Dynamic Multi-Objective Ensemble Control for Cost-Aware Intrusion Detection Systems

Gaetano Alessi, Mariagrazia Fugini

Department of Electronics, Information and Bioengineering, Politecnico di Milano, Milan, Italy
gaetano1.alessi@mail.polimi.it, mariagrazia.fugini@polimi.it

Abstract. Network intrusion detection is a critical component of network security monitoring, and it is essential to detect intrusions in real time. This paper proposes a resource-aware orchestration framework that redefines Network Intrusion Detection as a control problem, addressing the trade-off between high-fidelity deep learning architectures and the latency constraints of high-throughput streams. Our approach integrates heterogeneous classifiers within a weighted soft-voting ensemble architecture designed to provide detection performance while managing computational load. Existing ensemble methods typically aggregate all models at all times, incurring high computational cost regardless of threat complexity. To avoid this, our framework promotes or prunes models using a real-time utility function based on accuracy, diversity, and latency. Experimental evaluation on diverse datasets indicates that non-linear models contribute to detection of complex threat vectors, while lightweight linear filters demonstrate adequate performance for attacks where aggressive streaming learners exhibit catastrophic forgetting. Results quantify these trade-offs, showing a reduction in the variance of adaptive components across the evaluated datasets, and providing an approach for designing AI components within security infrastructures.

1 Introduction

In a digital environment, protecting private information and infrastructures using only traditional security solutions is challenging. Recent reports indicate an increase in the most common types of cybersecurity attacks, such as automated Distributed Denial of Service (DDoS) attacks, Man-in-the-Middle (MITM) interceptions, and sophisticated ransomware campaigns. Cyberattacks are increasing in volume and severity, and there is a growing mismatch between the scale of cyberattacks and the capacity of organizations to respond. AI-enabled intrusions, including fake-based fraud, further expand the threat landscape. In this context, the Network Intrusion Detection System (NIDS) serves as the primary sensor and line of defense. However, the operational landscape of these systems has shifted in the last decade. Traditional signature-based NIDS, while computationally efficient and effective against known threats, exhibit severe limitations in identifying zero-day attacks and addressing anomaly detection in encrypted streams (TLS/SSL). They rely on static rule-sets that are reactive; by

the time a signature is generated and deployed, the adversary has often pivoted. To overcome these limitations, the cybersecurity research community has progressively adopted Machine Learning (ML) and Deep Learning (DL) techniques [8, 23]. These methods demonstrate capabilities in anomaly detection by learning complex, non-linear patterns from network data (flow statistics, inter-arrival times), enabling them to identify deviations from normal behavior with detection rates exceeding those of baseline signature-based methods without inspecting encrypted payloads. The central challenge lies in the trade-off between **performance** (detection accuracy via F1-score) and **operational efficiency** (maintaining low inference latency during traffic spikes). We define operational efficiency as the system's ability to dynamically allocate computational resources based on threat complexity, avoiding static resource consumption. Although complex ML and DL architectures excel at learning hierarchical representations from network telemetry [15], they introduce significant computational overhead. Most state-of-the-art approaches continuously deploy resource-intensive ensembles to ensure maximum coverage. Although effective in traditional offline evaluations (e.g., using KDD-99), this static approach is inadequate in real-world, high-throughput environments where computational bottlenecks can lead to dropped packets and system failures. To address this, we propose a framework driven by three concurrent objectives: (1) maximizing detection performance (F1-score), (2) minimizing computational inference latency, and (3) maintaining ensemble diversity to improve generalization. Furthermore, to satisfy the critical need for alert verification, the framework provides post-hoc explainability as a passive auditing tool. These challenges motivate the contribution of this research: the design and evaluation of ELENIDS [6], a hybrid framework that balances detection accuracy with time efficiency by integrating DL components in an ensemble architecture. The system is built around the needs of passive IDS operation, where model transparency and report readability influence an analyst's ability to confirm threats and respond quickly. To support this, we define a minimal but complete feature partition that keeps inference light while maintaining semantic clarity for human operators. The objective is to contribute to anomaly-detection research by providing empirical analysis of how DL, ensemble strategies, and explainability tools can jointly operate within detection pipelines [2].

This paper is organized as follows. Section 2 reviews the state of the art. Section 3 formalizes the system model, details the ensemble architecture, and defines the controller logic governing the multi-objective utility function. Section 4 presents the experimental setup and performance-cost trade-off analysis. Finally, Section 5 concludes the paper.

2 Related Work

Many studies have applied DL techniques in the implementation of network intrusion detection. For instance, Javaid et al. [10] used a stacked autoencoder (SAE) on the UNSW-NB15 dataset, reporting higher accuracy metrics compared to traditional ML methods. Similarly, Kim et al. [11] applied a Deep Belief Net-

work (DBN) to the KDD Cup 1999 dataset, highlighting DL's ability to handle high-dimensional data. The development of novel and efficient DL architectures tailored for intrusion detection remains an active research area [12], with applications extending to specialized domains such as automotive CAN bus monitoring and securing IoT environments against threats [20]. Despite their high accuracy, a limitation in many of these studies is the lack of analysis concerning the computational resources required for training and real-time inference. A primary goal of current research (e.g., [3]) is to develop classifiers that not only achieve high accuracy but also maintain minimal false alarm rates, particularly when dealing with the high-dimensionality characteristic of network data. To this end, researchers are exploring hybrid approaches that combine the strengths of both DL and ML, as well as examining the robustness of these models against potential adversarial threats. To mitigate the weaknesses of individual classifiers and improve overall detection accuracy, ensemble learning has become a popular strategy; Sridevi et al. [21] proposed a supervised learning-based voting classifier that yielded improved accuracy over baseline methods. More recently, Milosevic et al. [13] explored a weighted voting ensemble using DL-based classifiers on the CICIDS-2017 dataset. Further exploring this trend, Jain et al. [9] proposed a large, hybrid ensemble of eleven ML and DL models. Emanet et al. [7] proposed VEL-IDS, a heterogeneous ensemble utilizing soft-voting based on classifier accuracy; similarly, Sunday et al. [22] introduced dynamic weighting mechanisms. Both approaches maintain all models in an active voting state throughout operation, without state transitions based on computational cost.

Recent works explore multi-stage ensemble strategies that trade accuracy for runtime efficiency: Ren et al. propose boosting-based ensembles that combine adaptive undersampling with adversarial augmentation (ADHS-EL) and incremental undersampling ensembles (DUEN) to tackle severe class imbalance while keeping computational overhead low [17]. Agate et al. present MIDES, a three-stage pipeline that routes flows through a fast binary filter, specialized detectors, and a meta-classifier, using per-flow decision routing to reduce average processing time [1]. Parallel to these algorithmic advances, literature explicitly targets cost-aware orchestration at the infrastructure level: some studies integrate cost-sensitive loss functions or class-balanced weighting to improve recall on rare attacks while controlling false alarms [4]. While dynamic selection and infrastructure orchestration have advanced, these domains remain isolated. Methods lack per-flow control, whereas adaptive selection schemes overlook inference overhead; similarly, adversarial defense is seldom evaluated against latency, and explainability serves as an analysis tool rather than an operational driver. We address these gaps with a streaming orchestration layer that jointly optimizes execution cost, training expense, and robustness. By decoupling execution from influence and embedding explainability within the utility function, the proposed framework enforces a direct trade-off between detection gain and computational cost.

3 Our Approach

The framework models the NIDS as a resource allocation problem, combining ML and DL classifiers in a parallelized ensemble for attack detection while keeping inference latency practical. The architecture operates on an infinite data stream, formalized as a sequence:

$$\mathcal{S} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_t, y_t), \ldots\} = \{(x_t, y_t)\}_{t=1}^{\infty}$$

where $x_t \in R^d$ represents the d-dimensional feature vector extracted from network telemetry at time t, and $y_t \in \{0, 1\}$ represents the associated ground truth label indicating benign or malicious activity. In the domain of high-speed network intrusion detection, the data generating process is non-stationary. The statistical properties of $P(x, y)$ evolve over time due to changes in user behavior and the emergence of novel attack vectors. Consequently, a static ensemble design, which aggregates a fixed set of predictors trained on a historical snapshot, is suboptimal in non-stationary environments. It enforces a constant consumption profile regardless of the threat landscape, potentially resulting in energy expenditure during benign periods and latency bottlenecks during volumetric attacks. To address this, we model the NIDS as a resource allocation problem. The system state at any time t is defined by the tuple $(\mathcal{M}, \Sigma, \Omega)$, where $\mathcal{M} = \{M_1, \ldots, M_K\}$ constitutes a heterogeneous pool of base classifiers, $\Sigma = \{\sigma_1, \ldots, \sigma_K\}$ represents the operational models status vector, and $\Omega = \{w_1, \ldots, w_K\}$ denotes the voting weights. The status σ_k of a model M_k is governed by a finite state machine managed by the *ensemble controller*. The controller partitions the classifier pool into three operational states:

- *active ($\sigma_k = 1$):* the model processes the input x_t, updates its internal weights (if online learning is enabled), and contributes its vote to the final ensemble decision, incurring the full computational cost.
- *shadow ($\sigma_k = 0$):* the model processes the input and updates its weights, but its prediction is excluded from the vote. Performance is tracked via prequential evaluation to detect emerging relevance under concept drift. The state incurs processing cost while isolating the ensemble from predictions of ill-suited models.
- *disabled ($\sigma_k = -1$):* the model is dormant. It performs no processing and incurs zero cost. This state is used for expensive models that are statistically redundant.

The operational context is a discrete-time control process governed by reporting windows W_j, which enforce temporal order and separate the validation protocol into three sequential phases: initialization, streaming adaptation, and post-hoc auditing. The protocol begins by sampling a finite, sequential segment of the historical data stream to constitute the initialization set, which is partitioned into a training fold and a disjoint validation fold. The primary objective of this phase is to overcome the cold start problem inherent in streaming systems by establishing a baseline competence. During this phase, every candidate classifier $M_k \in \mathcal{M}$ undergoes supervised training. Simultaneously, hyperparameter

optimization is conducted to locate the static configuration for each estimator. The initial voting weights $w_{k,0}$ are derived not from training accuracy, but from the performance metrics computed on the validation set. This means the ensemble's initial state prioritizes models with validated generalization capabilities, preventing the propagation of overfitting into the streaming phase.

The streaming phase simulates the continuous learning operational loop by modeling prediction latency, model adaptation, and resource contention. For each incoming instance x_t, the subset of active models generates independent probabilistic predictions. Feedback latency is simulated: the true label y_t may not be immediately available, governed by a Poisson distribution, decoupling inference from learning. Upon label arrival, the system triggers adaptation routines: instance-incremental learners perform immediate updates, while batch-incremental learners accumulate instances in a buffer until the window boundary, where batch retraining occurs at each window boundary. At each window boundary, the controller scores all ensemble members against recently labeled window data, driving the pruning logic.

Upon completion of the simulated stream, the collected metrics (including temporal performance, stability, and error patterns) are aggregated and analyzed. This phase supports the XAI and resilience validation goals, because it quantifies the total average computational cost and the utilization rate to validate the efficiency of the pruning strategy against the resource consumption goal. Furthermore, to satisfy requirements for explainability, the system generates global feature importance rankings and derives surrogate rules, providing transparent insights into the ensemble's long-term decision-making profile.

3.1 Base Classifiers

Real-world traffic processing begins with the capture of raw packets, typically in the form of PCAP files or live interface streams, using tools to extract flow-level features. This generates connection logs containing tuple information (source/dest IP, ports, protocol) and statistical features (duration, bytes sent/received, packet inter-arrival times). To handle the heterogeneity of real-world data, we apply a robust scaler (using median and quantile range) rather than standard scaling (mean/variance). Categorical features (e.g. protocol, service) are numerically encoded; we avoid one-hot encoding for fields with high cardinality to maintain dimensionality constraints. We apply Recursive Feature Elimination with Cross-Validation (RFECV), using a Random Forest estimator and optimizing PR-AUC rather than accuracy to prioritize minority-class detection under severe imbalance. The evaluation uses k-fold stratified cross-validation to ensure the selected subset generalizes.

The classifier pool comprises two learning paradigms. Online learners update their weights with each sample, providing low-latency adaptation. **Logistic Regression (LR)** scales input features via a running statistic scaler and filters linearly separable threat patterns, preserving the budget for heavier models. **Adaptive Random Forest (ARF)** maintains an ensemble of Hoeffding trees and manages drift via ADWIN detectors attached to leaf nodes. Upon detecting

a shift in leaf error rates, the algorithm trains a background tree in parallel; the replacement occurs only once the background tree surpasses the current estimator, avoiding catastrophic forgetting.

Batch learners accumulate samples in a buffer B_j over window W_j and retrain at the window boundary. The higher per-window cost is offset by stronger non-linear capacity. **Decision Tree (DT)** provides an interpretable baseline. **XGBoost (XGB)** handles non-linear classification via gradient boosting; other architectures such as **Explainable Boosting Machines (EBM)** were excluded due to training latency incompatible with streaming update requirements. To prevent overfitting to the current window and to mitigate catastrophic forgetting of historical patterns, the system maintains a FIFO buffer of the N most recent XGBoost estimators (e.g., $N = 3$). Each new window trains a fresh estimator that is added to the buffer while the oldest is discarded; predictions are computed as the averaged output of this temporal ensemble, preserving memory of recent traffic history. An enhancement within our batch learning pipeline is the integration of *adversarial training*. Modern intrusion detection systems must operate in adversarial environments where attackers actively attempt to evade detection by perturbing traffic features (e.g., modifying inter-arrival times or packet padding). To enhance robustness, the batch update process for the XGBoost component triggers an adversarial training loop. We utilize the HopSkipJump attack algorithm [5], a decision-based evasion attack that approximates the decision boundary direction through binary queries to generate adversarial perturbations. It is important to note that these perturbations are generated in the *feature space* rather than at the raw packet level. Although this represents a theoretical worst-case manipulation of the network statistics rather than a deployable payload modification, it provides a rigorous upper bound for evaluating model robustness. For a given batch B_j, the system identifies a subset of samples and generates their adversarial counterparts x'_t such that the model misclassifies them. The batch learners are then retrained on the augmented dataset $B'_j = B_j \cup \{(x'_t, y_t)\}$. This process regularizes the model by constraining the decision boundary around the data manifold.

The **Self-Reinforcement Attention (SRA)** network provides a deep learning component. The attention score a_i for each feature x_i is derived from the scaled alignment of the corresponding latent vectors, defined as:

$$a_i = \frac{\mathbf{q}_i \cdot \mathbf{k}_i}{\sqrt{p_k}} \tag{1}$$

where p_k denotes the attention head dimension and $\mathbf{q}_i, \mathbf{k}_i$ are the row vectors for feature i in the projected matrices. These scores generate a reinforced representation $\mathbf{o} = \mathbf{a} \odot \mathbf{x}$ via element-wise multiplication, effectively filtering input features according to the instance context. A final `LinearConstraint` layer aggregates this representation to yield the prediction, enabling global weights β to indicate feature importance while the attention vector $\mathbf{a}$ provides local explanations.

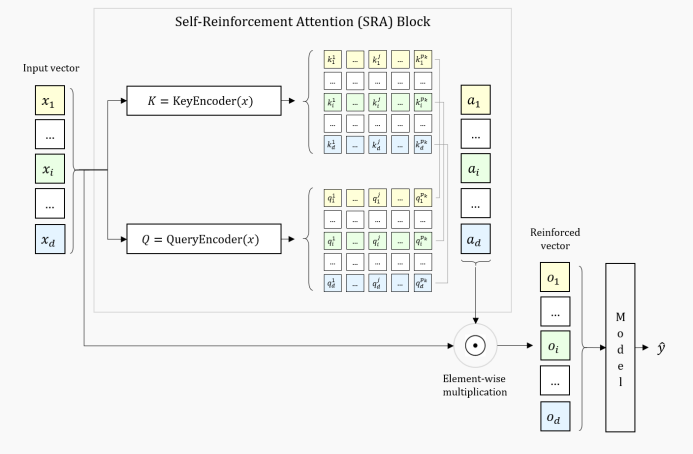

Fig. 1. The SRA block projects the input vector x into matrices K and Q to derive the attention weights a. The architecture computes the reinforced vector o via the element-wise multiplication $o = a \odot x$. This operation scales the original features before the aggregation model generates the prediction $\hat{y}$.

3.2 Ensemble Model Design

The output of these diverse classifiers must be synthesized into a single decision. In non-stationary environments, the reliability of a classifier is not constant; a model trained on HTTP traffic may perform poorly when the stream shifts to SMTP traffic. This motivates a recalibration mechanism at each window boundary. Let $S_{k,j}$ be the performance metric of model k evaluated on window W_j. We utilize the F1-score for $S_{k,j}$ because it penalizes both false positives and false negatives, a property essential for imbalanced network traffic. This ensures that a model's voting weight directly scales with its ability to maintain precision and recall on unseen streaming data, rather than its historical training fit. Furthermore, to stabilize the ensemble against transient fluctuations and the cold start problem of newly instantiated models, we introduce a support-based regularization term. Let N_k denote the cumulative support, representing the total number of samples observed by model k since its instantiation. The voting weight $w_{k,j}$ is defined as:

$$w_{k,j} = S_{k,j} \cdot (1 + \alpha \cdot \ln(1 + N_k)) \tag{2}$$

The parameter α controls the bias towards stable models. The logarithmic term ensures that models with extensive validation are prioritized, preventing the ensemble from being influenced by a low-support model that achieves a high score on a small, yet unrepresentative, sample of data. The ensemble probability for a class c (e.g., malicious) is computed as the normalized weighted sum of the probabilities $P_k(y = c|\mathbf{x})$ predicted by the subset of currently active classifiers:

$$P(y = c|\mathbf{x}) = \frac{\sum_{k \in \mathcal{M}} I(\sigma_k = 1) \cdot w_{k,j} \cdot P_k(y = c|\mathbf{x})}{\sum_{k \in \mathcal{M}} I(\sigma_k = 1) \cdot w_{k,j}} \tag{3}$$

where $I(\cdot)$ is the indicator function that evaluates to 1 if the model status σ_k is active and 0 otherwise. This soft-voting mechanism allows the ensemble to capture the confidence of constituent models, providing a more discriminative decision boundary than hard majority voting. The orchestration of this system is governed by the *ensemble controller*, which executes a resource-aware pruning logic at the boundary of each reporting window. We formulate the controller's objective as an optimization problem: to select a subset $\mathcal{M}^* \subseteq \mathcal{M}$ that maximizes the ensemble's joint detection accuracy while adhering to strict constraints on computational cost and predictive redundancy. To quantify the value of each constituent classifier, we define a multi-objective *utility score* $U_{k,j}$ for the k-th model at the conclusion of the j-th window:

$$U_{k,j} = S_{k,j} - \gamma \cdot C_k \tag{4}$$

In this formulation, C_k is the normalized cost of model k, aggregating inference latency and amortized training overhead. The penalty γ regulates the trade-off between detection fidelity and latency: without it ($\gamma = 0$), the optimization favors heavy DL models even when simpler linear classifiers would suffice. By tuning γ, the system demotes expensive models unless their accuracy gain justifies the cost. While the *utility score* $U_{k,j}$ provides a baseline ranking of individual competence, relying only on this metric leads to the *ensemble redundancy problem*: aggregating models that exhibit correlated error patterns yields diminishing returns; for example, adding a Decision Tree to an ensemble dominated by a similar tree adds computational cost without improving the decision boundary. To resolve this, we introduce a selection logic based on *marginal utility*. This approach posits that a model should only be promoted to the active voting pool if it corrects specific errors made by the current ensemble, rather than merely reinforcing existing correct predictions. The selection process iterates through the candidate models ranked by $U_{k,j}$. A candidate model M_k is integrated into the current active set only if it satisfies a strict criterion. The model must provide a marginal gain in performance, defined as the increase in the ensemble's joint AUC observed when M_k is added to the vote. To prevent system instability driven by stochastic noise, this gain must exceed a predefined sensitivity threshold ϵ. In our experimental configuration, we empirically set $\epsilon = 0.05$. This value was chosen through preliminary testing to prevent high-frequency oscillations (i.e., constantly turning models on and off due to micro-drifts) while requiring a minimum 5% improvement in detection capability to justify the computational overhead of an additional voter.

However, optimizing strictly for marginal accuracy in the current window W_j risks overfitting to short-term traffic patterns, discarding models that are robust to different attack vectors not present in W_j. To mitigate this risk and ensure long-term robustness, the controller implements an alternative activation path based on statistical orthogonality. Even if a candidate model yields a marginal gain lower than ϵ, it may be retained if it exhibits high diversity relative to the active set. We quantify this via the Pearson correlation coefficient ρ between the probability vectors of M_k and the current ensemble average. If the correlation

remains below a diversity threshold τ_{div} (set empirically to 0.95 to tolerate minor prediction overlaps while pruning near-identical learners), the model is deemed distinct and is therefore not discarded.

Models that fail both criteria are removed from the active pool. Rather than fully discarding them, which would require retraining upon reactivation, demoted models enter the *shadow* state: they continue updating their parameters against the stream without contributing to voting, keeping their distributions synchronized at training cost only. If a model's $U_{k,j}$ falls below a lower bound (expensive and inaccurate), it transitions to the *disabled* state, halting all processing. This three-tier management scales computational allocation to current threat complexity at each window boundary.

4 Experimental Setup and Results

This section presents the empirical evaluation of the proposed framework, quantifying the advantages of the ensemble architecture relative to its constituent base classifiers. The performance of the framework is contingent upon the optimal configuration of its components; consequently, each base classifier underwent an independent hyperparameter tuning process to maximize its individual efficacy prior to integration.

We benchmark the adaptive ensemble against these standalone models and a static ensemble baseline to evaluate whether the resource allocation strategy yields improved detection metrics relative to individual components and unpruned ensembles.

4.1 Dataset Preprocessing and Hyperparameter Tuning

To ensure the generalizability of our findings across distinct threat landscapes, experiments have been conducted on a diverse suite of four benchmark datasets, which were selected to stress-test the framework against varying degrees of class imbalance, feature dimensionality, and traffic velocity. The **UNSW-NB15** dataset [14] serves as a primary benchmark for network intrusion, characterized by a significant class imbalance where malicious activity constitutes approximately 3.2% of the 700 000 analyzed instances. To simulate a realistic environment, we utilized the **CIC-IDS-2017** dataset [19], which provides a temporally ordered sequence of traffic containing benign flows and modern denial of service attacks; the minority class represents roughly 36% of the traffic, presenting a more balanced but highly volumetric threat scenario. For the IoT domain, we employed the **CICIoT2023** dataset [16], which is dominated by volumetric DDoS floods. The **CIDDS-002** dataset [18] provided a large-scale evaluation ground with over 8 million flows, where normal activity constitutes 96.3% of the stream. Consistency in evaluation was enforced through a preprocessing pipeline embedded within the testbed. First, to establish a uniform binary classification task across all domains, target labels were binarized: benign states (e.g., Normal, BenignTraffic) were mapped to the negative class ($y = 0$), while all attack

vectors were consolidated into the positive class ($y = 1$). Second, to simulate a streaming environment and prevent look-ahead bias, strict temporal ordering was enforced. Datasets containing timestamp information (CIC-IDS-2017 and CIDDS-002) were sorted chronologically prior to ingestion, ensuring that the prequential evaluation respected the causality of the data generating process. To ensure numerical stability for the heterogeneous mix of linear and non-linear classifiers, categorical features were encoded using category codes, and features were standardized; this transformation is critical for models like the LR, which are sensitive to the scale of input variables.

Model specifications were tuned to maximize the ratio of predictive power to computational cost. The ARF was restricted to 20 active trees to address non-linear drift within strict memory bounds, preventing the latency penalties associated with standard 100-tree configurations. XGBoost was implemented with a bagging strategy and a maximum tree depth of 9 to mitigate sequential boosting latency. Finally, the SRA network was constrained to 5 epochs per window; this decision prioritizes rapid adaptation over asymptotic convergence, ensuring the controller promotes the deep learner only when linear baselines fail to detect current threat vectors.

4.2 Evaluation Methodology and Metrics

The data stream for each experiment was truncated to a fixed horizon of 100 000 instances; this volume provides sufficient statistical power to observe multiple cycles of concept drift while remaining within the computational bounds of repeated trials. The first 60 000 samples served for the pre-training phase to prime the base classifiers and calibrate the initial feature selection (RFECV), while the remaining 40 000 samples constituted the streaming evaluation phase. The temporal granularity of this evaluation was governed by a window of 2 500 instances. This parameter was selected as the optimal trade-off between statistical significance and reaction speed; it provides a sample size large enough to calculate robust error metrics, minimizing the variance associated with short-term noise, and small enough to allow the system to detect and react to short-duration volumetric attacks within a relevant time frame.

Given the severe class imbalance within network datasets, standard accuracy provides an overly optimistic view of performance and is therefore insufficient as a primary discriminative metric. Consequently, detection performance is primarily quantified and optimized via the Area Under the ROC Curve (AUC), the Area Under the Precision-Recall Curve (PR-AUC), and the F1-Score. Beyond accuracy, the framework validates resilience through two critical operational metrics: *inference latency*, which measures the average time required to process a single flow vector, and *model utilization rate*, which tracks the percentage of the operational timeline during which heavy classifiers are active. The controller parameters were set as defined in Section 3: marginal utility threshold $\epsilon = 0.05$ and diversity threshold $\tau_{\text{div}} = 0.95$.

Furthermore, a fixed decision threshold is not suitable for streaming settings in which the class probability distribution varies over time. For this reason, the

system employs a thresholding mechanism controlled through an adaptive F-beta criterion. The initial value of the beta parameter was set to 1.8: this configuration biases the optimization objective toward Recall during the early deployment phase, allowing a higher false positive rate in order to limit false negatives while the model behavior stabilizes. Experimental results indicate that this mechanism produced a threshold shift from 0.3427 to 0.4452 during the first windows of the CIC-IDS-2017 dataset as the system aligned with the observed event distribution. To regulate the long-term operating point, a beta decay process was applied, reducing β linearly to a lower bound of 0.5 across 10 windows, which reorients the objective toward Precision. In addition, an adaptive trigger was defined: when the average classification error exceeds a defined value (e.g., 0.03) within a window, the β parameter is reset, restoring Recall prioritization until the error rate returns below the specified limit.

4.3 Results and Comparative Analysis

Results on the UNSW-NB15 and CIC-IDS-2017 datasets quantify performance-efficiency trade-offs. As illustrated in Figure 2, ROC curves reveal a performance hierarchy on CIC-IDS-2017. The ARF model performed well, maintaining high ROC-AUC and F1-score across the streaming horizon. With a standard deviation of 0.0204, this observation is consistent with Hoeffding trees' reported capability in handling non-stationary intrusion data. However, the ARF recorded a processing latency of approximately 4.62 ms per flow, representing a higher latency overhead relative to linear classifiers and a relevant constraint in high-throughput environments. The linear baseline implemented via LR achieved a competitive mean PR-AUC of 0.9704 on UNSW-NB15 while requiring only 0.36 ms per flow. This suggests that a portion of the threat landscape in this dataset exhibits linear separability, consistent with the controller's logic to prioritize lightweight linear filters over complex non-linear solvers during periods of standard traffic. Across both datasets, the ensemble maintained an average of 3.1 active models per window (K=5 for CIC-IDS-2017, K=6 for UNSW-NB15). ARF was active in 25% of windows on UNSW-NB15 and was disabled on CIC-IDS-2017 after window 2 due to a FPR of 0.38. The controller prioritized lightweight models, activating ARF or SRA only when marginal utility exceeded the threshold. The SRA network achieved a mean PR-AUC of 0.9618 with a standard deviation of 0.0612, consistent with the 5-epoch training constraint. The SRA component transitioned between active and shadow states in 56% of windows, activated when the correlation threshold indicated distinct predictions from the linear baseline. The controller demoted SRA to shadow state during stable periods, promoting it only when complex feature interactions, undetectable by LR or ARF, were evident. Analysis of forgetting and generalization, measured via the fixed validation set, revealed a negative slope for ensemble metrics. This phenomenon is consistent with the plasticity-stability dilemma in continuous learning; as the model adapts to new attack vectors, its representation of historical traffic shifts. On CIC-IDS-2017, the ensemble achieved a PR-AUC of 0.9999, ranking first among all configurations, but the LR baseline achieved a PR-AUC of 0.9994.

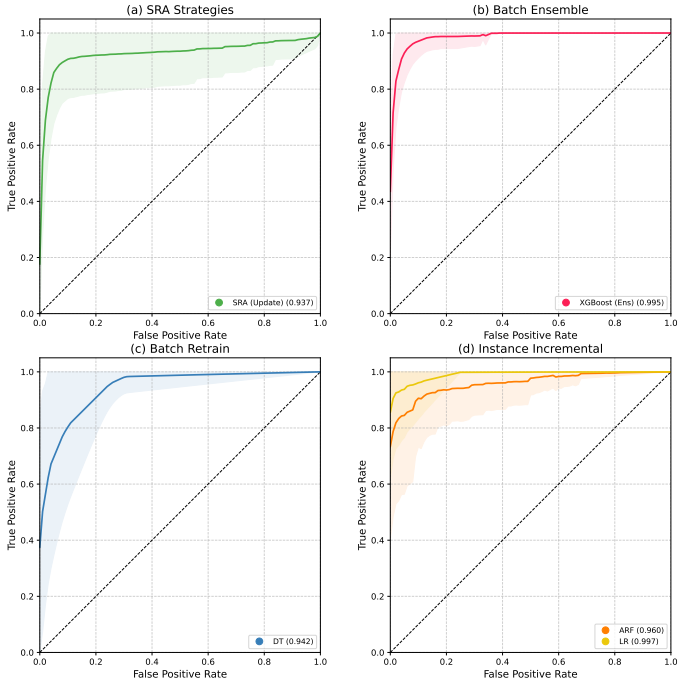

Fig. 2. Receiver Operating Characteristic (ROC) curves for each classifier on the CIC-IDS-2017 dataset. The analysis shows a performance hierarchy driven by the volumetric nature of the dataset. In the Instance Incremental panel (d), the linear baseline (LR) outperforms the ARF, while the SRA (a) exhibits a lower AUC of 0.937 and a wider interval (shaded area), reflecting the higher variance associated with training deep models under real-time streaming constraints.

This suggests that the volumetric signatures of attacks in this specific time-frame are linearly separable, contrasting with the non-linear patterns observed in the UNSW dataset.

A key differentiator in this scenario is the stability of the models in the presence of concept drift. The analysis of the "forgetting slope" on CIC-IDS-2017, measured by evaluating the models on a fixed historical validation set as the stream progresses, revealed that ARF maintained high accuracy on the immediate stream with a PR-AUC of 0.9991, but suffered from catastrophic forgetting, indicated by a 40.8% drop on the fixed validation set by the end of the stream. Furthermore, the ARF exhibited a mean False Positive Rate (FPR) of 0.38, suggesting that its drift adaptation mechanism (ADWIN) reacted to the bursty nature of the traffic, mistaking benign anomalies for threats. Conversely, the ensemble architecture mitigated this instability. Through the weighted voting mechanism, which penalizes models with high variance, the framework reduced the impact of the ARF's variable predictions, resulting in a forgetting slope of -3.6% and an FPR of 0.08, compared to ARF's standalone FPR of 0.38.

Unlike sequential stacking, our system executes base classifiers in concurrent threads, ensuring that the total system latency is bounded by the single slowest active component plus marginal orchestration overhead (approximately 2%). The profiling data identifies XGBoost as the primary latency contributor at approximately 50.00 ms per flow in its active state. In a static ensemble, this would impose a permanent bottleneck; however, the utility-based pruning logic transforms this from a fixed cost into a variable, controller-managed cost. In the volumetric scenarios observed in the CIC-IDS-2017 dataset, the ensemble controller identified that the marginal performance gain of the boosting component did not justify its computational demand. Accordingly, the system automatically transitioned the heavy estimators to the disabled state, collapsing the aggregate inference latency to match the combined profile of the LR and SRA models. While the DL component (SRA) requires a substantial window for backpropagation (approximately 32 seconds per batch), the linear baselines update in 0.73 seconds. Thus, the fast linear filters adapt to screen volumetric threats, maintaining protection while the deeper architectures reorganize their feature representations. The ensemble achieved the highest overall precision and F1-Score (Figure 3), indicating that the utility-driven orchestration maintains detection accuracy while reducing computational cost compared to a static ensemble.

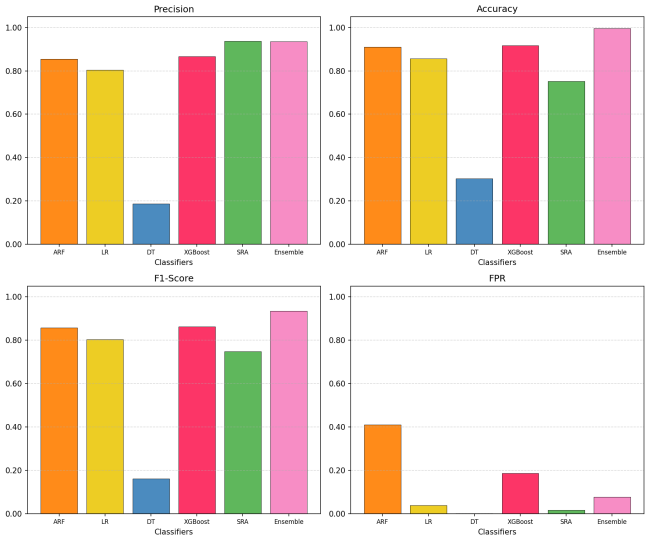

Fig. 3. Performance comparison of classifiers and the ensemble on four metrics (mean values over the stream) on the CIC-IDS-2017 dataset. The ensemble achieves the highest Accuracy, Precision, and F1-Score. For the False Positive Rate (FPR), the ensemble shows lower values than the ARF alone, though higher than LR and SRA individually.

5 Concluding Remarks

The framework shows that decoupling model execution from influence allows utility-driven allocation, where heterogeneous classifiers are managed according to their marginal contribution and computational cost. The experimental evaluation confirmed that ensemble utility is context-dependent: non-linear models dominate in complex threat landscapes, while lightweight linear filters suffice for volumetric attack scenarios, where aggressive streaming learners are prone to catastrophic forgetting. Future research will extend the utility function to explicitly incorporate interpretability metrics, such as surrogate fidelity scores, thereby allowing the controller to penalize black-box models in favor of explainable alternatives when predictive performance is comparable.

References

1. Agate, V., De Paola, A., Ferraro, P., Lo Re, G.: MIDES: A multi-layer Intrusion Detection System using ensemble machine learning. International Journal of Intelligent Networks **6**, 204–223 (2025). https://doi.org/https://doi.org/10.1016/j.ijin.2025.09.001
2. Alessi, G., Fugini, M.: Real-Time Fraud Detection Using Machine Learning. In: 2025 33rd International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises. pp. 1–6. IEEE (2025). https://doi.org/10.1109/WETICE67341.2025.11092218
3. Alsaffar, A.M., Nouri-Baygi, M., Zolbanin, H.: Enhancing Intrusion Detection Systems with Dimensionality Reduction and Multi-Stacking Ensemble Techniques. Algorithms **17**(12) (2024). https://doi.org/10.3390/a17120550
4. Binsawad, M.: Enhancing Network Intrusion Detection Systems through Cost-Sensitive Ensemble Learning with the CS-Forest Approach for Accurate Detection of Minority-Class Attacks (07 2025). https://doi.org/10.21203/rs.3.rs-7187958/v1
5. Chen, J., Jordan, M., Wainwright, M.: HopSkipJumpAttack: A Query-Efficient Decision-Based Attack. pp. 1277–1294 (05 2020). https://doi.org/10.1109/SP40000.2020.00045
6. Cusano, V., Fattibene, E., Fugini, M., Amarilli, F.: ELENIDS: An EnsembLE Network-based Intrusion Detection System. In: Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications (AICCSA 2025). pp. 1–6 (10 2025). https://doi.org/10.1109/AICCSA66935.2025.11315334
7. Emanet, S., Baydogmus, G.K., Demir, Ö.: An ensemble learning based IDS using voting rule: VEL-IDS. PeerJ Comput. Sci. **9**, e1553 (2023). https://doi.org/10.7717/PEERJ-CS.1553
8. Fugini, M.: Cyber Risk and Cyber Security: Cyber Access Control with Data Mining. Open Access Biostatistics Bioinformatics **3** (08 2024). https://doi.org/10.31031/OABB.2024.03.000575
9. Jain, N., Singh, M.P., Chaurasia, K., Ravindran, G.: Intrusion Detection System Using Ensemble Techinque. In: 2023 3rd International Conference on Innovative Sustainable Computational Technologies (CISCT). pp. 1–5 (2023). https://doi.org/10.1109/CISCT57197.2023.10351427

10. Javaid, A., Niyaz, Q., Sun, W., Alam, M.: A Deep Learning Approach for Network Intrusion Detection System. In: Proceedings of the 9th EAI International Conference on Bio-Inspired Information and Communications Technologies. p. 21–26. ICST (2016). https://doi.org/10.4108/eai.3-12-2015.2262516
11. Kim, T., Kang, B., Rho, M., Sezer, S., Im, E.G.: A Multimodal Deep Learning Method for Android Malware Detection Using Various Features. IEEE Transactions on Information Forensics and Security **14**(3), 773–788 (2019). https://doi.org/10.1109/TIFS.2018.2866319
12. Kimanzi, R., Kimanga, P., Cherori, D., Gikunda, P.: Deep Learning Algorithms Used in Intrusion Detection Systems – A Review (02 2024). https://doi.org/10.48550/arXiv.2402.17020
13. Milosevic, M., Ciric, V., Milentijevic, I.: Network Intrusion Detection Using Weighted Voting Ensemble Deep Learning Model. In: 2024 11th International Conference on Electrical, Electronic and Computing Engineering (IcETRAN). pp. 1–6 (2024). https://doi.org/10.1109/IcETRAN62308.2024.10645137
14. Moustafa, N., Slay, J.: UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: 2015 Military Communications and Information Systems Conference (MilCIS). pp. 1–6 (11 2015). https://doi.org/10.1109/MilCIS.2015.7348942
15. Muneer, S., Farooq, U., Athar, A., Ahsan Raza, M., Ghazal, T.M., Sakib, S.: A Critical Review of Artificial Intelligence Based Approaches in Intrusion Detection: A Comprehensive Analysis. Journal of Engineering **2024**(1), 3909173 (2024). https://doi.org/https://doi.org/10.1155/2024/3909173
16. Neto, E.C.P., Dadkhah, S., Ferreira, R., Zohourian, A., Lu, R., Ghorbani, A.A.: CICIoT2023: A Real-Time Dataset and Benchmark for Large-Scale Attacks in IoT Environment. Sensors (2023), https://www.mdpi.com/1424-8220/23/13/5941
17. Ren, H., Tang, Y., Ren, S., Wang, R., Dong, W.: ADHS-EL: Dynamic ensemble learning with adversarial augmentation for accurate and robust network intrusion detection. Journal of King Saud University Computer and Information Sciences **37** (2025), https://api.semanticscholar.org/CorpusID:277339435
18. Ring, M., Wunderlich, S., Grüdl, D., Landes, D., Hotho, A.: Creation of flow-based data sets for intrusion detection. Journal of Information Warfare **16**(4), 41–54 (2017), https://www.jstor.org/stable/26504117
19. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In: International Conference on Information Systems Security and Privacy (2018), https://api.semanticscholar.org/CorpusID:4707749
20. Sharma, S., Kumar, V., Dutta, K.: Multi-objective optimization algorithms for intrusion detection in IoT networks: A systematic review. Internet of Things and Cyber-Physical Systems **4**, 258–267 (2024). https://doi.org/https://doi.org/10.1016/j.iotcps.2024.01.003
21. Sridevi, S., Prabha, R., Reddy, K.N., Monica, K., Senthil, G., Razmah, M.: Network Intrusion Detection System using Supervised Learning based Voting Classifier pp. 01–06 (2022). https://doi.org/10.1109/IC3IOT53935.2022.9767903
22. Sunday, B., Abdullahi, Y., Matemilola, A., Sahabi, Y., Abdullahi, M.: Voting Ensemble Learning Model with Dynamic Weighting for Improved Network Intrusion Detection. Franklin Open **12**, 100375 (09 2025). https://doi.org/10.1016/j.fraope.2025.100375
23. Walling, S., Lodh, S.: An Extensive Review of Machine Learning and Deep Learning Techniques on Network Intrusion Detection for IoT **36**(2) (Feb 2025). https://doi.org/10.1002/ett.70064