, and outperforms random selection by 13.28% on average.

Author Contributions: Conceptualization, C.I.K.; Methodology, C.I.K., O.G. and B.A.; Software, C.I.K.; Validation, O.G.; Investigation, C.I.K. and O.G.; Resources, O.G. and B.A.; Data curation, C.I.K.; Writing—original draft, C.I.K.; Writing—review & editing, C.I.K., O.G., T.R. and B.A.; Visualization, C.I.K.; Supervision, O.G., T.R. and B.A.; Project administration, T.R. and B.A.; Funding acquisition, T.R. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been funded in part by NSF, with award numbers #2112665, #2112167, #2003279, #2120019, #2211386, #2052809, #1911095 and in part by PRISM and CoCoSys, centers in JUMP 2.0, an SRC program sponsored by DARPA.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest

References

1. Gungor, O.; Rios, A.; Mudgal, P.; Ahuja, N.; Rosing, T. A robust framework for evaluation of unsupervised time-series anomaly detection. In *Proceedings of the International Conference on Pattern Recognition*; Springer: Cham, Switzerland, 2024; pp. 48–64.
2. Holder, C.; Middlehurst, M.; Bagnall, A. A review and evaluation of elastic distance functions for time series clustering. *Knowl. Inf. Syst.* **2024**, *66*, 765–809. <https://doi.org/10.1007/s10115-023-01952-0>.
3. Ismail Fawaz, H.; Forestier, G.; Weber, J.; Idoumghar, L.; Muller, P.A. Deep learning for time series classification: A review. *Data Min. Knowl. Discov.* **2019**, *33*, 917–963. <https://doi.org/10.1007/s10618-019-00619-1>.
4. Gungor, O.; Li, E.; Shang, Z.; Guo, Y.; Chen, J.; Davis, J.; Rosing, T. Rigorous Evaluation of Machine Learning-Based Intrusion Detection Against Adversarial Attacks. In *Proceedings of the 2024 IEEE International Conference on Cyber Security and Resilience (CSR)*; IEEE: Piscataway, NJ, USA, 2024; pp. 152–158. <https://doi.org/10.1109/CSR61664.2024.10679443>.
5. Guang, L.; Wang, X.; Li, R. Multi-Scale RCNN Model for Financial Time-series Classification. *arXiv* **2019**, arXiv:1911.09359. <https://doi.org/10.48550/arXiv.1911.09359>.

6. Kim, J.; Truong, K.P.; Englebienne, G.; Evers, V. Learning spectro-temporal features with 3D CNNs for speech emotion recognition. In *Proceedings of the 2017 Seventh International Conference on Affective Computing and Intelligent Interaction (ACII)*; IEEE: Piscataway, NJ, USA, 2017; pp. 383–388. <https://doi.org/10.1109/ACII.2017.8273628>.
7. Varlı, M.; Yılmaz, H. Multiple classification of EEG signals and epileptic seizure diagnosis with combined deep learning. *J. Comput. Sci.* **2023**, *67*, 101943. <https://doi.org/10.1016/j.jocs.2023.101943>.
8. Ordóñez, F.J.; Roggen, D. Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition. *Sensors* **2016**, *16*, 115. <https://doi.org/10.3390/s16010115>.
9. Moazemi, S.; Vahdati, S.; Li, J.; Kalkhoff, S.; Castano, L.J.V.; Dewitz, B.; Bibo, R.; Sabouniaghdam, P.; Tootooni, M.S.; Bundschuh, R.A.; et al. Artificial Intelligence for Clinical Decision Support for Monitoring Patients in Cardiovascular ICUs: A Systematic Review. *Front. Med.* **2023**, *10*, 1109411. <https://doi.org/10.3389/fmed.2023.1109411>.
10. Susanto, A.P.; Lyell, D.; Widiantoro, B.; Berkovsky, S.; Magrabi, F. Effects of Machine Learning-Based Clinical Decision Support Systems on Decision-Making, Care Delivery, and Patient Outcomes: A Scoping Review. *J. Am. Med. Inform. Assoc.* **2023**, *30*, 2050–2063. <https://doi.org/10.1093/jamia/ocad180>.
11. Suárez-Cetrulo, A.L.; Quintana, D.; Cervantes, A. A survey on machine learning for recurring concept drifting data streams. *Expert Syst. Appl.* **2023**, *213*, 118934. <https://doi.org/10.1016/j.eswa.2022.118934>.
12. Dempster, A.; Petitjean, F.; Webb, G.I. ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels. *Data Min. Knowl. Discov.* **2020**, *34*, 1454–1495. <https://doi.org/10.1007/s10618-020-00701-z>.
13. Finlayson, S.G.; Bowers, J.D.; Ito, J.; Zittrain, J.L.; Beam, A.L.; Kohane, I.S. Adversarial attacks on medical machine learning. *Science* **2019**, *363*, 1287–1289. <https://doi.org/10.1126/science.aaw4399>.
14. Ozbulak, U.; Vandersmissen, B.; Jalalvand, A.; Couckuyt, I.; Van Messem, A.; De Neve, W. Investigating the significance of adversarial attacks and their relation to interpretability for radar-based human activity recognition systems. *Comput. Vis. Image Underst.* **2021**, *202*, 103111.
15. Kocal, C.I.; Gungor, O.; Tartz, A.; Rosing, T.; Aksanli, B. ReLATE: Resilient Learner Selection for Multivariate Time-Series Classification Against Adversarial Attacks. In *Proceedings of the 2025 IEEE International Conference on Cyber Security and Resilience (CSR)*; IEEE: Piscataway, NJ, USA, 2025; pp. 419–424. <https://doi.org/10.1109/CSR64739.2025.11130062>.
16. Papernot, N.; McDaniel, P.; Goodfellow, I. Transferability in Machine Learning: From Phenomena to Black-Box Attacks using Adversarial Samples. *arXiv* **2016**, arXiv:1605.07277. <https://doi.org/10.48550/arXiv.1605.07277>.
17. Karim, F.; Majumdar, S.; Darabi, H. Adversarial Attacks on Time Series. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *43*, 3309–3320. <https://doi.org/10.1109/TPAMI.2020.2986319>.
18. Costa, J.C.; Roxo, T.; Proença, H.; Inácio, P.R.M. How Deep Learning Sees the World: A Survey on Adversarial Attacks & Defenses. *IEEE Access* **2024**, *12*, 61113–61136. <https://doi.org/10.1109/ACCESS.2024.3395118>.
19. Hsieh, T.Y.; Wang, S.; Sun, Y.; Honavar, V. Explainable Multivariate Time Series Classification: A Deep Neural Network Which Learns to Attend to Important Variables As Well As Time Intervals. In *WSDM '21: Proceedings of the 14th ACM International Conference on Web Search and Data Mining*; Association for Computing Machinery: New York, NY, USA, 2021; pp. 607–615. <https://doi.org/10.1145/3437963.3441815>.
20. Lin, G.; Shen, C.; Lin, A. Higher-order Cross-structural Embedding Model for Time Series Analysis. *arXiv* **2024**, arXiv:2410.22984. <https://doi.org/10.48550/arXiv.2410.22984>.
21. Zheng, Y.; Liu, Q.; Chen, E.; Ge, Y.; Zhao, J.L. Exploiting multi-channels deep convolutional neural networks for multivariate time series classification. *Front. Comput. Sci.* **2016**, *10*, 96–112. <https://doi.org/10.1007/s11704-015-4478-2>.
22. Grabocka, J.; Wistuba, M.; Schmidt-Thieme, L. Fast classification of univariate and multivariate time series through shapelet discovery. *Knowl. Inf. Syst.* **2016**, *49*, 429–454. <https://doi.org/10.1007/s10115-015-0905-9>.
23. Ruiz, A.P.; Flynn, M.; Large, J.; Middlehurst, M.; Bagnall, A. The great multivariate time series classification bake off: A review and experimental evaluation of recent algorithmic advances. *Data Min. Knowl. Discov.* **2021**, *35*, 401–449. <https://doi.org/10.1007/s10618-020-00727-3>.
24. Baldán, F.J.; Benítez, J.M. Multivariate times series classification through an interpretable representation. *Inf. Sci.* **2021**, *569*, 596–614. <https://doi.org/10.1016/j.ins.2021.05.024>.
25. Cong, X.Y.; Yu, Z.H.; Fanti, M.P.; Mangini, A.M.; Li, Z.W. Predictability verification of fault patterns in labeled Petri nets. *IEEE Trans. Autom. Control* **2025**, *70*, 1973–1980.
26. Marks, R.E. Validation and model selection: Three similarity measures compared. *Complex. Econ.* **2013**, *2*, 41–61. <https://doi.org/10.7564/13-COEC10>.
27. Bounliphone, W.; Belilovsky, E.; Blaschko, M.B.; Antonoglou, I.; Gretton, A. A Test of Relative Similarity For Model Selection in Generative Models. *arXiv* **2016**, arXiv:1511.04581. <https://doi.org/10.48550/arXiv.1511.04581>.
28. Assegie, T.A.; Salau, A.O.; Omeje, C.O.; Braide, S.L. Multivariate sample similarity measure for feature selection with a resemblance model. *Int. J. Electr. Comput. Eng.* **2023**, *13*, 3359–3366. <https://doi.org/10.11591/ijece.v13i3.pp3359-3366>.

29. Cazelles, E.; Robert, A.; Tobar, F. The Wasserstein-Fourier Distance for Stationary Time Series. *IEEE Trans. Signal Process.* **2021**, *69*, 709–721. <https://doi.org/10.1109/TSP.2020.3046227>.
30. Xie, F.; Wang, C.; Li, X. Dynamic Time Warping for Time Series Curve Similarity. *Online J. Robot. Autom. Technol.* **2024**, *3*, OJRAT.MS.ID.000553. <https://doi.org/10.33552/OJRAT.2024.03.000553>.
31. Harford, S.; Karim, F.; Darabi, H. Adversarial Attacks on Multivariate Time Series. *arXiv* **2020**, arXiv:2004.00410. <https://doi.org/10.48550/arXiv.2004.00410>.
32. Galib, A.H.; Bashyal, B. On the Susceptibility and Robustness of Time Series Models through Adversarial Attack and Defense. *arXiv* **2023**, arXiv:2301.03703. <https://doi.org/10.48550/arXiv.2301.03703>.
33. Siddiqui, S.A.; Dengel, A.; Ahmed, S. Benchmarking adversarial attacks and defenses for time-series data. In *Proceedings of the International Conference on Neural Information Processing*; Springer: Cham, Switzerland, 2020; pp. 544–554. https://doi.org/10.1007/978-3-030-63836-8_45.
34. Gungor, O.; Rosing, T.; Aksanli, B. STEWART: STacking Ensemble for White-Box Adversarial Attacks Towards more resilient data-driven predictive maintenance. *Comput. Ind.* **2022**, *140*, 103660. <https://doi.org/10.1016/j.compind.2022.103660>.
35. Metzen, J.H.; Genewein, T.; Fischer, V.; Bischoff, B. On Detecting Adversarial Perturbations. *arXiv* **2017**, arXiv:1702.04267. <https://doi.org/10.48550/arXiv.1702.04267>.
36. Xu, W.; Evans, D.; Qi, Y. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks. In *Proceedings of the 2018 Network and Distributed System Security Symposium*; Internet Society: Reston, VA, USA, 2018. <https://doi.org/10.14722/ndss.2018.23198>.
37. Lee, K.; Lee, K.; Lee, H.; Shin, J. A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks. In *Proceedings of the Advances in Neural Information Processing Systems*; Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2018; Volume 31.
38. Harford, S.; Attia, J.; Lipton, Z.C. RADAR: Recurrent Autoencoder Based Detector for Adversarial Examples on Temporal EHR. In *Proceedings of the ACM Conference on Health, Inference, and Learning (CHIL)*; ACM: New York, NY, USA, 2020; pp. 162–172. <https://doi.org/10.1145/3368555.3384448>.
39. Gao, Y.; Peng, H.; Ma, H.; Dai, Z.; Wang, S.; Hu, H.; Fu, A.; Xue, M. From Pixels to Trajectory: Universal Adversarial Example Detection via Temporal Imprints. In *Proceedings of the IEEE Transactions on Dependable and Secure Computing*; IEEE: Piscataway, NJ, USA, 2026; pp. 5542–5558. <https://doi.org/10.1109/TDSC.2026.3656508>.
40. Wang, Z.; Yan, W.; Oates, T. Time series classification from scratch with deep neural networks: A strong baseline. In *Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN)*; IEEE: Piscataway, NJ, USA, 2017; pp. 1578–1585. <https://doi.org/10.1109/IJCNN.2017.7966039>.
41. Karim, F.; Majumdar, S.; Darabi, H.; Chen, S. LSTM Fully Convolutional Networks for Time Series Classification. *IEEE Access* **2018**, *6*, 1662–1669. <https://doi.org/10.1109/ACCESS.2017.2779939>.
42. Elsayed, N.; Maida, A.S.; Bayoumi, M. Deep Gated Recurrent and Convolutional Network Hybrid Model for Univariate Time Series Classification. *Int. J. Adv. Comput. Sci. Appl.* **2019**, *10*. <https://doi.org/10.14569/ijacsa.2019.0100582>.
43. Bai, S.; Kolter, J.Z.; Koltun, V. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv* **2018**, arXiv:1803.01271. <https://doi.org/10.48550/arXiv.1803.01271>.
44. Karim, F.; Majumdar, S.; Darabi, H.; Harford, S. Multivariate LSTM-FCNs for time series classification. *Neural Netw.* **2019**, *116*, 237–245. <https://doi.org/10.1016/j.neunet.2019.04.014>.
45. Fawaz, H.I.; Lucas, B.; Forestier, G.; Pelletier, C.; Schmidt, D.F.; Weber, J.; Webb, G.I.; Idoumghar, L.; Muller, P.-A.; Petitjean, F. Inceptiontime: Finding alexnet for time series classification. *Data Min. Knowl. Discov.* **2020**, *34*, 1936–1962. <https://doi.org/10.1007/s10618-020-00710-y>.
46. Zou, X.; Wang, Z.; Li, Q.; Sheng, W. Integration of residual network and convolutional neural network along with various activation functions and global pooling for time series classification. *Neurocomputing* **2019**, *367*, 39–45. <https://doi.org/10.1016/j.neucom.2019.08.023>.
47. Tang, W.; Long, G.; Liu, L.; Zhou, T.; Blumenstein, M.; Jiang, J. Omni-Scale CNNs: A simple and effective kernel size configuration for time series classification. *arXiv* **2022**, arXiv:2002.10061. <https://doi.org/10.48550/arXiv.2002.10061>.
48. Fauvel, K.; Lin, T.; Masson, V.; Fromont, É.; Termier, A. XCM: An Explainable Convolutional Neural Network for Multivariate Time Series Classification. *Mathematics* **2021**, *9*, 3137. <https://doi.org/10.3390/math9233137>.
49. Zhang, Z.; Zeng, Y.; Liu, Q.; Zhou, S. Towards a Novel Perspective on Adversarial Examples Driven by Frequency. *arXiv* **2024**, arXiv:2404.10202. <https://doi.org/10.48550/arXiv.2404.10202>.
50. Kiruluta, A.; Lemos, A. A Hybrid Wavelet-Fourier Method for Next-Generation Conditional Diffusion Models. *arXiv* **2025**, arXiv:2504.03821. <https://doi.org/10.48550/arXiv.2504.03821>.
51. Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In *KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*; Association for Computing Machinery: New York, NY, USA, 2016; pp. 785–794. <https://doi.org/10.1145/2939672.2939785>.

52. Bagnall, A.; Dau, H.A.; Lines, J.; Flynn, M.; Large, J.; Bostrom, A.; Southam, P.; Keogh, E. UEA Multivariate Time Series Classification Repository. 2018. Available online: <https://www.timeseriesclassification.com/dataset.php> (accessed on 2 January 2026).
53. Petitjean, F.; Ketterlin, A.; Gançarski, P. A Global Averaging Method for Dynamic Time Warping, with Applications to Clustering. *Pattern Recognit.* **2011**, *44*, 678–693. <https://doi.org/10.1016/j.patcog.2010.09.013>.
54. Flamary, R.; Cuturi, M.; Courty, N.; Rakotomamonjy, A. Wasserstein Discriminant Analysis. *Mach. Learn.* **2017**, *107*, 1923–1945. <https://doi.org/10.1007/s10994-018-5717-1>.
55. Uddin, S.; Lu, H. Dataset meta-level and statistical features affect machine learning performance. *Sci. Rep.* **2024**, *14*, 1704. <https://doi.org/10.1038/s41598-024-51825-x>.

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