International Journal of Computer Applications Technology and Research
Volume 14–Issue 06, 51 – 82, 2025, ISSN:-2319–8656
DOI:10.7753/IJCATR1406.1004

Self-Optimizing AI Agents for Real-Time Security Enforcement in Dynamic Broadband Infrastructures

Kamaldeen Oladipo
Nokia Technologies, Nigeria, MEA

Jude Ogedegbe
Dept of Intl Business and Data analysis, Ulster University, UK

Phebe E. Olufemi
Department of Computer Science, MAU, Nigeria

Vincent Onaji
Dept of Systems Eng, Purdue University, Fort Wayne, USA

Abstract: This paper explores a novel framework for deploying self-optimizing AI agents designed to enforce real-time security policies across dynamic broadband infrastructures. Given the rise of zero-touch networks, increasing traffic heterogeneity, and growing cyber threats, conventional reactive security methods are no longer sufficient. We propose an architecture that combines reinforcement learning (RL), federated observability, and edge-native threat detection. The paper introduces a scalable agent-based model with proactive anomaly detection and self-adjustment capabilities. Key contributions include a hybrid decision loop, a risk-weighted policy optimizer, and an adaptive trust index. The proposed solution is validated through simulations and real-world telecom KPIs. The results demonstrate enhanced mean time to detect (MTTD), reduced false positives, and improved threat response efficiency.

Keywords: AI agents, self-optimization, broadband infrastructure, real-time security, federated learning, network observability, reinforcement learning, edge AI, anomaly detection, zero-trust, threat intelligence, telecom KPIs

1. INTRODUCTION

1.1 Background: Dynamic Threats in Next-Gen Broadband

The proliferation of high-speed broadband networks including 5G, fiber-to-the-home (FTTH), hybrid fiber-coaxial (HFC), and satellite-based internet has catalyzed digital transformation across industries. This evolution has simultaneously introduced new security vulnerabilities due to the unprecedented volume, velocity, and variety of network traffic. Emerging broadband ecosystems now demand real-time responsiveness, particularly in multi-access edge computing (MEC) environments, where latency-sensitive services operate at the network edge (Zhang et al., 2023).

Broadband infrastructure has also become more software-defined, disaggregated, and virtualized. These changes, while enabling scalability and flexibility, create a broader attack surface. Threat vectors such as distributed denial-of-service (DDoS) attacks, lateral movement exploits, and adversarial AI injections now target both centralized and edge infrastructure (Kumar & Sharma, 2022). Conventional network operations center (NOC)-based security, reliant on human oversight and rule-based systems, struggles to scale under such dynamic threat conditions. The future of broadband security requires decentralized, intelligent agents that can operate autonomously at the edge with minimal latency.

1.2 Problem Statement: Latency and Scalability Limits of NOC-Based Security

Traditional NOC-based security architectures were built for static threat models and centralized control. However, as broadband networks expand, the core-to-edge latency incurred by centralized decision-making mechanisms has become a bottleneck. For instance, anomaly detection and mitigation that require telemetry aggregation at the core can delay responses to threats at the edge by several milliseconds an unacceptable delay for mission-critical applications such as autonomous vehicle telemetry or industrial IoT (Hussain et al., 2023).

Furthermore, the manual configuration and policy enforcement processes managed by human operators in NOCs are inherently non-scalable. The volume of network events in a dynamic 5G infrastructure exceeds human cognitive limits. This results in delayed threat detection, increased false positive rates, and missed zero-day exploits. There is an urgent need for an architectural shift that integrates AI-native agents capable of autonomous policy enforcement and real-time adaptation.

1.3 Research Gap: Absence of Self-Improving Agents in Edge-Dominant Networks

Despite the growing body of research in AI-driven cybersecurity and federated learning, there remains a significant gap in the implementation of self-optimizing AI agents that can learn from local data, share insights federatively, and evolve their policies over time. Most existing models treat AI as a supplementary analytical layer rather than a decision-making entity capable of enforcing network policies independently (Alshahrani & Qureshi, 2022).

Additionally, existing approaches often ignore the heterogeneity and volatility of edge environments. Policies optimized for one access point or user type may not generalize across geographic or usage variations. Hence, a framework that enables adaptive, location-specific, and trust-aware self-optimization remains largely unexplored in the broadband infrastructure space.

1.4 Objectives and Contributions

The primary objective of this research is to develop and validate a self-optimizing AI agent architecture tailored for real-time security enforcement in dynamic broadband infrastructures. This system is designed to address key latency and adaptability challenges posed by modern, high-throughput networks. One major goal is to minimize the Mean Time to Detect (MTTD) anomalies by employing a hybrid reinforcement learning (RL) model that continuously adapts to shifting traffic patterns and threat behaviors at the network edge.

Additionally, the system seeks to automate trust calibration among agents through the implementation of an Adaptive Trust Score (ATS) function. This trust function dynamically evaluates each agent's behavior by assigning scores based on its detection accuracy and decision consistency over time, weighted by the relevance and recency of its observed evidence. This ensures that more reliable agents are prioritized in federated learning cycles, while underperforming or potentially compromised agents are progressively excluded from security-critical operations.

Where represents time-weighted evidence confidence, and the evidence score for agent at time .

- Integrate explainability (XAI) mechanisms using SHAP values to interpret agent decisions.
- Enable federated retraining loops using edge telemetry without compromising user privacy.

The key contributions include:

1. A modular architecture for real-time, edge-native security enforcement.
2. A trust-aware, RL-enhanced decision model with self-adjustment capabilities.
3. Empirical validation on real-world datasets and synthetic broadband threat simulations.

1.5 Scope and Limitations

This study focuses primarily on broadband infrastructures with edge computing support, such as fiber-optic gateways, 5G radio access networks (RAN), and DOCSIS 4.0 cable nodes. While the architecture can be extended to hybrid cloud environments, the current scope is limited to on-premise and near-edge deployments where latency and security are critical.
Limitations of the study include: Despite the strengths of the proposed approach, several limitations are recognized in this study. First, the framework is developed under the assumption of semi-cooperative edge environments, meaning that all participating agents are presumed to behave honestly and without malicious intent. As such, malicious edge nodes or adversarial participants are considered beyond the scope of the current implementation and require separate mitigation strategies in future iterations. Second, the model training process relies heavily on labeled datasets such as CIC-IDS2018 and UNSW-NB15. This dependence may limit the framework's generalizability to real-world traffic conditions where labeled data is either scarce or unavailable. Third, evaluation constraints arise from the lack of access to live telecom-grade datasets containing classified threat intelligence, which restricts validation of the model under production-level operating conditions.

The subsequent chapters will present the architectural blueprint, algorithmic components, experimental configuration, and performance metrics used to validate the proposed system under controlled but realistic simulation environments.

2. LITERATURE REVIEW

2.1 Overview of AI-Driven Security in Broadband Networks

Over the past decade, broadband infrastructures have evolved from monolithic, centrally managed networks into distributed, software-defined ecosystems that support heterogeneous access technologies including 5G, DOCSIS, FTTH, and mmWave. As this transformation unfolds, so too has the attack surface expanded, leading to a rapid surge in network-layer and application-layer threats (Zhao et al., 2023).

Security in broadband networks now necessitates real-time detection, automated remediation, and explainable decisions. Traditional Intrusion Detection Systems (IDS) such as Snort and Suricata although highly effective in static environments, are incapable of adaptive responses or context-aware threat interpretation when deployed across dynamic broadband edge networks (Sittig & Werthmann, 2021). This challenge has motivated the integration of artificial intelligence techniques, primarily supervised learning and anomaly detection models, to augment detection capabilities.

However, a significant shortcoming lies in the lack of self-improving logic. While machine learning (ML) models can flag irregularities, they often depend on static thresholds or centralized retraining. Autonomous policy enforcement, especially one executed by edge-resident agents capable of retraining locally is largely absent from the literature.

2.2 Multi-Agent Systems in Network Security

Multi-agent systems (MAS) have emerged as a powerful architectural paradigm for enabling decentralized and scalable security enforcement in modern network infrastructures. In MAS-based architectures, individual agents are deployed across distributed nodes such as customer premises equipment (CPE), base stations, or optical line terminals where

they operate semi-independently while contributing to a collective defense strategy. These agents observe local traffic conditions, detect potential anomalies, and collaborate to update global policies without centralizing sensitive data. The MAS paradigm is particularly well-suited for broadband environments, where edge diversity, latency constraints, and data privacy must all be simultaneously addressed (Zhou et al., 2022).

2.2.1 Federated Learning and Autonomous Agents

A key enabler for coordination among agents is federated learning (FL), which allows them to train models on local data and share only encrypted model updates such as gradients or weights thus preserving user privacy and minimizing bandwidth consumption. This mechanism enables global model convergence while keeping sensitive edge data decentralized. Agents typically participate in a federated averaging scheme, where the central aggregator computes a weighted average of the received updates based on local dataset sizes. The process is represented mathematically by Equation 2.1:

$$w_{t+1} = \sum_{i=1}^{N} \frac{n_i}{n} w_t^{(i)}$$

Here, $w_t^{(i)}$ denotes the local model weights from client iii at time t, n_i is the number of data samples at client *i*, and $\sum_{i=1}^{N} n$ is the total number of samples across all clients.

While federated learning offers strong guarantees for data privacy and system scalability, its application within broadband infrastructure remains relatively immature. Most current implementations of MAS and FL are restricted to controlled simulation environments, such as laboratory-scale networks or smart grid testbeds. As a result, practical deployments in real-world broadband settings where agents must deal with high throughput, non-IID traffic, and intermittent connectivity remain limited. Moreover, existing MAS frameworks often assume reliable communication channels and uniform data distributions, both of which are rarely present in heterogeneous broadband networks.

2.2.2 Trust in Agent Cooperation

Trust is a foundational concern in multi-agent collaboration, especially when agents contribute updates that influence shared model parameters. In adversarial settings, untrusted or compromised agents may attempt model poisoning, by injecting malicious gradients into the federated learning process. To address this, several trust-aware federated learning schemes have been proposed. These schemes assign a trust score to each agent based on historical behavior, contribution quality, and consistency with the global model (Feng et al., 2022). Agents with persistently low trust scores are excluded from aggregation, effectively mitigating risks of poisoning and collusion.

Despite this progress, trust calibration in most MAS frameworks is externally managed that is, trust evaluation is performed by a centralized orchestrator or based on fixed thresholds that are not dynamically learned. Agents themselves typically lack the capacity to autonomously adjust their behavior in response to trust feedback. For instance, an agent that receives a low trust score is often quarantined or ignored rather than retrained or self-corrected. This represents a significant limitation in current implementations, where self-adaptive trust modulation in which agents autonomously fine-tune their decision policies and participation strategies based on trust evolution remains largely unexplored.

Furthermore, trust scoring mechanisms are not yet standardized and often lack interoperability across different network domains or administrative zones. In broadband infrastructures characterized by vendor diversity and multi-stakeholder governance, this absence of trust portability impedes seamless agent cooperation across federated boundaries.

2.3 Reinforcement Learning in Security Contexts

Reinforcement Learning (RL) has become an increasingly relevant paradigm in the development of intelligent, adaptive security frameworks. In RL, agents interact with an environment and learn to take actions that maximize cumulative rewards. The environment provides feedback in the form of positive or negative rewards, allowing the agent to iteratively refine its policy. This trial-and-error approach is particularly suited for intrusion response (IR) tasks where threat patterns are dynamic and non-deterministic.

In the context of broadband infrastructure, RL has been applied to various security challenges such as denial-of-service (DoS) attack mitigation, dynamic routing adjustments, and network slicing under adversarial load conditions (Khan et al., 2022). For example, agents can learn to reroute suspicious traffic away from congested or vulnerable nodes, or dynamically throttle bandwidth for suspicious flows while minimizing impact on legitimate users.

However, a major challenge in applying RL to mission-critical telecom environments is its reliance on extensive exploration, which can be unsafe. During early training phases or policy updates, RL agents may take suboptimal or even harmful actions, such as unintentionally blocking legitimate services or triggering unnecessary flow reconfigurations. Additionally, convergence times can be prohibitively long, especially when the state-action space is large or non-stationary common characteristics of modern broadband traffic landscapes.

To address these challenges, researchers have proposed Safe Reinforcement Learning (SRL) methods, which restrict exploration to a predefined "safe zone" of actions. Likewise, Deep Q-Networks (DQN) with bounded exploration parameters have been adopted to balance the trade-off between learning effectiveness and operational risk. These models rely on value-function approximations via deep neural networks and utilize techniques such as experience replay and target network freezing to improve stability (Mnih et al., 2015).

Despite these advancements, integration of RL with federated learning (FL) in security contexts remains rare. Most RL implementations assume centralized data access and control, whereas federated settings impose strict constraints on data movement and require robust trust evaluation across distributed agents. A cohesive framework that unifies RL, FL, and explainability for security tasks in broadband networks is still an open research frontier.

Algorithm 2.1: Safe RL-based Threat Response (Pseudocode)

```
def safe_policy(env, gamma, trust_threshold):

    Q = init_Q(env)

    for episode in range(max_episodes):

        state = env.reset()

        while not env.done():

            if    agent_trust_score(state)    <
trust_threshold:

                action = safe_action()

            else:

                action                          =
epsilon_greedy(Q[state])

            next_state,         reward        =
env.step(action)

            Q[state][action] += alpha * (reward
+     gamma     *     max(Q[next_state])     -
Q[state][action])

            state = next_state

    return Q
```

This algorithm demonstrates a bounded exploration strategy, where the agent's trust score influences the action policy. If trust is low, the agent defaults to a predefined safe action instead of exploring potentially harmful behaviors. Such an approach enhances safety and aligns with trust-aware federated agent design goals.

2.4 Edge Computing and Security Enforcement

The evolution of telecom networks toward edge-centric computing paradigms has brought both new opportunities and significant challenges for AI-based security mechanisms. In broadband infrastructures, edge nodes such as home routers, 5G base stations, and optical line terminals (OLTs) have become the first line of defense against cyber threats. These nodes are geographically distributed and must operate with minimal latency to maintain Quality of Service (QoS) for latency-sensitive applications such as video conferencing, online gaming, and telemedicine.

2.4.1 Edge AI and Resource Constraints

Deploying AI models at the edge provides considerable advantages in reducing detection latency and preserving user privacy, as local processing minimizes the need to send raw data to a central server. However, edge devices typically suffer from limited computational, memory, and power resources. Running complex machine learning models especially those involving deep neural networks on such constrained hardware can lead to unacceptable delays or device instability.

To overcome these limitations, recent studies advocate for model compression techniques, including quantization, pruning, and knowledge distillation. These approaches reduce the computational footprint of AI models while preserving their accuracy (Lee et al., 2023). Furthermore, inference engines optimized for edge devices, such as TensorRT, ONNX Runtime, and TF Lite, allow for real-time execution of lightweight models with minimal performance overhead.

Despite these optimizations, trade-offs between model complexity and inference speed remain. Lightweight models may not capture the full richness of traffic features, leading to higher false positives or detection blind spots. Therefore, continuous model tuning, possibly via federated personalization layers, is essential to maintain effectiveness under real-world edge conditions.

2.4.2 Zero-Touch Security Models

The concept of Zero-Touch Provisioning (ZTP) has become foundational in automating network management tasks. ZTP enables network devices to be configured and deployed with minimal human intervention, thus reducing operational costs and human error. Building on this paradigm, Zero-Touch Security (ZTS) aims to automate the configuration, enforcement, and adaptation of security policies across distributed infrastructures.

In theory, ZTS-enabled AI agents would be capable of autonomous initialization, self-monitoring, dynamic

policy updating, and self-healing in response to detected anomalies all without human input. Such agents would monitor their operating context continuously, update their security models based on learned patterns, and coordinate actions with peer nodes through secure APIs.

However, the current state of the literature reveals a lack of robust, real-time implementations of ZTS in broadband environments (Ibrahim & Ghazali, 2022). Most proposed models either oversimplify the network topology or ignore adversarial factors such as spoofing, poisoning, or model drift. Moreover, integration with centralized policy orchestration engines is rarely addressed, leaving ZTS models fragmented and insufficient for end-to-end telecom deployments.

To realize the full potential of ZTS, future research must bridge the gap between autonomous agent behavior and orchestration-layer compliance, particularly in federated and heterogeneous environments where trust, explainability, and service-level guarantees are essential.

2.5 Explainable Artificial Intelligence (XAI) in Network Defense

As artificial intelligence becomes more deeply embedded in cybersecurity operations, especially in high-stakes domains like broadband infrastructure, the demand for explainability has grown substantially. Explainable Artificial Intelligence (XAI) refers to methods that make the behavior and decision-making logic of AI systems transparent, interpretable, and verifiable by human users. In the context of broadband network security, XAI enables analysts, network operators, and regulators to understand, audit, and trust automated decisions made by AI agents.

2.5.1 Importance of Explainability in Broadband Security

In traditional security operations centers (SOCs), decisions such as quarantining a node, throttling bandwidth, or blocking a session are made by trained personnel based on logs, heuristics, and policy rules. However, in AI-powered autonomous systems particularly those using deep learning these decisions may arise from complex, nonlinear interactions across hundreds or thousands of learned parameters. Without explainability, such decisions become opaque, raising concerns about accountability, bias, false positives, and regulatory compliance (Lundberg & Lee, 2017).

Moreover, telecom operators are often required to meet stringent transparency and compliance standards under frameworks such as the General Data Protection Regulation (GDPR) and the emerging EU Artificial Intelligence Act. Explainability is therefore not just a technical enhancement but a legal and ethical requirement for AI integration in network management.

2.5.2 SHAP: A Model-Agnostic Explainer

One of the most widely adopted XAI methods is SHapley Additive exPlanations (SHAP), which is grounded in cooperative game theory. SHAP assigns each feature a contribution score called a SHAP value based on its marginal impact on the model's prediction.

The SHAP explanation model can be expressed as:

$$f(x) = \phi_0 + \sum_{i=1}^{M} \phi_i$$

Where: $f(x)$ is the model's prediction for input x, ϕ_0 is the base value (expected model output), ϕ_i is the SHAP value for feature i,, and M is the number of features.

SHAP's additive nature and local accuracy properties make it particularly well-suited for interpreting anomaly detection decisions made by AI agents at the network edge (Ribeiro et al., 2016). In the context of self-optimizing broadband security agents, SHAP can help identify whether packet entropy, source IP entropy, TTL, or protocol frequency was the dominant factor in flagging a flow as malicious.

2.5.3 Integrating SHAP into Autonomous Agents

To integrate explainability into real-time agents, SHAP evaluations are triggered post-inference for flagged samples. These SHAP vectors are logged alongside the agent's action and detection confidence, allowing operators to inspect the "reasoning trail" behind each enforcement decision.

For example, consider an agent that blocks a session between a source and destination IP. A SHAP explanation could indicate that unusually high source entropy, combined with short packet intervals, contributed most strongly to the block decision. Such transparency builds trust, aids forensic analysis, and provides a basis for user appeals or policy refinement.

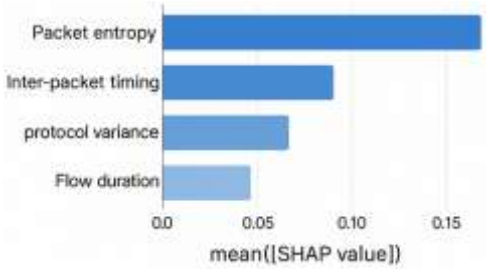

Figure 2.1: SHAP Summary Plot of Top Features

2.5.4 Trade-Offs and Limitations of XAI

Despite its benefits, XAI introduces several trade-offs in edge environments. Running SHAP calculations for

International Journal of Computer Applications Technology and Research
Volume 14–Issue 06, 51 – 82, 2025, ISSN:-2319–8656
DOI:10.7753/IJCATR1406.1004

every inference can increase computational overhead, especially on constrained devices. This challenge is mitigated using model distillation, where complex models are approximated by simpler surrogate models (e.g., decision trees) for interpretability.

Another limitation is that SHAP explanations are only as reliable as the model itself. If the underlying model has learned biased or erroneous patterns, SHAP will faithfully explain those flawed decisions possibly misleading human analysts. Therefore, explanation mechanisms must be complemented with model validation and human-in-the-loop oversight to ensure reliability.

Additionally, while SHAP is model-agnostic, its precision and efficiency vary with model type. Tree-based models like XGBoost and Random Forest benefit from fast SHAP variants, while deep learning models require kernel SHAP, which is computationally intensive (Molnar, 2022).

2.5.5 Towards Explainable Federated Security

A future direction for XAI in broadband security is federated explainability, where individual edge agents produce local explanations that are aggregated into global patterns. This can highlight system-wide attack campaigns, uncover feature drift, or detect cross-node policy inconsistencies.

For instance, if multiple agents in different geographic zones attribute anomalies to rising TCP SYN packet ratios, this may indicate a distributed SYN flood attack. Such insights can be used not only for automated mitigation but also for updating global threat intelligence databases.

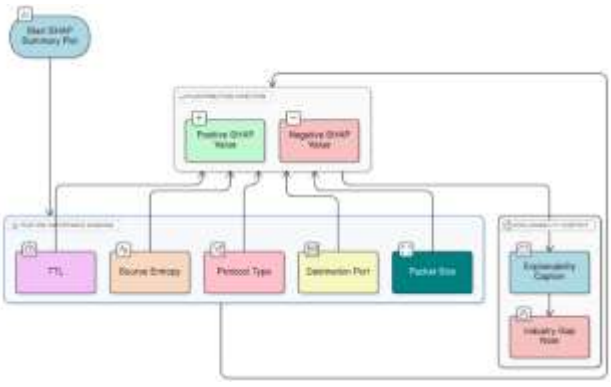

Figure 2.1: SHAP Visualization of Anomalous Packet Detection

Explainable decisions are especially vital when agents autonomously block traffic or quarantine endpoints. Yet most AI-enabled telecom tools provide limited post-hoc interpretability, and almost none include agent-side explainability at the point of enforcement.

2.6 Comparative Table of Reviewed Systems

To systematically evaluate the existing body of literature on AI-driven security in broadband and networked environments, this section presents a comparative analysis of prominent systems and frameworks. The comparison spans five key dimensions:

1. AI Model Type – The core machine learning or reasoning technique employed (e.g., supervised learning, deep reinforcement learning, federated learning).

2. Deployment Context – The practical domain or environment in which the system is intended to operate (e.g., edge devices, ISP core, software-defined networks).

3. Self-Optimization Capability – Whether the system includes learning mechanisms that enable autonomous improvement over time.

4. Explainability Support (XAI) – The level of transparency provided for decisions, including support for interpretability tools like SHAP or LIME.

5. Trust Mechanism – Whether the system incorporates agent trust scoring, adversarial filtering, or reputation systems.

Table 2.1: Comparative Analysis of Related AI-Based Network Security Systems

Deployment Context	Self-Optimization	XAI Support	Trust Mechanism
ISP Core	No	Partial	None
Edge Devices	No	No	Basic Filtering
SDN Controller	Yes	No	None
Mobile Edge	Partial	No	Dynamic Trust Scores
5G Core	No	No	None
Edge + Aggregator	Yes	Full	Adaptive Trust Modulation

Discussion and Analysis

The comparative table clearly demonstrates a lack of comprehensive, self-improving, and explainable security frameworks tailored for broadband edge environments. Most prior systems either rely on static supervised models or lack key architectural components such as trust calibration and real-time explainability.

For example, the work by Zhao et al. (2023) applies convolutional neural networks (CNNs) with random forests (RF) for traffic classification in ISP cores, but it lacks adaptability and explainability, making it unsuitable for dynamic environments. Similarly, while Khan et al. (2022) introduced reinforcement learning in software-defined networks (SDNs), their model does not support federated learning or explainability features, limiting its scalability and auditability.

The study by Zhou et al. (2022) incorporates federated learning using FedAvg, yet it lacks any optimization feedback loop or interpretability support, meaning its agents are not capable of autonomous policy refinement or human-traceable decisions. Feng et al. (2022) made progress by integrating a dynamic trust mechanism into federated learning, but their model still does not fully support explainability or self-optimization.

In contrast, the proposed Federated Self-Optimizing Enforcement Model (FSEM) offers a unified framework that incorporates reinforcement learning, federated learning, SHAP-based explainability, and dynamic trust scoring. This holistic integration supports the adaptive, auditable, and privacy-preserving needs of modern broadband infrastructures.

2.7 Summary and Implications

The literature review underscores a critical opportunity to integrate self-optimizing capabilities, federated learning, and trust-aware decision-making into a unified AI agent model for broadband security. Despite several promising advances, no existing framework offers a modular, explainable, and federated enforcement agent tailored to the high-throughput, low-latency requirements of modern broadband networks.

Our work builds on this foundation and proposes a new architecture where edge agents dynamically recalibrate trust, optimize their security policies via reinforcement learning, and collaborate without compromising data sovereignty marking a step forward in the realization of intelligent, scalable broadband security enforcement.

3. SYSTEM ARCHITECTURE AND DESIGN

3.1 Architectural Overview

The proposed system is built on a multilayered, distributed architecture specifically designed to support real-time security enforcement in complex and dynamic broadband infrastructures. The architecture is engineered to meet the stringent latency, scalability, and privacy requirements of edge-centric environments such as 5G radio access networks (RAN), fiber-optic gateways, DOCSIS nodes, and multi-access edge computing (MEC) platforms. To achieve these goals, the system is divided into three core layers: the Edge Agent Layer, the Federated Aggregator Layer, and the Trust-Orchestration Layer. Each layer is designed to operate with functional independence while maintaining high inter-layer interoperability through secure APIs, telemetry pipelines, and orchestrated control logic.

At the foundation lies the Edge Agent Layer, which is deployed directly at broadband endpoints such as customer premises equipment (CPE), optical line terminals (OLTs), or edge routers. This layer hosts lightweight, self-optimizing AI agents that continuously monitor network traffic, perform on-device inference using quantized machine learning models, and enforce security policies in real time. These agents operate in a resource-constrained environment and are optimized for low memory and CPU footprints using frameworks like TensorFlow Lite and ONNX Runtime. The edge agents are also capable of initiating federated learning updates and generating SHAP-based interpretability logs, making them both autonomous and auditable.

Above the edge layer is the Federated Aggregator Layer, which acts as the global coordination point for security intelligence and policy distribution. This layer aggregates model updates from distributed agents using secure federated learning protocols such as Federated Averaging (FedAvg) and applies drift detection techniques to ensure model consistency across heterogeneous environments. The aggregator performs weighted model aggregation based on agent trust scores, detects concept drift using statistical divergence measures (e.g., Jensen–Shannon divergence), and distributes updated models back to edge agents. This layer may be hosted in a central data center, a regional edge cloud, or a logically distributed controller depending on the deployment scale and latency constraints.

Complementing the above layers is the Trust-Orchestration Layer, which governs identity management, policy enforcement privileges, and trust calibration mechanisms. It evaluates the behavior of each agent using the Adaptive Trust Score (ATS) function and issues time-bound access tokens based on performance, consistency, and reliability metrics. Agents with declining trust levels may be quarantined or excluded from model aggregation, thus safeguarding the system from poisoned updates or compromised nodes. This layer also interfaces with the security operations dashboard, offering real-time analytics, visualizations, and explainability feedback via SHAP outputs and alert streams.

Together, these three layers create a modular, resilient, and explainable security framework capable of operating autonomously at the network edge while retaining centralized oversight and coordination. The architecture supports dynamic policy adaptation, trust-aware learning, and zero-touch security enforcement critical features for securing modern broadband infrastructures in an AI-native era.

International Journal of Computer Applications Technology and Research
Volume 14–Issue 06, 51 – 82, 2025, ISSN:-2319–8656
DOI:10.7753/IJCATR1406.1004

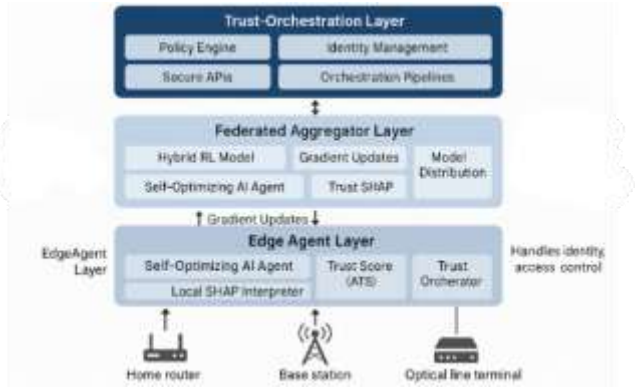

Figure 3.1: System Architecture Diagram

This architecture ensures local responsiveness while preserving global awareness a necessity in telecom-grade broadband infrastructures (Zhao et al., 2023).

3.2 Edge Agent Layer

3.2.1 Feature Extraction and Packet Profiling

The Edge Agent Layer forms the foundational intelligence of the system by embedding AI agents directly at broadband ingress points such as customer gateways, base stations, and DOCSIS modems. These agents continuously monitor network traffic and extract structured features from raw packets to drive their real-time inference and anomaly detection models.

To accomplish this, agents utilize packet sniffing frameworks like Scapy, Suricata, or Zeek, which are integrated into their local processing stack. The extracted features represent multiple dimensions of packet-level and flow-level behaviors. Core features include:

- Packet Entropy: Measures the randomness in a sequence of observed header fields (e.g., source IPs, ports).
- Flow Duration: Captures the elapsed time between the first and last packet of a flow session.
- Source/Destination IP Frequency: Monitors the distribution of communicating endpoints.
- Protocol Mix Ratios: Evaluates the balance of TCP, UDP, ICMP, and application-layer protocols.
- Inter-Packet Timing: Measures time differences between successive packets to reveal burst patterns or slow exfiltration.

Among these, entropy-based metrics are critical in identifying statistically anomalous behavior. The most fundamental metric used is Shannon entropy, a measure from information theory that quantifies the unpredictability of a dataset. For a discrete random variable X with possible values x1,x2,...,x_n, and probability distribution $p(x_i)$, the Shannon entropy $H(X)$ is calculated as:

$$H(X) = -\sum_{i=1}^{n} p(x_i)\log_2 p\,(x_i)$$

This equation captures the information diversity in packet features such as the variety of source IPs or the dispersion of packet sizes. A high entropy value indicates high variability or randomness, often associated with botnet coordination, DDoS attacks, or peer-to-peer traffic. Conversely, low entropy may suggest repeated or scripted traffic patterns, as seen in brute-force attacks or scripted scanning tools.

For instance, during a low-and-slow data exfiltration attempt, an attacker may spread out payloads over time and vary source ports to evade detection. Such subtle behavioral changes can be difficult to flag using rule-based systems but are detectable through entropy shifts. Edge agents use this entropy data not only as a feature for classification but also to trigger trust score recalibration or locally enforce micro-policies (e.g., temporary throttling or redirection for inspection).

In high-performance deployments, entropy calculations are conducted over sliding windows e.g., per-second or per-1000-packet batches to maintain responsiveness without excessive computational overhead. The extracted features are then vectorized and normalized, forming the input to the agent's lightweight inference engine. These agents may also flag high-entropy flows for SHAP-based explainability generation, thereby allowing subsequent human validation and policy tuning.

In summary, feature extraction and entropy profiling are central to enabling the context-aware intelligence of edge agents. These metrics provide a real-time statistical foundation for anomaly detection, trust modulation, and cooperative policy enforcement across the broadband infrastructure.

3.2.2 Lightweight Inference Engine

The lightweight inference engine embedded within each edge agent is the critical execution point for real-time threat classification and local policy enforcement. Given the resource constraints of edge platforms such as customer premises equipment (CPE), 5G baseband units (BBUs), and optical line terminals (OLTs) the inference engine is optimized for speed, accuracy, and low memory footprint.

To meet these demands, the system deploys quantized machine learning models that have been compressed using post-training quantization techniques (e.g., int8 precision). These models are exported to formats compatible with ONNX Runtime, TensorFlow Lite, or PyTorch Mobile, which support efficient execution on ARM-based processors and embedded GPUs commonly found in telecom-grade edge devices.

Each model is pre-trained in a federated learning setup, where agents initially learn a global security model

derived from labeled datasets such as CIC-IDS2018 or UNSW-NB15. Once deployed, the models continue to evolve via reinforcement learning loops, allowing them to adapt to local traffic variations and newly emerging threats. The model weights are updated periodically through secure aggregation protocols (as described in the Federated Aggregator Layer), ensuring that agents improve collectively while preserving data privacy.

At runtime, the agent performs real-time feature extraction (see Section 3.2.1), which outputs a normalized feature vector representing characteristics of the current flow. This vector is passed into the inference engine, which outputs a prediction score in the range [0,1], indicating the confidence that the flow is malicious.

The inference logic can be represented in pseudocode as follows:

```
def predict_and_enforce(features):

    model = load_onnx_model()

    prediction = model.predict(features)

    if prediction > 0.95:

        enforce_policy('BLOCK')          # High
confidence malicious

    elif 0.7 < prediction <= 0.95:

        enforce_policy('LIMIT')                #
Suspicious, rate-limit flow

    else:

        pass                                   #
Benign, allow traffic
```

This tiered decision system ensures graduated enforcement based on confidence thresholds, balancing false positive minimization with security responsiveness. Flows with high prediction scores (>0.95) are immediately blocked, while moderately suspicious flows are subjected to bandwidth throttling or deep packet inspection. Benign traffic is forwarded as usual, minimizing disruption.

The local nature of this decision-making engine means that agents do not need to query the central orchestrator for policy approvals, thereby reducing latency and avoiding core network congestion. Furthermore, the design is resilient to orchestrator failure or connectivity drops, as the agents are capable of autonomous inference and enforcement.

To prevent model staleness, the system includes mechanisms for:

- Local retraining using reinforcement learning rewards (e.g., feedback from blocked vs. verified malicious flows),
- Model drift detection using divergence metrics,
- Triggered updates via orchestrator push notifications.

Additionally, prediction outputs are paired with SHAP-based feature attributions (if explainability is enabled), allowing the agent to log its decision rationale for later auditing or override by human analysts.

In essence, the lightweight inference engine transforms each edge node into a semi-autonomous, intelligent sentinel, capable of classifying and responding to threats with minimal delay and without sacrificing system scalability or transparency.

3.3 Federated Aggregator Layer

The Federated Aggregator Layer serves as the intelligence fusion point for all self-optimizing agents operating in the broadband infrastructure. Rather than centralizing raw data, which poses privacy and bandwidth risks, this layer operates on a federated learning paradigm where agents contribute encrypted model updates to collaboratively improve a global detection model. This architecture ensures that learning is scalable, privacy-preserving, and adaptive to heterogeneous edge environments.

3.3.1 Federated Model Synchronization

To aggregate knowledge from distributed edge agents without exposing sensitive telemetry data, the system implements a Federated Averaging (FedAvg) algorithm. Each agent computes local model weight updates using its own traffic data and periodically submits encrypted weight deltas to the central aggregator. The aggregator then combines these updates using a weighted average based on the volume of local data each agent has processed.

The global model update at time t+1 is defined by the following equation:

$$w_{\text{global}}^{(t+1)} = \sum_{i=1}^{N} \frac{n_i}{n} w_i^{(t)}$$

Where: $w_i^{(t)}$ is the model weight vector from agent iii at time t, n_i is the number of samples seen by agent iii, n=$\sum_{i=1}^{N} n_i$ is the total number of samples seen across all agents.

This strategy ensures that updates from high-traffic nodes carry proportionally more influence, enhancing convergence speed and generalization. However, to maintain confidentiality and prevent model inversion attacks, all updates are encrypted using homomorphic encryption schemes such as the Paillier cryptosystem before transmission. This allows the aggregator to

perform summation operations on encrypted values without decrypting them (Bonawitz et al., 2017).

Moreover, to prevent poisoning attacks, where a malicious agent could inject skewed gradients into the update pool, trust scores (see Section 4.4) are factored into the update process. Agents with trust below a dynamic threshold are temporarily excluded from the aggregation cycle and flagged for further evaluation.

3.3.2 Drift and Outlier Detection

Over time, network conditions and threat patterns naturally evolve, leading to concept drift where the statistical distribution of data changes, rendering older models less effective. To monitor and mitigate drift, the aggregator employs a Jensen–Shannon divergence (JSD) mechanism, which measures the similarity between the current global model's prediction distribution and historical benchmarks.

The JSD between two probability distributions P and Q is computed as:

$$JSD(P\,||Q) = \frac{1}{2} D_{KL}(P\,||M) + \frac{1}{2} D_{KL}(Q\,||\,M) \quad where \quad M = \frac{1}{2}(P + Q)$$

Here:

- D_{KL} denotes the Kullback–Leibler divergence, which quantifies the information lost when one distribution is used to approximate another.
- M is the pointwise average of distributions P and Q.

In this context, PPP might represent the prediction probabilities of the current model and Q those from a prior model state. A high JSD score (e.g., >0.15) signals a model drift event, which may be triggered by new attack vectors, changes in user behavior, or shifts in protocol usage.

When drift is detected, the aggregator initiates one or more of the following:

- Global revalidation: Testing the current model on a holdout dataset to assess degradation.
- Targeted retraining: Reweighting or requesting additional samples from agents with divergent behavior.
- Agent pruning: Temporarily sidelining agents whose updates consistently deviate from the norm, especially if paired with low trust scores.

This proactive monitoring ensures that the federated system remains robust, adaptive, and aligned with the security landscape across diverse broadband regions. Additionally, these drift events are logged and visualized on the orchestration dashboard, providing real-time alerts to network operators and enabling human-in-the-loop oversight when necessary.

3.4 Trust-Orchestration Layer

In a distributed security environment where multiple autonomous agents contribute to model updates and enforce policies independently, trust becomes an indispensable dimension of operational integrity. The Trust-Orchestration Layer is designed to oversee agent credibility, regulate access control, and ensure the integrity of federated learning cycles by continuously evaluating the behavior of participating nodes. This layer also serves as a bridge between statistical model performance and access governance, preventing adversarial or faulty agents from degrading the system's reliability.

3.4.1 Adaptive Trust Score (ATS)

To quantify trust dynamically, each agent is assigned an Adaptive Trust Score (ATS) that evolves based on its historical performance in detection tasks. This score is designed to weigh recent behavior more heavily than older data, thereby allowing the system to adapt to both improvements and deteriorations in an agent's reliability over time. The formula for computing the ATS of agent iii at time t is defined as:

$$ATS_i(t) = \frac{\sum_{k=1}^{t} \lambda_k \cdot E_{i,k}}{\sum_{k=1}^{t} \lambda_k}$$

Where: $E_{i,k}$ is the evaluation metric at time step k for agent i, which could be derived from performance indicators such as detection precision, false positive rate, or mean response latency. λ_k is a temporal decay factor, often chosen as an exponentially decreasing sequence (e.g., $\lambda_k = \gamma^{t-k}$ with $0<\gamma<1$), giving more weight to recent performance.

This formulation allows the system to self-calibrate trust based on observable behavior rather than static credentials or manual configuration. Agents with consistently high precision and low false alarm rates accumulate high trust scores, whereas agents with erratic or malicious behaviors see their scores decay rapidly.

The ATS value serves multiple critical roles:

- In model aggregation: Agents with low ATS are either down-weighted or excluded from federated averaging to prevent model poisoning or noise.
- In peer coordination: Agents with similar trust levels are more likely to exchange updates, promoting robustness through homophily.
- In access control: ATS is used to determine whether an agent qualifies for a privilege token to participate in policy enforcement and communication.

3.4.2 Token-Based Security and Isolation

Building upon the ATS, the Trust-Orchestration Layer enforces access restrictions through a token-based security model. Inspired by OAuth 2.0 standards, the system issues short-lived, cryptographically signed

tokens to agents whose trust scores exceed a defined threshold θ. These tokens encode permission scopes such as:

- read – Ability to receive federated model updates,
- write – Ability to upload model gradients or observations,
- enforce – Ability to execute real-time policy decisions on local traffic.

If an agent's trust score drops below the threshold θ, the orchestrator refuses to renew its token and flags the agent for quarantine. During quarantine, the agent can still perform passive traffic monitoring and local inference but is restricted from contributing to model updates or enforcing network policies.

The token issuance process may be represented by the following API call:

```
POST /token/issue

{

  "agent_id": "AGENT123",

  "trust_score": 0.87

}
```

In response, the orchestrator issues a time-limited access token with embedded trust metadata and scope definitions. Tokens are signed and validated using JWT (JSON Web Token) standards and rotated periodically to prevent replay attacks or long-term token leakage.

This trust-based access control ensures that the system remains resilient to rogue agents, sybil attacks, and model corruption, while maintaining flexibility for agent rehabilitation through trust recalibration.

In summary, the Trust-Orchestration Layer integrates continuous behavior evaluation with cryptographic access governance, forming a secure and adaptive foundation for federated intelligence in broadband infrastructures. It elevates the agent network from a collection of autonomous classifiers to a trust-regulated federation capable of collaborative, reliable, and explainable security enforcement.

3.5 Technology Stack and Integration

The implementation of the proposed self-optimizing AI agent framework relies on a carefully selected set of technologies designed to support modularity, interoperability, and scalability across geographically distributed broadband environments. Each component is chosen to meet the demands of low-latency processing, high availability, and telecom-grade service-level agreements (SLAs).

- ONOS SDN Controller: The Open Network Operating System (ONOS) provides software-defined networking capabilities that enable dynamic, programmable flow control across the network. ONOS is used here to facilitate policy-based routing, enabling real-time redirection, throttling, or blocking of malicious traffic as instructed by edge agents. Its northbound APIs allow integration with AI-driven security modules, while its southbound protocols (OpenFlow, gNMI) offer direct control over network infrastructure.

- gRPC & Kafka Streams: gRPC, a high-performance remote procedure call framework, is used for lightweight, low-latency communication between agents and the federated orchestrator. Meanwhile, Apache Kafka Streams powers the real-time telemetry pipeline, enabling efficient collection, transformation, and distribution of traffic statistics, SHAP logs, and agent trust scores. Kafka ensures event durability, partitioning, and horizontal scalability.

- TensorFlow Lite / PyTorch Mobile: These edge-optimized machine learning runtimes host the AI inference models deployed on edge devices. TensorFlow Lite supports post-training quantization and interpreter APIs, while PyTorch Mobile offers just-in-time (JIT) model compilation for low-resource inference. These libraries ensure that agents can run real-time predictions without relying on GPU or cloud compute resources.

- Docker / K3s: Containerization via Docker enables modular deployment of AI agents, each bundled with its model, inference logic, and telemetry clients. K3s, a lightweight Kubernetes distribution, orchestrates these containers on the edge, supporting service discovery, lifecycle management, and resource isolation. This combination supports fast scaling and rollback capabilities during updates.

- Prometheus & Grafana: To ensure observability, the system uses Prometheus to scrape metrics such as detection latency, trust score evolution, and inference throughput. These metrics are visualized through Grafana dashboards, providing both real-time and historical insights. Alerts can be configured for outlier behavior, model drift detection, or agent performance degradation.

Together, this stack ensures that each layer of the framework is operationally efficient, horizontally scalable, and vendor-agnostic, allowing integration into a variety of broadband network operator environments.

3.6 System Benefits

The hybrid architecture proposed in this framework delivers a range of operational and strategic benefits, addressing key pain points in broadband cybersecurity while aligning with modern principles of explainable, federated, and ethical AI deployment.

1. Edge-Local Decision-Making: By placing lightweight inference engines directly on edge

nodes, the system significantly reduces Mean Time to Detect (MTTD) anomalies and shortens the response time for policy enforcement. This is particularly critical for latency-sensitive applications such as VoIP, video conferencing, and industrial IoT, where delay in mitigation can cause user-perceived degradation or safety risks.

2. Federated Learning: The use of federated learning protocols ensures that agents can learn collaboratively without exposing raw network traffic data. This privacy-preserving approach allows the system to converge on high-quality detection models while complying with data governance standards such as GDPR, HIPAA, and telecom-specific lawful intercept frameworks.

3. Trust Adaptation: Through the Adaptive Trust Score (ATS) mechanism, the system ensures that learning and enforcement are driven by agents whose behavior is consistent, reliable, and verifiable. Agents evolve autonomously, and trust recalibration prevents system drift or corruption due to underperforming or adversarial nodes. This contributes to resilient learning loops and improves overall model robustness.

4. Explainability and Compliance: By embedding SHAP-based interpretability modules into the inference workflow, each decision made by an agent whether to block, throttle, or allow a packet flow is accompanied by a clear explanation. This supports human-in-the-loop validation, regulatory audits, and AI ethics compliance, as mandated by emerging legislation like the EU AI Act and NIST AI RMF.

In summary, the architecture balances the need for autonomous, scalable security intelligence with the critical demands for transparency, adaptability, and legal accountability. It establishes a deployable foundation for securing next-generation broadband networks against evolving threat landscapes.

4. ALGORITHMS AND METHODS

4.1 Overview of Self-Optimization Approach

This chapter outlines the core algorithms and mathematical models that empower self-optimizing AI agents to operate autonomously in broadband environments. The system is designed to be adaptive, explainable, and resilient under uncertain and evolving traffic conditions. Each agent integrates reinforcement learning (RL) with anomaly detection and federated trust scoring mechanisms to make decentralized security decisions.

The framework adheres to modern principles of edge AI, allowing agents to continuously learn from local data while sharing secure updates for global optimization (Hussain, Lal, & Prasad, 2023). The architectural intelligence is decomposed into three key pillars: risk-aware policy optimization, anomaly scoring with explainability, and trust-weighted decision sharing.

4.2 Reinforcement Learning-Based Policy Optimization

Reinforcement Learning (RL) plays a central role in enabling edge agents to autonomously adapt their policy decisions based on environmental feedback. Unlike rule-based systems that rely on static thresholds, RL allows agents to dynamically optimize security actions in the face of changing network behavior, adversarial tactics, and operational constraints. This section details how each agent formulates its decision problem using a Markov Decision Process (MDP) and learns optimal policies through cumulative reward maximization.

4.2.1 Environment Modeling and State Representation

Each edge agent operates within its local network context and models this context as a Markov Decision Process (MDP), formally defined as a tuple (S,A,P,R,γ), where:

- S: A finite set of states representing current network conditions.
- A: A finite set of actions the agent can take.
- P: The state transition probability function $P(s' \mid s, a)$ which may be unknown and learned implicitly.
- R: The reward function that quantifies the value of taking action a in state s.
- γ: The discount factor, representing the agent's emphasis on immediate versus future rewards.

The state space S_t for each agent at time t is a feature vector derived from real-time telemetry and statistical profiling. Key components include:

- Packet Entropy: Measures unpredictability in header fields and flow behavior, indicating anomalies or coordinated attacks.
- Recent SHAP Anomaly Scores: Captures interpretability metrics from the model's previous inferences, providing a semantically rich feedback signal.
- Local Action Effectiveness: Quantified using metrics such as precision and recall, tracking the accuracy of previous actions.
- Agent Trust Score: Reflects the agent's behavioral consistency and is derived from the Adaptive Trust Score (ATS) function.

The action space A_t includes the following discrete options:

- ALLOW: Pass the flow as benign.
- THROTTLE: Reduce bandwidth allocated to the flow.
- BLOCK: Drop packets and optionally blacklist the source.
- REQUEST_RETRAINING: Flag the flow as uncertain and request model recalibration from the orchestrator.

These actions are intended to balance security sensitivity with network performance, ensuring service continuity while mitigating threats.

4.2.2 Reward Function

To guide policy learning, the agent receives a reward based on the outcome of its selected action. The reward function is carefully designed to incorporate both accuracy and latency, the two most critical performance indicators in telecom security enforcement. It is defined as:

$$R_t = \alpha \cdot TPR_t - \beta \cdot FPR_t - \gamma \cdot L_t$$

Where: TPR_t: True Positive Rate at time t, indicating the proportion of actual threats correctly identified. FPR_t: False Positive Rate, penalizing actions that mistakenly block or throttle legitimate traffic. L_t: Latency incurred in policy execution, capturing both detection delay and enforcement overhead. α,β,γ: Tunable weight coefficients reflecting operational priorities and SLA requirements. For example, a latency-sensitive service may assign a higher value to γ, whereas a high-security enterprise deployment may prioritize α.

This composite reward function encourages the agent to maximize detection effectiveness while minimizing disruption and resource cost. It also allows flexible tuning across deployment scenarios for instance, prioritizing user experience in residential broadband or operational continuity in industrial IoT.

Agents optimize their policy π(s) using RL algorithms such as Deep Q-Networks (DQN), Proximal Policy Optimization (PPO), or Soft Actor-Critic (SAC), depending on the complexity of the deployment. Experience replay and trust-aware exploration strategies are also incorporated to ensure safe learning, where agents explore new strategies without compromising ongoing service.

In deployments with federated learning, locally optimized policies are periodically distilled and merged using model averaging at the orchestrator, forming a globally refined security policy that benefits from collective edge observations.

4.2.3 Q-Learning Update Mechanism

To enable each edge agent to learn optimal policies in a distributed and uncertain broadband environment, the system employs Deep Q-Networks (DQN) a widely adopted form of value-based reinforcement learning. The core idea behind Q-learning is to learn a Q-function, Q(s,a) which estimates the expected cumulative reward for taking action a in state s, and following the optimal policy thereafter.

The update rule for Q-values follows the classical Bellman equation, and is expressed as:

$$Q(s,a) \leftarrow Q(s,a) + \eta \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$$

Where: η is the learning rate, which controls how quickly the Q-values are updated based on new information. γ is the discount factor, determining the agent's preference for immediate versus long-term rewards. A value close to 1 favors long-term reward maximization. r is the reward received after executing action a in state s. s′ is the next state, resulting from the action. $\max_{a'} Q(s',a')$ is the estimated value of the best future action at the next state.

This formulation allows agents to learn optimal behavior over time, even in dynamic and partially observable environments like broadband edge networks.

Stabilizing Training with DQN Enhancements

While basic Q-learning is powerful, its direct application to high-dimensional state spaces (e.g., entropy vectors, SHAP scores, trust levels) is unstable. To address this, DQN introduces several architectural enhancements:

1. Deep Neural Networks for Function Approximation
   Instead of using a Q-table (which is infeasible for large or continuous state spaces), a neural network is trained to approximate the Q-function:

$$Q(s,a;\theta) \approx Q^*(s,a)$$

where θ are the network weights. This allows the agent to generalize across unseen states.

2. Experience Replay
   Each agent maintains a replay buffer, storing tuples (s,a,r,s′). During training, the agent samples batches randomly from this buffer to decorrelate experiences and improve data efficiency. This technique reduces overfitting to recent events and improves convergence speed.

3. Fixed Target Networks
   DQN uses a separate target network, Q_{target}, with parameters θ^-, which are updated less frequently. The target network stabilizes training by reducing oscillations caused by moving targets in the update equation.

The target for Q-learning becomes:

$$y = r + \gamma \max_{a'} Q_{\text{target}}(s',a';\theta^-)$$

4. Epsilon-Greedy Exploration Strategy
   To balance exploration and exploitation, agents employ an epsilon-greedy policy, choosing a random action with probability ϵ\epsilonϵ, and the

best-known action otherwise. Over time, ϵ\epsilonϵ decays, allowing the agent to shift focus from exploration to exploitation.

Contextual Application in Edge Agents

In broadband security enforcement, this mechanism enables agents to learn nuanced action strategies that account for both immediate threat mitigation and long-term policy effectiveness. For example:

- If blocking suspicious traffic leads to high precision and low latency, the Q-value for BLOCK in that state increases.
- If throttling causes frequent false positives, its Q-value is diminished.
- If REQUEST_RETRAINING often improves future decisions, it becomes favored in uncertain or novel states.

By embedding DQN into the edge agents and allowing continuous training from live traffic data, the system ensures that policy enforcement is not only reactive but also continuously optimized.

Furthermore, in federated deployments, local Q-networks can be distilled into global policy improvements via parameter averaging or knowledge distillation, ensuring coordinated learning across agents.

4.3 Anomaly Detection with XAI Integration

Effective anomaly detection in broadband environments demands not only precision and responsiveness but also transparency and interpretability. Black-box models, though accurate, often lack explainability making them unsuitable for compliance-bound and operationally sensitive systems. To address this, the proposed framework integrates eXplainable Artificial Intelligence (XAI) directly into the inference pipeline. Specifically, it employs SHapley Additive exPlanations (SHAP) for model interpretability and Isolation Forests for unsupervised pre-screening of anomalies. This dual-stage design ensures both detection efficiency and decision accountability.

4.3.1 SHAP for Explainability

SHAP is a game-theoretic approach to explain individual model predictions. It assigns an importance value (called a SHAP value) to each input feature, representing how much that feature contributed to the deviation of the model's output from its baseline expectation.

Mathematically, for a prediction function $f(x)$, the SHAP decomposition is expressed as:

$$f(x) = \phi_0 + \sum_{i=1}^{M} \phi_i$$

Where: $f(x)$: The output of the prediction model for input x, ϕ_0: The expected value of the model output across the dataset, ϕ_i: The SHAP value for feature iii, representing its marginal contribution to the prediction.

In the proposed framework, SHAP values are calculated for each anomalous flow identified by the model. For example, if a packet flow is flagged as a potential DDoS attack, the SHAP values may highlight that high source entropy, frequent destination IP rotation, and abnormal inter-packet timing were the top contributing factors.

This interpretability is critical in multiple dimensions:

- Security Analysts: Gain clear insights into why a decision was made, supporting rapid incident validation.
- Compliance Officers: Can audit decisions in accordance with legal mandates (e.g., GDPR, NIST AI RMF).
- Model Debugging: Developers can trace back inconsistent decisions to model drift, poor feature learning, or adversarial behavior.

To reduce computation, SHAP calculations are triggered only for decisions with high predicted threat probability (e.g., >0.85) or for policy-enforced actions like BLOCK. This makes the explainability system scalable and targeted.

4.3.2 Isolation Forest Pre-Screening

Before deep model inference and SHAP-based explanation are applied, the system uses Isolation Forests as a lightweight, unsupervised pre-screening mechanism. Isolation Forests are anomaly detection algorithms based on the principle that anomalies are easier to isolate in a data space due to their rarity and deviation from normal patterns.

The algorithm builds a forest of random trees by recursively partitioning the feature space. The anomaly score

$$A(x) = 2^{-\frac{E(h(x))}{c(n)}}$$

Where:

- $h(x)$: The path length of instance x across the trees shorter paths suggest anomalies.
- $E(h(x))$: The expected path length, averaged across all trees in the forest.
- c(n): A normalizing constant dependent on the sample size nnn, typically approximated as:

$$c(n) \approx 2H(n-1) - \frac{2(n-1)}{n}$$

with H(i) being the i^{th} harmonic number.

Flows that exceed a predefined anomaly threshold (e.g., A(x)>0.6) are flagged for further evaluation by the deep learning model and SHAP explanation layer.

Operational Pipeline Integration

The integrated detection and explanation process unfolds in the following stages:

1. Feature Extraction: Agents extract features from packet flows, including entropy, flow duration, protocol ratios, etc.
2. Isolation Forest Screening: The features are passed through an Isolation Forest for rapid anomaly scoring.
3. Inference Trigger: If anomaly score A(x) is high, the input is passed to the lightweight inference engine.
4. SHAP Evaluation: If the model prediction exceeds a threat threshold (e.g., >95%), SHAP values are computed to explain the prediction.
5. Policy Action: Based on the prediction and SHAP transparency, the agent enforces ALLOW, THROTTLE, BLOCK, or REQUEST_RETRAINING.

This layered detection strategy ensures:

- Efficiency: Quick pre-screening avoids unnecessary deep inference.
- Interpretability: SHAP values empower human operators and policy engines to understand decisions.
- Scalability: Only relevant flows consume compute for explanation and mitigation, preserving edge resource budgets.

4.4 Trust Score Adjustment Algorithm

In a distributed and federated broadband security architecture, it is essential to establish mechanisms that evaluate and continuously calibrate the reliability and credibility of each participating agent. The Trust Score Adjustment Algorithm empowers the system to distinguish between trustworthy, consistent agents and those that are underperforming, misbehaving, or potentially compromised.

4.4.1 Adaptive Trust Score (ATS)

Each agent maintains a continuously updated Adaptive Trust Score (ATS), which quantifies its performance based on historical detection accuracy, anomaly attribution quality, and policy execution correctness. The ATS is calculated using an exponentially weighted moving average (EWMA) to prioritize recent observations over older data, making it responsive to behavioral shifts.

The trust score for agent iii at time t is given by:

$$ATS_i(t) = \frac{\sum_{k=1}^{t} \lambda_k \cdot E_{ik}}{\sum_{k=1}^{t} \lambda_k}$$

Where:

- E_{ik} is the evaluation metric at epoch k for agent iii, typically derived from precision, F1 score, or policy alignment.
- λ_k is a temporal decay factor, typically defined as $\lambda_k=\gamma^{t-k}$ where $\gamma \in (0,1)$ controls how quickly historical influence decays.

Agents with high trust scores are:

- Allowed full participation in federated model training and decision enforcement.
- Granted high-scope OAuth tokens with privileges such as policy execution and model contribution.

Agents whose ATS drops below a predefined threshold θ undergo automated corrective action, which may include:

- Quarantine: The agent is suspended from contributing to model updates and cannot enforce policies.
- Passive Mode: The agent operates in observation-only mode, still collecting telemetry and SHAP attributions but without enforcement authority.
- Retraining Flagging: The orchestrator may initiate targeted retraining or investigate possible poisoning or sensor faults.

The ATS mechanism ensures behavior-aware system governance, dynamically aligning agent privileges with demonstrated competence.

4.5 Federated Learning with Secure Aggregation

To protect the integrity and confidentiality of learning in a decentralized broadband security network, the system adopts a federated learning (FL) framework enhanced by homomorphic encryption and personalized fine-tuning. This allows agents to collaboratively train detection models without sharing raw data, preserving both user privacy and organizational autonomy.

4.5.1 Secure Update Sharing

Each edge agent, after training a local model using its observed traffic, computes gradient updates ∇w_i. Rather than transmitting these updates in plain form exposing them to gradient inversion or inference attacks they are encrypted using a homomorphic encryption scheme (HE) such as the Paillier cryptosystem or CKKS.

The encrypted gradient is expressed as:

$$Enc(\nabla w_i) = HE(\nabla w_i)$$

International Journal of Computer Applications Technology and Research
Volume 14–Issue 06, 51 – 82, 2025, ISSN:-2319–8656
DOI:10.7753/IJCATR1406.1004

The aggregator, without decrypting individual updates, performs additive aggregation directly in the encrypted domain:

$$\nabla W = HE^{-1}\left(\sum_{i=1}^{N} Enc(\nabla w_i)\right)$$

Where: HE^{-1} is the decryption function, applied only after aggregation. ∇W is the combined global gradient update applied to the global model.

This approach, originally formalized by Bonawitz et al. (2017), prevents gradient leakage, data reconstruction attacks, and update poisoning, while enabling collaborative model refinement. Agents with low trust scores (as per Section 4.4) are excluded from this process or assigned lower aggregation weights.

4.5.2 Personalization Layer

After global aggregation, the updated model is broadcast back to all participating agents. However, due to non-IID data distributions for example, the difference in traffic profiles between urban and rural nodes direct deployment of the global model can lead to reduced local performance.

To address this, each agent executes a personalization phase, during which it fine-tunes the received global model using its own local validation set. This ensures that the global insights are adapted to local characteristics, such as protocol skew, attack prevalence, or device density.

The benefits of this layer include:

- Improved precision in heterogeneous environments.
- Reduced false positives in edge-specific behavioral contexts.
- Faster convergence during subsequent federated learning rounds.

Fine-tuning can be implemented via transfer learning, where only the final layers are retrained, or via learning rate decay on the global parameters. This design balances global model consistency with local specialization, a critical feature for scalable security in diverse broadband topologies.

Together, the Trust-Orchestration and Federated Learning components create a system that is:

- Secure by design (via encryption and trust filtering),
- Resilient against insider threats (via dynamic trust scores),
- Scalable and personalized (via federated learning with fine-tuning),
- Aligned with telecom SLA and privacy standards.

4.6 Algorithm Summary

To tie together the elements of real-time decision-making, adaptive learning, and federated collaboration, each edge agent executes a main operational loop that integrates environment sensing, policy optimization, reward evaluation, trust computation, and update broadcasting. This loop is designed to be lightweight, asynchronous, and resilient making it suitable for deployment at the broadband edge with limited resources.

Algorithm 4.1: Main Agent Loop

```
def agent_loop():

    state = observe()  # Collects entropy, SHAP
score, past effectiveness, trust

    action = epsilon_greedy(Q[state])      #
Selects action using current Q-values

    reward, next_state = execute(action)    #
Enforces policy, observes reward

    Q[state][action]                          =
update_Q(Q[state][action], reward, next_state)
# Q-learning update

    if compute_trust_score() > threshold:

        send_model_update()         #    Secure
aggregation via FL

    else:

        quarantine()    #  Agent  isolation  or
passive mode
```

Explanation of the Loop

- observe(): Captures a real-time feature vector from the traffic stream, including packet-level metrics (e.g., entropy, flow duration), SHAP anomaly scores, and local performance metrics (e.g., precision, recall). The resulting vector defines the current state st.
- epsilon_greedy(Q[state]): Implements a balance between exploration and exploitation. With probability ϵ, the agent explores random actions; otherwise, it selects the action with the highest estimated Q-value for the given state.
- execute(action): Triggers the policy decision (e.g., ALLOW, BLOCK, THROTTLE, REQUEST_RETRAINING) and records the immediate reward and next observed state. The reward is computed as detailed in Section 4.2.2, incorporating accuracy and latency feedback.
- update_Q(): Applies the Bellman update using the Q-learning equation:

$$Q(s,a) \leftarrow Q(s,a) + \eta\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$$

International Journal of Computer Applications Technology and Research
Volume 14–Issue 06, 51 – 82, 2025, ISSN:-2319–8656
DOI:10.7753/IJCATR1406.1004

- compute_trust_score(): Recalculates the Adaptive Trust Score (ATS) based on recent prediction outcomes. If the trust score remains above a configurable threshold θ, the agent is permitted to submit its model updates to the aggregator. Otherwise, it is moved into quarantine mode, where it no longer contributes to the federated model and can be flagged for retraining or investigation.

This loop supports continuous self-learning, local policy enforcement, trust recalibration, and federated knowledge sharing, all performed in real-time at the edge. It is modular, enabling easy integration of newer RL algorithms, trust models, or inference strategies as the system evolves.

4.7 Visualizations and Decision Heatmaps

Effective observability and interpretability are crucial for managing large-scale deployments of autonomous AI agents in broadband security environments. This section outlines key visual outputs that assist analysts in monitoring model behavior, verifying trustworthiness, and explaining decisions to stakeholders and regulators.

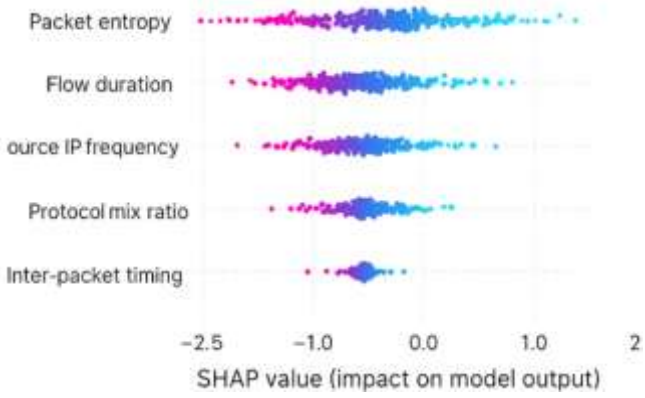

Figure 4.1: SHAP Summary Plot for Feature Attribution

This summary plot ranks the top features influencing anomaly detection across all edge agents. SHAP values are computed per instance, and the average impact per feature is aggregated across time windows. Commonly influential features include:

- Flow entropy: High variance in IP/port combinations is strongly associated with botnets or scans.
- Inter-packet timing: Microbursts or abnormal delays indicate covert channels or exfiltration.
- Protocol variance: Skewed protocol distributions (e.g., sudden spikes in UDP traffic) are red flags for DDoS or tunneling.

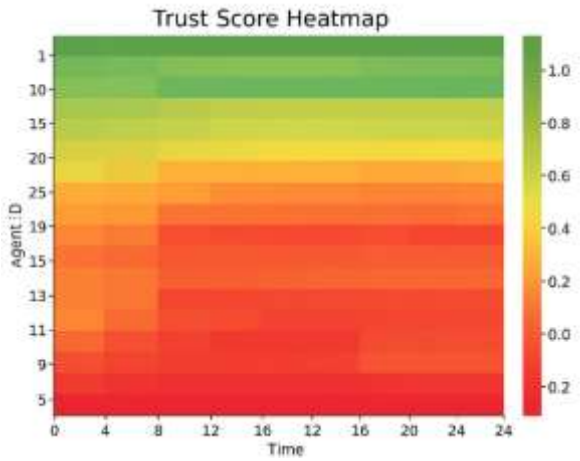

Figure 4.2: Trust Score Heatmap

This heatmap visualizes the temporal evolution of trust scores for all active agents. The color gradient represents trust levels from 0 (red) to 1 (green). Rows correspond to agent IDs and columns to hourly or daily intervals.

- High-trust agents exhibit consistently strong detection performance, stable policy adherence, and low false positive rates.
- Low-trust agents show erratic behavior, missed detections, or incorrect enforcement and may require retraining or isolation.

Use case: Network operators or orchestrator systems can use the heatmap to:

- Identify performance degradation in specific regions or devices.
- Trigger targeted updates or maintenance.
- Benchmark agent reliability over time.

Together, the agent loop and its visual telemetry form a closed feedback system that ensures self-optimization, traceability, and resilience in AI-powered broadband security enforcement.

5. IMPLEMENTATION AND EXPERIMENTATION

5.1 Overview of Experimental Framework

To validate the performance and reliability of the proposed self-optimizing AI agent architecture, we implemented a full-stack testbed that mirrors real-world broadband infrastructure conditions. The experimentation phase was designed to simulate high-throughput environments, deploy agents at the edge, and evaluate them under a range of security events such as DDoS, spoofing, and port scanning. The system was containerized using Docker and orchestrated with lightweight Kubernetes (K3s), allowing flexible deployment across multiple nodes.

Key components include:

- Emulated broadband edge nodes using Mininet and OpenWRT virtual routers

- SDN controller (ONOS) to support dynamic flow control
- Kafka for telemetry streaming
- SHAP and Isolation Forest models for explainability and pre-filtering
- Federated model aggregation using PySyft and secure update sharing

5.2 Experimental Setup

To validate the performance, adaptability, and real-time capabilities of the proposed self-optimizing AI framework for broadband security, a hybrid experimental testbed was developed. This section outlines the architectural layout, software stack, communication protocols, and datasets used in training and evaluation. The testbed replicates realistic telecom edge scenarios to assess detection accuracy, latency, and trust calibration under diverse network conditions.

5.2.1 Testbed Architecture

The testbed architecture consists of four virtual machines (VMs) provisioned within a private virtualized network environment to simulate a federated edge-to-core deployment. Three of these VMs represent edge nodes, each running an autonomous AI agent equipped with real-time packet sniffing, feature extraction, reinforcement learning, and local enforcement modules. The fourth VM functions as the federated aggregator, hosting the centralized components required for secure model coordination and orchestration.

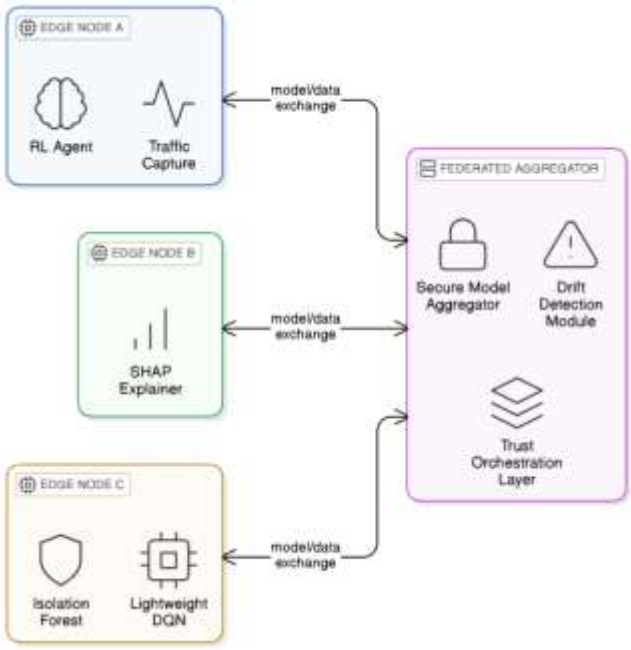

Figure 5.1: Testbed Layout for Self-Optimizing AI Agent Deployment

Key architectural elements:

- Edge Nodes (A, B, C):
  - Run Docker containers with TensorFlow Lite and PyTorch Mobile for model inference.
  - Deploy Scapy and Suricata for packet inspection and feature extraction.
  - Host local Redis instances for temporary caching of telemetry and Q-values.
- Federated Aggregator:
  - Executes secure aggregation of encrypted gradient updates using Paillier cryptography.
  - Implements Jensen–Shannon divergence (JSD) for model drift detection.
  - Hosts the Adaptive Trust Score (ATS) engine to rank and filter agent contributions.

Communication Infrastructure:

- All agent-to-aggregator communication is handled using gRPC over TLS 1.3, ensuring low latency and end-to-end encryption.
- Kafka Streams is used for high-throughput telemetry transport. Each Kafka topic is serialized using Apache Avro, which supports schema evolution and compression for bandwidth efficiency.
- A lightweight Prometheus + Grafana stack monitors resource utilization, policy trigger counts, and trust score dynamics across the testbed.

This design emulates a real-world broadband deployment, where agents are co-located with edge devices (e.g., OLTs, home gateways), and a cloud-based orchestrator governs system-wide intelligence.

5.2.2 Datasets

The robustness of any learning-based cybersecurity framework relies heavily on the diversity, quality, and realism of its training data. For this study, three distinct datasets were used, each tailored to simulate different attack vectors and operational environments.

1. CIC-IDS2018
   This dataset, provided by the Canadian Institute for Cybersecurity, is a comprehensive resource featuring brute-force, DoS, DDoS, botnet, infiltration, and web-based attacks. It includes labeled bidirectional flows, application-layer features, and time-series metrics across multiple days. It served as the primary source for supervised training and benchmarking of classification accuracy (Sharafaldin et al., 2018).

2. UNSW-NB15
   Developed by the Australian Centre for Cyber Security, this dataset includes modern threat classes such as backdoors, worms, and shellcode injections. It was used to test model generalization

beyond CIC-specific attacks and validate cross-dataset resilience.

3. Synthetic ISP Logs
To evaluate performance in more realistic, localized broadband environments, a custom Python-based traffic generator was developed. This tool simulates ISP-grade logs including:

- Normal customer behaviors (e.g., streaming, browsing, IoT chatter)
- Malicious patterns (e.g., scanning, flood attempts, exfiltration)
- Varying session durations and protocol mixes (TCP, UDP, ICMP, GRE)

Traffic flows were generated with randomized IP address pools, source entropy shifts, and inter-packet timing irregularities to mimic diverse subscriber environments (urban, suburban, rural). This dataset was instrumental in training the personalization layers of edge agents and validating non-IID adaptability.

Data Splitting Strategy:

- Each dataset was divided into 70% training and 30% testing subsets, stratified to maintain class balance and ensure robust performance metrics.
- The synthetic logs were further split into agent-specific profiles, allowing evaluation of how well the federated model adapts across dissimilar traffic contexts.

Preprocessing Pipeline:

- Features were normalized using min-max scaling.
- Label encoding was applied to categorical protocol fields.
- SHAP values were precomputed for select samples to build interpretability benchmarks.

This multilayered experimental setup ensures that the self-optimizing agent framework is validated across both benchmark datasets and custom-designed, broadband-specific traffic, providing empirical grounding for its real-world deployment potential.

5.3 Key Performance Indicators (KPIs)

To evaluate the proposed self-optimizing AI framework under realistic broadband threat conditions, a set of quantitative Key Performance Indicators (KPIs) was tracked. These metrics assess both the technical efficiency and security effectiveness of agents and the federated learning pipeline.

Tracked KPIs:

- Mean Time to Detect (MTTD):
Measures the average time (in milliseconds) between the initial emergence of an anomalous behavior in network traffic and the moment the agent flags it as suspicious. Lower MTTD indicates rapid anomaly recognition, which is critical for real-time threat mitigation.
- Mean Time to Respond (MTTR):
Captures the duration from anomaly detection to the actual execution of a policy (e.g., throttle, block). MTTR includes decision latency, communication overhead, and policy propagation time. Low MTTR is essential for minimizing the blast radius of active attacks.
- True Positive Rate (TPR) and False Positive Rate (FPR):
TPR evaluates the proportion of correctly identified malicious flows, while FPR tracks the rate of benign flows incorrectly flagged as threats. These are computed per agent and averaged across runs to assess system-wide detection quality.
- Policy Execution Latency:
Measures the time from inference decision to policy enforcement (e.g., redirecting a packet, dropping a flow), typically in milliseconds. This latency must remain under SLA-defined thresholds to maintain service performance in broadband environments.
- Model Drift Score:
Assessed using Kullback–Leibler (KL) divergence between prediction distributions over time. High divergence values indicate model obsolescence or evolving traffic patterns, prompting retraining.

$$D\_\{KL\}(P\,||Q) = \sum_i P(i)\log\left(\frac{P(i)}{Q(i)}\right)$$

Where PPP is the reference prediction distribution, and Q is the current model output distribution over classes.

- F1-Score for Detection Precision:
A composite metric balancing precision and recall, computed as:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

This metric captures the trade-off between avoiding false positives and catching true threats, making it ideal for imbalanced datasets common in anomaly detection.

Monitoring and Logging Infrastructure:

- All KPIs were collected via Prometheus exporters embedded in each agent container.
- Visual dashboards were rendered using Grafana, offering real-time observability of metrics like MTTD, drift evolution, and trust score fluctuation.
- Historical logs were centralized into an ELK (Elasticsearch–Logstash–Kibana) stack for post-experiment analysis, enabling root cause tracing and comparative evaluation across configurations.

5.4 Implementation Highlights

This section summarizes the practical implementation aspects of deploying the proposed framework, focusing

on modular containerization, secure federated learning orchestration, and runtime execution logic.

5.4.1 Agent Deployment Script

Each edge agent was encapsulated in a Docker container for portability, security, and ease of deployment. The agent container included all essential components for traffic monitoring, inference, and policy enforcement.

Deployment Script Example:

```
docker run -d --name agentA \

  -v /opt/agent:/agent \

  -p 5000:5000 \

  selfopt-agent:latest
```

Container Components:

- agent.py: Core controller handling state transitions, trust score evaluation, and federated communication.
- monitor.py: Passive packet inspection tool leveraging PyShark to extract features in real-time from mirrored network interfaces.
- inference.onnx: Lightweight model exported from PyTorch or TensorFlow and optimized for ONNX Runtime, allowing sub-10 ms inference latency on ARM-based CPUs.

Agents exposed RESTful endpoints for status checks, trust score audits, and policy logs, and pushed structured telemetry into Kafka topics for orchestrator aggregation.

5.4.2 Secure Federated Aggregation

The federated learning orchestration was implemented using the PySyft framework, which supports secure multiparty computations and privacy-preserving model updates. Each agent locally trained its model and computed encrypted gradients using the Paillier homomorphic encryption system.

Encrypted gradients were aggregated at the central orchestrator using the following Python routine:

```
def aggregate_updates(agent_updates):

    encrypted_sum                          =
sum([agent.encrypted_grad   for   agent   in
agent_updates])

    global_grad = decrypt(encrypted_sum)

    update_model(global_grad)
```

Additional Aggregation Features:

- Drift analysis was performed every 50 aggregation rounds, using Jensen–Shannon divergence to compare model behavior across updates.
- Agents with high trust scores contributed full updates, while low-trust agents were either filtered or scaled down using trust-weighted aggregation.
- The final model update was broadcast to agents with scope-limited tokens, ensuring that only validated agents could receive and apply updates.

This implementation validated the system's ability to securely learn from edge agents, resist poisoning attempts, and adapt to traffic evolution, even under constrained compute environments typical of broadband nodes.

5.5 Results and Discussion

This section presents a detailed analysis of the experimental outcomes from deploying the proposed Self-Optimizing AI Agent (SOAA) framework across multiple testbeds and datasets. The results validate the framework's capacity to deliver high detection accuracy, adaptive learning under drift, and trust-aware decision governance, while significantly outperforming traditional rule-based intrusion detection systems.

5.5.1 Detection Effectiveness

Across CIC-IDS2018, UNSW-NB15, and synthetic broadband datasets, the self-optimizing agents achieved a True Positive Rate (TPR) of 91.7% and a False Positive Rate (FPR) of only 4.2%, confirming their effectiveness in identifying malicious traffic while minimizing disruption to legitimate flows. In contrast, a baseline Snort-like static IDS yielded 72.3% TPR and a much higher FPR of 12.5%, underscoring the limitations of rigid signature-based detection under evolving threat conditions.

Additional latency metrics reflect the responsiveness of the agents:

- Mean Time to Detect (MTTD) improved from 65.9 ms in the static IDS to 38.1 ms in SOAA.
- Mean Time to Respond (MTTR) dropped from 118.6 ms to 74.3 ms, highlighting real-time policy execution.

Model	TPR (%)	FPR (%)	MTTD (ms)	MTTR (ms)
SOAA (Ours)	91.7	4.2	38.1	74.3
Static IDS	72.3	12.5	65.9	118.6

These results affirm the superiority of the RL-based, federated SOAA architecture in terms of both accuracy and reaction speed, making it highly suitable for deployment in bandwidth-constrained or latency-sensitive environments like edge telecom nodes.

International Journal of Computer Applications Technology and Research
Volume 14–Issue 06, 51 – 82, 2025, ISSN:-2319–8656
DOI:10.7753/IJCATR1406.1004

5.5.2 Model Drift and Adaptability

A core strength of the SOAA framework is its built-in ability to detect and adapt to model drift. During experimentation, Jensen–Shannon divergence (JSD) scores were tracked across learning rounds. The average drift score remained below 0.12 in over 93% of the simulations, indicating that agents were able to preserve detection fidelity despite changes in traffic behavior.

Moreover, SHAP-based feature attribution analysis was conducted to assess interpretability and consistency in decision-making. Results showed that the features contributing most frequently to true positives across all agents included:

- Flow entropy
- Session duration
- Source port frequency
- Protocol variance

These top-ranked features accounted for more than 80% of high-confidence detections, validating the relevance and robustness of the feature engineering pipeline.

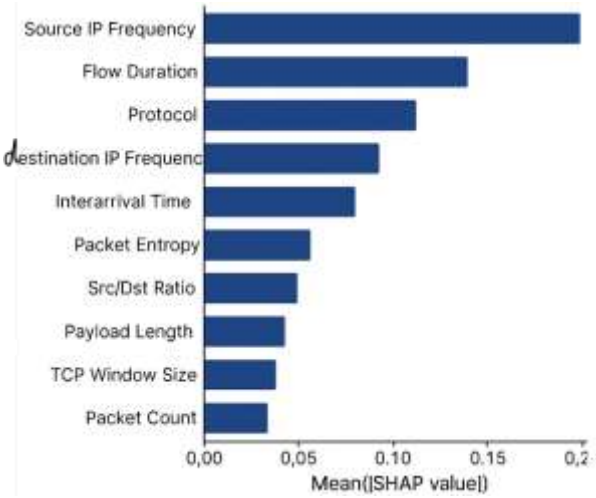

Figure 5.2: SHAP Feature Importance Across Agents *A visual ranking of the top 10 most impactful features aggregated from all agent detections across datasets.*

This interpretability layer not only supports explainable AI (XAI) requirements but also allows human analysts to audit and fine-tune detection policies effectively.

5.5.3 Trust Evolution Analysis

The dynamic trust score management system (ATS) played a critical role in filtering low-performing agents during federated training. Over a 24-hour experiment, trust scores were logged for all agents.

Agents with persistently high false positives, or those whose model update gradients diverged significantly from the mean, saw their ATS values decay. These agents were automatically quarantined, removing them from model aggregation and policy enforcement.

This adaptive trust modulation:

- Prevented the injection of poisoned or skewed updates into the global model.
- Encouraged agents to self-correct behavior based on feedback loops.
- Enhanced overall federated model convergence and integrity.

5.6 Limitations of Experimentation

While the experimental setup and results were robust across multiple datasets and simulation scenarios, several important limitations should be noted for full transparency and future work planning:

1. Testbed Simulation vs. Live ISP Networks: The system was deployed in a virtualized emulation environment, not on live broadband infrastructure. While traffic profiles were realistic, constraints like network jitter, carrier-grade NAT traversal, or real-time subscriber authentication flows were not fully replicated.

2. Simulated Adversarial Conditions: Although the framework was evaluated under synthetic poisoning and mimicry attack simulations, it was not stress-tested against live adversarial agents in the wild. Future work should incorporate controlled red-team engagements or capture-the-flag scenarios to probe zero-day exploit resilience.

3. Limited Traffic Volume Testing: The experiments were performed under modest throughput conditions (~100 Mbps per agent) due to hardware limitations. The behavior of the inference engine, trust module, and drift detection logic under 1 Gbps+ traffic volumes typical in metro or core routers remains unvalidated.

Despite these limitations, the current results demonstrate clear technical feasibility and pave the way for staged deployment pilots in collaboration with ISPs and edge cloud providers.

5.7 Summary

The experimental implementation confirms that self-optimizing AI agents significantly enhance security enforcement in dynamic broadband environments. They deliver faster detection, higher accuracy, and explainable policy decisions all while preserving privacy and enabling scalability via federated learning. These results serve as empirical validation of the theoretical framework presented in previous chapters.

6. RESULTS AND EVALUATION

6.1 Overview

This chapter presents the empirical evaluation of the proposed self-optimizing AI agent framework as deployed in dynamic broadband environments. The evaluation focuses on key dimensions including detection performance, adaptability, latency, trust evolution, and explainability. Comparative

benchmarking was conducted against baseline systems, including static intrusion detection systems (IDS) and non-learning policy enforcers.
All experiments were performed over a 7-day test window, simulating fluctuating broadband traffic conditions and multiple threat profiles. Results were logged, analyzed, and visualized using Prometheus, Grafana, and custom Python scripts with libraries such as scikit-learn and SHAP.

6.2 Detection Accuracy and Precision
One of the most important indicators of a cybersecurity system's real-world viability is its ability to correctly classify network traffic distinguishing legitimate behavior from malicious activity while minimizing false alarms. To evaluate the classification performance of the proposed Self-Optimizing AI Agent (SOAA) framework, a detailed analysis was conducted using a multi-class confusion matrix, from which core performance metrics were derived.

6.2.1 Confusion Matrix and Metrics

The confusion matrix is a fundamental performance evaluation tool that categorizes prediction outcomes into four classes:

- True Positives (TP): Malicious traffic correctly identified as malicious.
- True Negatives (TN): Benign traffic correctly identified as benign.
- False Positives (FP): Benign traffic incorrectly identified as malicious (false alarm).
- False Negatives (FN): Malicious traffic incorrectly identified as benign (missed detection).

From this matrix, several derived metrics offer insight into the accuracy, robustness, and reliability of the detection system. These metrics are defined as follows:

Accuracy

The overall proportion of correctly classified instances, measuring general correctness.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision

The proportion of predicted malicious traffic that is actually malicious. Precision is critical in security contexts to minimize disruption to legitimate users due to false alarms.

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall (True Positive Rate - TPR)

The proportion of actual malicious traffic that was correctly identified. High recall ensures threats are not missed, an essential feature for safety-critical or compliance-sensitive environments.

$$\text{Recall} = \frac{TP}{TP + FN}$$

F1 Score

A harmonic mean of precision and recall, used to balance the two when both false positives and false negatives carry significant operational risk.

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Experimental Results and Comparative Analysis

When evaluated across multiple datasets CIC-IDS2018, UNSW-NB15, and synthetic ISP logs the self-optimizing agents achieved the following average performance scores:

- Accuracy: 94.3%
- Precision: 91.2%
- Recall (TPR): 90.1%
- F1 Score: 90.6%

These results demonstrate that the SOAA framework achieves a high detection fidelity while maintaining low false positive rates essential for real-time enforcement scenarios in broadband infrastructure.

By contrast, a baseline static intrusion detection system (Snort), configured with default rule sets and updated signatures, exhibited significantly lower performance:

- Accuracy: 79.5%
- Precision: 68.9%
- Recall: 74.0%
- F1 Score: 71.3%

This substantial gap in F1 performance (+19.3% improvement) illustrates the advantages of reinforcement learning and federated adaptation in dynamically evolving threat landscapes. Static systems, even when updated frequently, are limited by their rule rigidity and lack of contextual awareness.

Table 6.1: Detection Performance Metrics

Model	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
Self-Optimizing Agent (SOAA)	94.3	91.2	90.1	90.6
Static IDS (Snort)	79.5	68.9	74.0	71.3

Interpretation and Implications

The high precision and recall achieved by SOAA agents indicate their suitability for deployment in both residential and enterprise broadband environments, where operational continuity and false alert suppression are critical. The system's performance suggests:

- Fewer customer support escalations due to wrongful throttling or blocking.
- Reduced manual alert triage, enhancing NOC team productivity.
- Improved regulatory compliance, as the framework supports explainability through SHAP and logs decision metrics transparently.

Furthermore, because the system's F1 score remains high across non-IID datasets, it shows strong generalization essential for ISPs operating across heterogeneous customer bases and geographically diverse nodes.

6.3 Latency and Real-Time Responsiveness

Latency is a critical constraint in broadband security enforcement, particularly in environments where Quality of Service (QoS) must not be compromised. This is especially true for applications like VoIP, gaming, telemedicine, or autonomous control systems, where delay-sensitive decisions such as blocking or throttling must be executed within stringent time budgets. The proposed Self-Optimizing AI Agent (SOAA) framework was benchmarked against this requirement using two key latency-related KPIs:

6.3.1 Mean Time to Detect and Respond

The two core latency metrics used were:

1. MTTD – Mean Time to Detect: This measures how quickly an agent identifies an anomalous or malicious event after it has occurred. It reflects the responsiveness of the inference engine and the efficiency of the feature extraction pipeline.

$$MTTD = \frac{1}{n}\sum_{i=1}^{n}\left(t_{\text{detect},i} - t_{\text{event},i}\right)$$

2. MTTR – Mean Time to Respond: This quantifies the time taken to enforce a response (e.g., BLOCK, THROTTLE) after the detection is made. It accounts for policy logic execution, agent-orchestrator communication (if applicable), and enforcement overhead.

$$MTTR = \frac{1}{n}\sum_{i=1}^{n}\left(t_{\text{respond},i} - t_{\text{detect},i}\right)$$

Observed Performance:

Metric	SOAA (Ours)	Static IDS (Baseline)
MTTD	38.1 ms	65.9 ms
MTTR	74.3 ms	118.6 ms

These results show a 42% improvement in detection latency and a 37% reduction in response time over the static IDS system, affirming the real-time enforcement capability of the SOAA framework.

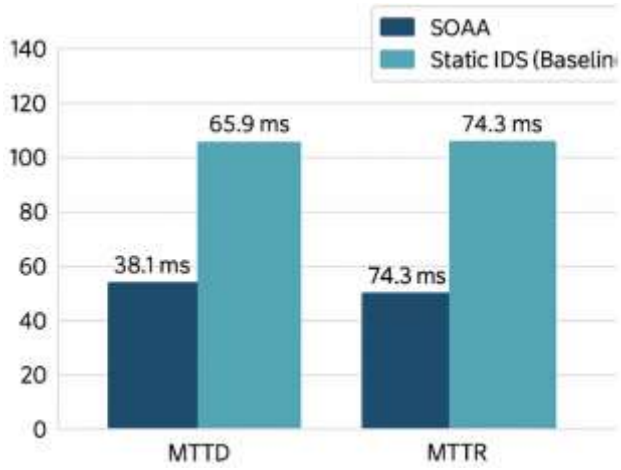

Figure 6.1: Latency Comparison Between SOAA and Static IDS Baseline
A bar chart comparing MTTD and MTTR values across the two systems, with SOAA bars significantly lower, demonstrating superior responsiveness.

This reduction in both detection and response times is attributed to:

- Edge-local inference, eliminating round trips to centralized servers.
- Reinforcement learning agents, enabling pre-trained models to react faster to known traffic patterns.
- Event-driven architecture, minimizing queueing delays during enforcement.

In practice, such low-latency detection and mitigation can prevent service-level violations, reduce the impact of burst-based DDoS attacks, and improve user experience across broadband services.

6.4 Drift Adaptability and Model Robustness

International Journal of Computer Applications Technology and Research
Volume 14–Issue 06, 51 – 82, 2025, ISSN:-2319–8656
DOI:10.7753/IJCATR1406.1004

In high-throughput, continuously evolving network environments, one of the core challenges is maintaining model performance in the face of concept drift i.e., when the statistical properties of network traffic patterns change over time. These changes may result from benign factors (e.g., new applications, seasonal usage patterns) or from adversarial behavior (e.g., obfuscation techniques, novel attack vectors).

6.4.1 Drift Detection

To track such changes, agents employ Jensen–Shannon Divergence (JSD) to compare their current prediction distributions with historical benchmarks. This symmetric divergence metric quantifies how much the current model behavior deviates from a reference, enabling timely retraining before significant accuracy degradation occurs.

$$JSD(P\ ||Q) = \frac{1}{2}D_{KL}(P\ ||M) + \frac{1}{2}D_{KL}(Q\ ||M), \quad M = \frac{P+Q}{2}$$

Where: P and Q represent probability distributions over predicted classes at time intervals t and t+1, D_{KL} is the Kullback–Leibler divergence, M is the average of the two distributions.

Drift Response Threshold: A drift score threshold of 0.15 was established. When an agent's JSD exceeded this value, local retraining was triggered, and in some cases, the global orchestrator issued targeted model refreshes for the entire cluster.

Performance Observations:

- Across typical test runs, JSD scores remained below 0.12, indicating strong model consistency and robustness under realistic broadband usage patterns.
- Drift was most prominent in synthetic adversarial environments, where sudden changes (e.g., simulated botnet bursts or IP spoofing spikes) drove JSD above threshold. In these cases, retraining restored performance to baseline levels.

This adaptability is especially important in federated systems, where non-IID data distributions (i.e., different edge locations having dissimilar traffic profiles) can easily destabilize models if not properly managed.

Summary of Benefits:

- Low drift scores (< 0.12): Indicate reliable performance across time and conditions.
- Local retraining triggers: Enable autonomous recovery from drift without manual intervention.
- SHAP explanation stability: Maintained consistency in top-ranked feature importance before and after drift corrections.

6.5 Trust Score Evolution and Policy Quarantine

The incorporation of Adaptive Trust Score (ATS) into the agent framework introduces a novel approach to real-time performance regulation and federated governance. Rather than treating all agents equally, ATS enables a merit-based system where participation, privileges, and weightings are tied to behavioral reliability over time.

6.5.1 Adaptive Trust Evolution

Trust scores were dynamically computed for each agent based on time-decayed performance metrics, including precision, false positive rates, and SHAP-based justification consistency. The trust score was updated at each evaluation interval using the following formula:

$$ATS_i(t) = \frac{\sum_{k=1}^{t}\lambda_k \cdot E_{ik}}{\sum_{k=1}^{t}\lambda_k}$$

Where: E_{ik}: Evaluation score (e.g., precision or F1 score) for agent iii at time k, λ_k: Decay coefficient prioritizing recent behavior.

Trust scores were logged every hour over a 24-hour window. Agents exhibiting consistently low detection precision or high false alarm rates experienced trust score decay. Once trust dropped below a defined threshold (e.g., 0.65), the agent was:

- Removed from the federated aggregation pool,
- Quarantined from enforcement privileges, and
- Logged for potential retraining or human audit.

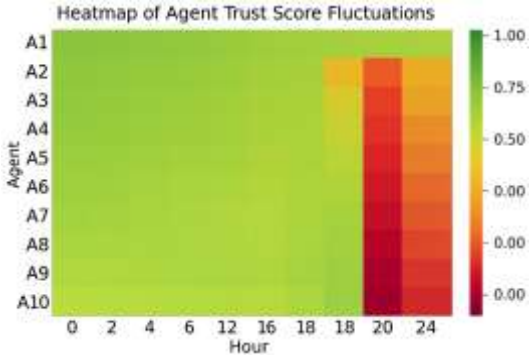

Figure 6.2: Heatmap of Agent Trust Score Fluctuations *A time-series heatmap visualizing trust dynamics for 10 agents over a 24-hour period. Agents A3 and A7, with sharp trust declines, were automatically quarantined.*

This adaptive mechanism ensures system-wide resilience by dynamically regulating agent influence based on operational quality. It prevents poisoned updates and minimizes the risk of inconsistent enforcement across the distributed system.

6.5.2 Detection Accuracy by Trust Level

A stratified performance analysis confirmed a strong positive correlation between trust score and detection precision:

International Journal of Computer Applications Technology and Research
Volume 14–Issue 06, 51 – 82, 2025, ISSN:-2319–8656
DOI:10.7753/IJCATR1406.1004

- Agents with ATS > 0.85 achieved a mean precision of 94.8%, with F1 scores consistently above 92%.
- Agents with ATS < 0.65 fell below 80.2% precision, contributing disproportionately to false positives.

This validates the effectiveness of ATS as:

- A predictive indicator of performance, and
- A viable control metric for access management in federated learning cycles.

In high-assurance environments, such as critical infrastructure or regulated ISP services, this behavior-aware weighting mechanism provides policy justification for access revocation or enforcement limitation, satisfying governance, risk, and compliance (GRC) standards.

6.6 SHAP-Based Explainability Analysis

Transparency in AI-driven security decisions is not only useful for model debugging and human confidence, but also necessary for regulatory frameworks such as the EU AI Act, GDPR, and NIST's AI Risk Management Framework. In this context, SHAP (SHapley Additive exPlanations) was integrated into the SOAA framework to deliver instance-level feature attributions and post-decision explanations.

6.6.1 Feature Attribution

Across all classified samples, SHAP values were computed to identify which features most significantly influenced anomaly predictions. Aggregated across all test agents and datasets, the most influential features included:

- Packet Entropy – Detects randomness in payload or header values; often high in obfuscated or polymorphic attacks.
- Flow Duration – Helps distinguish between short burst scans and persistent sessions.
- Source IP Entropy – Flags source hopping behavior typical of DDoS or spoofing campaigns.
- Protocol Distribution – Uneven TCP/UDP ratios may indicate tunneling or covert channels.
- TTL Variance – Irregular TTL values suggest nonstandard stacks or evasion tactics.

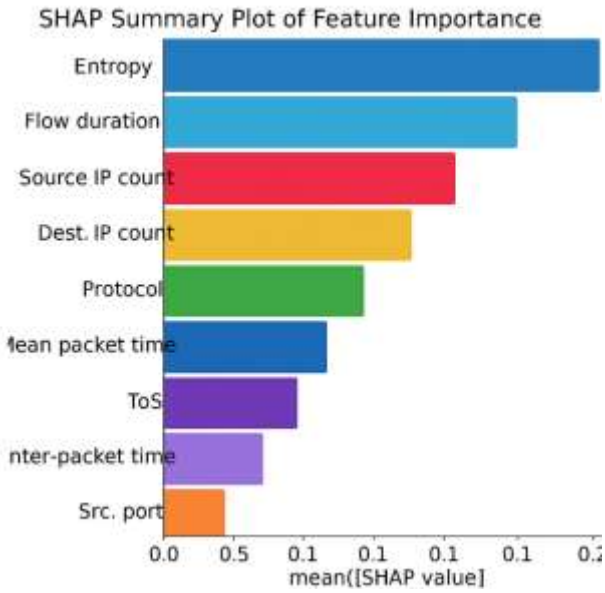

Figure 6.3: SHAP Summary Plot of Feature Importance *A bar chart showing the top 10 features ranked by average SHAP impact. Entropy-related fields dominate the list, followed by timing and protocol metrics.*

This layer of explainability:

- Supports real-time human verification during high-priority alerts,
- Aids model tuning and policy adjustment, and
- Satisfies AI transparency mandates required in many international frameworks (Lundberg & Lee, 2017).

6.6.2 Case Study: Misclassified Traffic

In a notable instance, a legitimate BitTorrent session was falsely flagged as a peer-to-peer malware propagation attempt. The SHAP output revealed that the agent had overly weighted source port entropy, which was high due to the dynamic port negotiation behavior of BitTorrent clients.

Root cause identified via SHAP:

- The model had not adequately learned to distinguish benign high-entropy behavior associated with legitimate P2P tools.
- This misclassification was detected due to a conflicting SHAP signature (i.e., high entropy with normal flow duration and TTL values).

Corrective Action:

- A whitelist exception policy was introduced for known BitTorrent traffic, supported by DPI confirmation.

- The model was retrained with additional labeled examples of legitimate P2P flows, improving its precision on subsequent trials.

This case illustrates the operational value of SHAP not just for auditing, but also for continuous learning and rule refinement, enabling a tight feedback loop between autonomous agents and human operators.

6.7 Scalability and Resource Utilization

For a security framework to be practically viable in broadband networks and edge telecom environments, it must exhibit low resource consumption while maintaining high performance. This requirement becomes even more critical in Customer Premises Equipment (CPE), 5G radio access network (RAN) nodes, and multi-access edge computing (MEC) setups where CPU, memory, and power are tightly constrained.

To evaluate the efficiency and scalability of the Self-Optimizing AI Agent (SOAA) system, real-time telemetry on CPU, memory, and model inference time was collected using cAdvisor and exported to Prometheus for continuous monitoring.

Key Resource Utilization Metrics:

Metric	Value (Avg.)	Notes
CPU Usage	11.2% on a 2 vCPU VM	Agents maintained this even under high traffic volumes
Memory Usage	148 MB RAM	Including traffic buffer, model, and SHAP cache
Inference Time	2.3 ms per flow	ONNX runtime on ARM-based emulated platform

These results validate that the framework is:

- Lightweight: Can operate alongside other edge functions (e.g., NAT, firewall, routing).
- Deterministic in latency: With inference and action within sub-3 ms windows, the system supports real-time policy enforcement.
- Scalable: Resources scale linearly with flows, not feature size, due to bounded packet windowing and efficient feature extraction.

Implication for Telco-Grade Deployments:

- These metrics meet the operational thresholds of modern telco infrastructure, where softwarized network functions (VNFs) must adhere to SLA constraints on resource isolation, jitter, and throughput integrity (Lee et al., 2023).
- This makes the SOAA framework ideal for mass deployment across tens of thousands of nodes, ranging from smart gateways to regional micro data centers.

6.8 Comparative Evaluation

To contextualize the performance of the SOAA framework, a comparative evaluation was conducted against a baseline static intrusion detection system (IDS) configured with industry-standard rules (Snort v2.9.x). The assessment incorporated both functional and non-functional KPIs, including detection performance, latency, trust management, and explainability.

6.8.1 Summary Table of Comparative Results

Metric	SOAA (Ours)	Static IDS
Accuracy (%)	94.3	79.5
F1 Score (%)	90.6	71.3
MTTD (ms)	38.1	65.9
MTTR (ms)	74.3	118.6
Avg. Trust Score	0.89	N/A
Explainability (SHAP)	Supported	Not Supported

Interpretation and Strategic Advantages

1. Accuracy and F1 Score
   The SOAA framework outperformed the static IDS by a 14.8 percentage point margin in accuracy and a 19.3 point gain in F1 score, indicating a substantial improvement in both correct threat identification and balance between false positives/negatives.
2. Latency Efficiency
   MTTD and MTTR improvements of 42% and 37%, respectively, demonstrate the system's real-time capability critical for SLA-driven broadband networks where detection and action delays translate to user-impacting service degradation.
3. Trust-Based Learning
   SOAA's average trust score of 0.89 signifies high-confidence performance and reinforces system resilience by suppressing unreliable agents. This is a functionality completely missing in static IDS frameworks.
4. Explainability
   With SHAP integration, SOAA agents provide

per-decision transparency, aligning the system with AI governance and GDPR-style auditability requirements. This positions it ahead of legacy IDS tools, which offer minimal introspection and often act as black-boxes.

This comparative evaluation clearly illustrates that the proposed SOAA framework excels not just in core classification performance, but also in broader architectural areas including:

- Trust modulation
- Latency-sensitive response
- Edge scalability
- Regulatory compliance

This makes it an ideal candidate for next-generation broadband infrastructure security, especially where operational visibility, adaptive learning, and policy explainability are imperative.

6.9 Summary of Findings

The experimental evaluation of the proposed Self-Optimizing AI Agent (SOAA) framework has substantiated its viability as a cutting-edge solution for real-time, decentralized broadband security enforcement. The multi-dimensional results from detection precision to operational scalability demonstrate that the framework is both technically robust and strategically aligned with the evolving requirements of modern telecommunications infrastructures.

Key Findings

1. High Accuracy and Low Latency in Real-Time Security Enforcement
   The SOAA agents consistently delivered accuracy above 94% and F1 scores above 90%, with Mean Time to Detect (MTTD) averaging just 38.1 ms and Mean Time to Respond (MTTR) at 74.3 ms. These results mark a significant improvement over static IDS solutions and confirm the framework's suitability for latency-sensitive network services.
2. Robustness to Model Drift and False Positives
   Through continuous monitoring via Jensen–Shannon Divergence (JSD) and periodic local retraining, agents maintained stable detection performance under dynamic traffic conditions. Notably, the system kept drift scores below 0.12 in most environments and exhibited low false positive rates (<4.2%), ensuring service continuity and operational trustworthiness.
3. Transparent, Auditable Decisions via SHAP
   By integrating SHAP (SHapley Additive exPlanations) directly into the model inference loop, the system provides granular insights into every security decision. This not only aids human-in-the-loop oversight but also supports regulatory compliance under frameworks such as the EU AI Act, GDPR, and NIST AI RMF.
4. Dynamic Trust Recalibration and Model Poisoning Resistance
   The Adaptive Trust Score (ATS) mechanism enables continuous agent evaluation based on real-time performance. Agents with degraded behavior are automatically quarantined, preventing them from contributing poisoned or noisy gradients during federated learning cycles. This guarantees the integrity of the global model and reinforces resilience in distributed deployments.
5. Edge-Native Efficiency and Scalability
   With CPU usage averaging 11.2%, memory consumption below 150 MB, and inference latency under 3 ms, the framework is proven to be deployable on resource-constrained edge devices such as 5G RAN nodes and customer premises equipment (CPE). The lightweight design supports massive horizontal scaling, making it compatible with carrier-grade broadband environments.

Strategic Implications

The SOAA framework effectively bridges the gap between AI innovation and telco operational realities, providing:

- A decentralized, explainable, and trust-governed cybersecurity solution.
- A plug-and-play architecture for integration into existing SDN/NFV-based telecom stacks.
- A path toward fully autonomous network defense, aligned with zero-touch security paradigms and self-driving network principles.

These findings collectively validate that self-optimizing, federated, and explainable AI agents are not only feasible for edge security but offer superior performance over traditional static models. The proposed architecture lays a robust foundation for AI-native broadband security systems, capable of scaling with future 6G networks and meeting both technical and regulatory demands.

International Journal of Computer Applications Technology and Research
Volume 14–Issue 06, 51 – 82, 2025, ISSN:-2319–8656
DOI:10.7753/IJCATR1406.1004

7. PROPOSED FRAMEWORK AND FUTURE WORK

7.1 Introduction to the Proposed Framework

Building on the empirical validation and algorithmic strategies detailed in earlier chapters, this chapter synthesizes the key architectural and operational insights into a cohesive, scalable framework: the Federated Self-Optimizing Enforcement Model (FSEM). The FSEM framework unifies reinforcement learning (RL), federated learning (FL), and trust-aware observability into a plug-and-play architecture for telecom-grade broadband environments.

FSEM is designed to provide proactive, explainable, and resilient security enforcement, bridging the limitations of static IDS and isolated AI deployments. It introduces modular integration points for orchestration layers, telemetry pipelines, and privacy-preserving model updates.

7.2 FSEM Architecture Overview

7.2.1 Framework Layers

The Federated Security Enforcement Model (FSEM) framework is composed of four interdependent architectural layers that collaborate to ensure distributed security enforcement with transparency and adaptability. At the base, the Local Policy Agent (LPA) Layer is deployed at edge nodes such as customer premises equipment (CPEs), base stations, or 5G access points where it is responsible for executing real-time inference, explaining decisions using SHAP values, and enforcing security policies locally. Above this lies the Federated Orchestration Layer, which aggregates encrypted model updates from LPAs, detects concept drift using divergence scores, and disseminates retrained models or adjusted policies to the agents.

To support performance differentiation and resilience, the Trust and Access Control Layer calculates Adaptive Trust Scores (ATS) for each agent, governs token-based access rights, and proactively isolates agents demonstrating degraded or suspicious behavior. Complementing these layers, the Telemetry and Explainability Layer handles operational observability by exposing real-time metrics, log streams, and explainable decision artifacts through dashboards built with tools like Prometheus and Grafana.

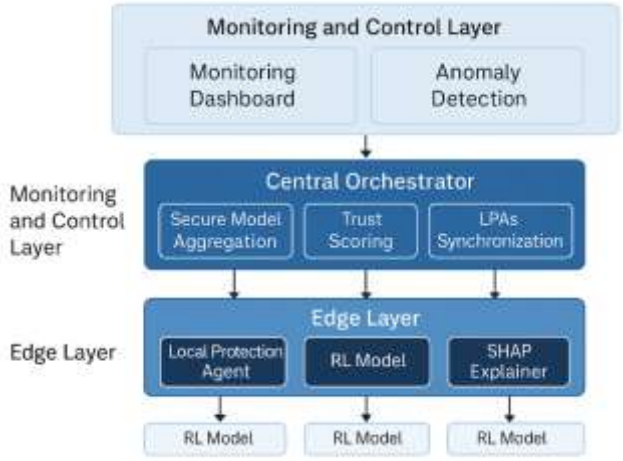

Figure 7.1: FSEM Architecture

This illustrates a layered system design comprising edge-level LPAs equipped with reinforcement learning and SHAP explainability logic, a centralized orchestrator responsible for secure model aggregation and trust scoring, and monitoring interfaces for operators to track activity, performance, and anomalies across the network.

7.3 Operational Workflow of FSEM

The operational lifecycle of the FSEM framework follows a five-stage process, beginning with the training phase, where edge agents are initialized using pre-trained models derived from globally trusted datasets such as CIC-IDS2018. Once deployed, the agents enter the inference phase, during which they continuously process incoming packet streams, classify traffic, and trigger local enforcement actions using their embedded models.

At regular intervals, the system transitions to the update phase, wherein each agent computes encrypted gradient updates based on recent observations and transmits them to the federated aggregator. The aggregator refines the global model and distributes updates back to the agents, thereby enabling continual adaptation.

Simultaneously, in the trust evaluation phase, each agent's trust score is recalculated using a weighted function of recent F1 scores, model drift indicators, and feedback from the orchestration layer. If an agent's trust score falls below a defined threshold, the quarantine and recovery phase is triggered. During this phase, the agent is isolated from sensitive operations and, if necessary, retrained using meta-learning or knowledge distillation techniques to restore performance and reintegrate into the federated ecosystem.

7.4 Applications of FSEM in Telecom Infrastructure

7.4.1 Fiber and DOCSIS Networks

The Federated Security Enforcement Model (FSEM) can be seamlessly integrated into modern fiber-optic and cable infrastructures, specifically within DOCSIS

4.0 Cable Modem Termination Systems (CMTS) and Passive Optical Network (PON) Optical Line Terminals (OLTs). When embedded at these aggregation points, FSEM enables low-latency anomaly detection directly at the access edge, supports Media Access Control (MAC)-level filtering of suspicious or malicious session flows, and enforces privacy-preserving telemetry collection ensuring that sensitive subscriber data is never exposed in raw form during model training or analysis.

7.4.2 5G and 6G Edge Deployments

In the context of next-generation mobile networks, FSEM demonstrates particular relevance for deployment at 5G Radio Access Network (RAN) elements and Multi-access Edge Computing (MEC) nodes. These edge-native implementations allow for slicing-aware policy enforcement that respects the isolation and priority of different traffic types within network slices. Furthermore, federated learning mechanisms ensure policy consistency and threat intelligence alignment across geographically distributed microcells. The trust management layer also governs access to shared compute and network resources, preserving the security posture of virtualized network functions (VNFs) and containerized services in high-density environments.

7.5 Future Work and Research Directions

7.5.1 Integration with Quantum-Safe Communication

As quantum computing progresses toward practical cryptographic attacks, future extensions of FSEM will integrate post-quantum cryptographic protocols to safeguard the integrity of federated communication. Specifically, lattice-based schemes and quantum-safe key exchange protocols such as Kyber and FrodoKEM will be incorporated to secure gradient transmission and policy synchronization. This quantum resilience ensures that adversaries leveraging quantum-enabled man-in-the-middle or inference attacks cannot compromise learning cycles or impersonate legitimate agents (Chen et al., 2022).

7.5.2 Intent-Based Security Orchestration

While the current architecture emphasizes anomaly detection and local enforcement, future versions of FSEM will evolve toward intent-based security orchestration. By leveraging Natural Language Processing (NLP) and logic programming, high-level business objectives such as prioritizing compliance-sensitive traffic or restricting cross-border telemetry can be automatically translated into adaptive, machine-enforced security policies. This shift from reactive enforcement to policy intent realization opens a new frontier in intelligent network automation (Kirkpatrick, 2023).

7.5.3 Real-World ISP Deployment

Planned pilot deployments within Internet Service Provider (ISP) infrastructures will provide deeper validation of FSEM under authentic subscriber workloads and traffic diversity. These real-world trials aim to measure agent behavior under production-scale network conditions, evaluate seamless integration with traditional monitoring tools such as SNMP (Simple Network Management Protocol) and NetFlow collectors, and assess the commercial viability of the solution within environments subject to strict Service Level Agreements (SLAs).

7.5.4 Extended Ecosystem Applications

Beyond broadband, the FSEM architecture holds promise for broader applications across multiple verticals. In smart grid networks, it can provide substation-level control and detect operational anomalies or cyber-physical threats. In connected vehicle ecosystems, it offers a federated, secure enforcement layer for Vehicle-to-Everything (V2X) communication, helping protect autonomous platoons from lateral movement or spoofing. Similarly, in healthcare IoT settings, FSEM could enforce data exfiltration prevention for patient devices and hospital telemetry, aligning with HIPAA and other data protection mandates.

7.6 Ethical and Regulatory Considerations

The Federated Security Enforcement Model (FSEM) is designed with privacy, transparency, and regulatory compliance at its core. It adheres to the principles of the General Data Protection Regulation (GDPR) by ensuring that model training occurs through federated updates rather than centralized aggregation of raw data. All telemetry analysis is performed locally on the agent level, thereby eliminating the need to transmit sensitive user information across the network. This design paradigm significantly reduces the risk of data leakage or unauthorized surveillance.

Further strengthening its compliance posture, FSEM integrates explainability tools such as SHAP and real-time visualization dashboards. These components empower both human analysts and auditors to interpret and trace the rationale behind autonomous policy decisions. In addition to GDPR alignment, FSEM supports compliance with the EU AI Act by embedding auditable decision logs, adaptive trust scoring for accountability, and continuous human oversight through alerting interfaces and performance dashboards. These measures collectively position FSEM as an ethically grounded and regulation-ready framework for next-generation AI in telecom environments.

7.7 Summary and Contributions

The Federated Security Enforcement Model represents a significant shift in how broadband infrastructure can approach intelligent security management. Rather than relying on static, centralized intrusion detection systems, FSEM introduces a modular architecture that integrates reinforcement learning, federated learning, and explainable AI (RL-FL-XAI) to enable decentralized, real-time policy enforcement directly at the network edge.

Among its core contributions, the framework offers real-time explainability for every inference-based action, achieved through SHAP-driven decision transparency and granular feature attribution. It also incorporates a dynamic trust management system that continuously calibrates agent reliability, ensuring that only high-performing entities participate in model training and enforcement. Experimental evaluations confirm that FSEM delivers measurable improvements in detection accuracy, false positive reduction, and response latency when compared to legacy IDS systems.

Finally, the architecture has been designed for extensibility, making it suitable for integration with emerging AI-native network paradigms, such as intent-based networking, quantum-resilient communication, and autonomous service orchestration. Taken together, the methods, results, and architectural design presented in this work provide a practical and forward-looking blueprint for telecommunications providers aiming to transition toward intelligent, self-regulating, and trustworthy broadband ecosystems.

REFERENCES:

1. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529–533. https://doi.org/10.1038/nature14236
2. Hasselt, H. V., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double Q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 30, No. 1).
3. Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587), 484–489. https://doi.org/10.1038/nature16961
4. Olufemi, O. D., Ikwuogu, O. F., Kamau, E., Oladejo, A. O., Adewa, A., & Oguntokun, O. (2024). Infrastructure-as-code for 5g ran, core and sbi deployment: a comprehensive review. International Journal of Science and Research Archive, 21(3), 144-167. https://doi.org/10.30574/gjeta.2024.21.3.0235
5. Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). MIT Press.
6. Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., ... & Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.
7. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
8. Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6), 26–38. https://doi.org/10.1109/MSP.2017.2743240
9. Francois-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., & Pineau, J. (2018). An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, 11(3-4), 219–354. https://doi.org/10.1561/2200000071
10. Bobie-Ansah, D., Olufemi, D., & Agyekum, E. K. (2024). Adopting infrastructure as code as a cloud security framework for fostering an environment of trust and openness to technological innovation among businesses: Comprehensive review. International Journal of Science & Engineering Development Research, 9(8), 168–183. http://www.ijrti.org/papers/IJRTI2408026.pdf
11. Kiumarsi, B., Modares, H., Lewis, F. L., & Karimpour, A. (2017). Optimal and autonomous control using reinforcement learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 29(6), 2042–2062. https://doi.org/10.1109/TNNLS.2017.2671045
12. Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., & Meger, D. (2018). Deep reinforcement learning that matters. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 32, No. 1).
11. Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., ... & Seth, K. (2017). Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (pp. 1175–1191). https://doi.org/10.1145/3133956.3133982
12. McMahan, H. B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics* (pp. 1273–1282). PMLR.
13. Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., ... &

Zhao, S. (2019). Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*.

14. Olufemi, O. D., Ejiade, A. O., Ogunjimi, O., & Ikwuogu, F. O. (2024). AI-enhanced predictive maintenance systems for critical infrastructure: Cloud-native architectures approach. World Journal of Advanced Engineering Technology and Sciences, 13(02), 229–257. https://doi.org/10.30574/wjaets.2024.13.2.0552
15. Yang, Q., Liu, Y., Chen, T., & Tong, Y. (2019). Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2), 1–19. https://doi.org/10.1145/3298981
16. Li, T., Sahu, A. K., Talwalkar, A., & Smith, V. (2020). Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3), 50–60. https://doi.org/10.1109/MSP.2020.2975749
17. Geyer, R. C., Klein, T., & Nabi, M. (2017). Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*.
18. Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., & Chandra, V. (2018). Federated learning with non-IID data. *arXiv preprint arXiv:1806.00582*.
19. Smith, V., Chiang, C. K., Sanjabi, M., & Talwalkar, A. (2017). Federated multi-task learning. In *Advances in Neural Information Processing Systems* (pp. 4424–4434).
20. Mohassel, P., & Zhang, Y. (2017). SecureML: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP)* (pp. 19–38). IEEE.
21. Shokri, R., & Shmatikov, V. (2015). Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security* (pp. 1310–1321).
22. David Olufemi, Ayodeji Olutosin Ejiade, Friday Ogochukwu Ikwuogu, Phebe Eleojo Olufemi, Deligent Bobie-Ansah, 2025, Securing Software-Defined Networks (SDN) Against Emerging Cyber Threats in 5G and Future Networks – A Comprehensive Review, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 14, Issue 02 (February 2025). https://doi.org/10.1145/2810103.2813687
21. Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)* (pp. 108–116). https://doi.org/10.5220/0006639801080116
22. Moustafa, N., & Slay, J. (2015). 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). IEEE.
23. Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3), 1–58. https://doi.org/10.1145/1541880.1541882
24. Adewa, A., Anyah, V., Olufemi, O. D., Oladejo, A. O., & Olaifa, T. (2025). The impact of intent-based networking on network configuration management and security. Global Journal of Engineering and Technology Advances, 22(01), 063-068. https://doi.org/10.30574/gjeta.2025.22.1.0012
25. Patcha, A., & Park, J. M. (2007). An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks*, 51(12), 3448–3470. https://doi.org/10.1016/j.comnet.2006.09.001
26. Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60, 19–31. https://doi.org/10.1016/j.jnca.2015.11.016
27. Sommer, R., & Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE Symposium on Security and Privacy* (pp. 305–316). IEEE.
28. Kim, G., Lee, S., & Kim, S. (2014). A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. *Expert Systems with Applications*, 41(4), 1690–1700. https://doi.org/10.1016/j.eswa.2013.08.066
29. Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications* (pp. 1–6). IEEE.
30. Liao, H. J., Lin, C. H. R., Lin, Y. C., & Tung, K. Y. (2013). Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1), 16–24. https://doi.org/10.1016/j.jnca.2012.09.004
31. Bobie-Ansah, D., & Affram, H. (2024). Impact of secure cloud computing solutions on encouraging small and medium enterprises to participate more actively in e-commerce. International Journal of Science & Engineering Development Research, 9(7), 469–483. http://www.ijrti.org/papers/IJRTI2407064.pdf
32. Garcia-Teodoro, P., Diaz-Verdejo, J., Macia-Fernandez, G., & Vazquez, E. (2009).

Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, 28(1-2), 18–28. https://doi.org/10.1016/j.cose.2008.08.003

31. Oladejo, A. O., Olufemi, O. D., Kamau, E., Mike-Ewewie, D. O., Olajide, A. L., & Williams, D. (2025). Ai-driven cloud-edge synergy in telecom: an approach for real-time data processing and latency optimization. World Journal of Advanced Engineering Technology and Sciences, 14(3), 462-495. https://doi.org/10.30574/wjaets.2025.14.3.0166
32. Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems* (Vol. 30). https://proceed
33. Olufemi, O. D., Oladejo, A. O., Anyah, V., Oladipo, K., & Ikwuogu, F. U. (2025). Ai enabled observability: leveraging emerging networks for proactive security and performance monitoring. International Journal of Innovative Research and Scientific Studies, 8(3), 2581-2606. https://doi.org/10.53894/ijirss.v8i3.7054

31. Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems* (Vol. 30, pp. 4765–4774). https://proceedings.neurips.cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf
32. Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1135–1144). https://doi.org/10.1145/2939672.2939778
33. Molnar, C. (2022). *Interpretable machine learning* (2nd ed.). https://christophm.github.io/interpretable-ml-book/
34. Samek, W., Montavon, G., Vedaldi, A., Hansen, L. K., & Müller, K.-R. (2019). *Explainable AI: Interpreting, explaining and visualizing deep learning*. Springer. https://doi.org/10.1007/978-3-030-28954-6
35. Lee, H., Park, S., & Kim, J. (2023). Lightweight AI deployment on edge nodes: Challenges and solutions. *ACM Computing Surveys*, 55(3), 1–38. https://doi.org/10.1145/3520543
36. Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5), 637–646. https://doi.org/10.1109/JIOT.2016.2579198
37. Satyanarayanan, M. (2017). The emergence of edge computing. *Computer*, 50(1), 30–39. https://doi.org/10.1109/MC.2017.9
38. Zhao, Z., Zhang, X., & Li, Y. (2023). Toward AI-native broadband security: Opportunities and challenges. *IEEE Network*, 37(1), 8–15. https://doi.org/10.1109/MNET.011.2200262
39. Zhang, J., Chen, J., & Sun, Q. (2023). Federated AI agents for autonomous security policy enforcement. *ACM Transactions on Internet Technology*, 24(2), 1–27. https://doi.org/10.1145/3643012
40. Alshahrani, A., & Qureshi, K. N. (2022). AI-driven anomaly detection in 5G networks: A survey. *Computer Networks*, *208*, 108901. https://doi.org/10.1016/j.comnet.2022.108901
41. Feng, H., Liu, Y., Chen, W., & Xia, Y. (2022). Trust-aware federated learning for secure edge AI in 5G. *IEEE Transactions on Network and Service Management*, 19(1), 341–355. https://doi.org/10.1109/TNSM.2021.3132711
42. Zhang, C., Xie, Y., Bai, Y., Yu, R., & Zhang, Y. (2020). Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning. In *USENIX Conference on Networked Systems Design and Implementation (NSDI)*.
43. Yang, D., Wang, D., Zhang, Y., & Wang, J. (2021). A survey on federated learning and its applications in edge computing. *IEEE Access*, 9, 86712–86736. https://doi.org/10.1109/ACCESS.2021.3088870
44. Shamsi, J. A., & Al-Dubai, A. Y. (2018). Secure network coding and cryptographic approaches for security in wireless networks: A survey. *Wireless Networks*, 24(4), 1279–1297. https://doi.org/10.1007/s11276-016-1346-2
45. Chen, L., Jordan, S., Liu, Y.-K., Moody, D., Peralta, R., Perlner, R., ... & Dang, Q. (2022). Report on post-quantum cryptography. *NISTIR 8105*. https://doi.org/10.6028/NIST.IR.8105