International Journal of Web of Multidisciplinary Studies
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
website: *https://ijwos.com*
Volume 03, Issue 10,October 2026
OPEN ACCESS
E-ISSN : 3049-2424
DOI: https://doi.org/10.71366/ijwos

Clinically Accountable Predictive Models for Alzheimer's disease Stratification from Phenotypic Data.

SHEIK FIDAUL HUSSAIN[1], TAMARANA KOUSALYA[2], TAMARANA MANI[3],M SAGAR[4]
[1,2,3]*Master of Computer Applications Final Year Students, Avanthi Institute of Engineering & Technology, Anakapalli, A.P, India*
[4]*Asst Prof in CSE Department, Avanthi Institute of Engineering & Technology, Anakapalli, A.P, India*

Article Info

Article History:
Published: 31 Oct 2026

Publication Issue:
Volume 03, Issue 10
October 2026

Page Number:
660-669

Corresponding Author:
SHEIK FIDAUL HUSSAIN

Abstract:
Modern enterprise network architectures demand highly resilient, ultra-low latency intrusion detection frameworks capable of parsing massive volumes of heterogeneous packet telemetrics while uncovering sophisticated, distributed cyber threats. Traditional static signature-matching and isolated anomaly detection models fail to capture the evolving, multi-stage spatiotemporal signatures characteristic of advanced persistent threats (APTs) and coordinated
lateral movements. This paper presents an end-to-end, event-driven machine learning architecture engineered for flexible, real-time Cyber Attack Detection Using Spatiotemporal Patterns. The proposed platform decouples compute workloads
across a synchronized four-layer pipeline consisting of an edge-enabled Ingestion Layer, a concurrent Streaming and Storage Layer, a Spatiotemporal Feature Graph Assembly Pipeline, and a hardware-accelerated Predictive Inference
Engine. Transactional flows transition continuously from raw network packet records ingested via non-blocking proxies into distributed Apache Kafka partitions, stream-aggregated in Apache Flink, enriched with topological contexts via an in- memory Redis cluster, and evaluated via a hybrid Graph Convolutional Network (GCN) and Long Short-Term Memory (LSTM) network on NVIDIA Triton Inference Server. Operating within a strict sub-15ms SLA, the platform achieves a PR-AUC of 0.892, an attack recall of 92.4%, and triggers automated Software-Defined Network (SDN) containment interrupts upon threshold violation ($P(Attack) \geq 0.78$).

Keywords: Cyber Attack Detection, Advanced Persistent Threats (APTs), Graph-Contextual Vector Stitching, Automated Incident Containment, Software-Defined Networking (SDN), GCN-LSTM.

1. Introduction

Modern enterprise networks generate massive volumes of heterogeneous log telemetry, characterized by high-velocity packet transfers, host authentication sequences, and decentralized firewall state changes. For enterprise security operations centers (SOCs) and cloud infrastructure providers, isolating malicious actions within this streaming data requires threat detection systems that balance complex pattern recognition with deterministic, sub-millisecond execution. Traditional intrusion detection systems (IDS) rely heavily on static signature matching or isolated point-in-time anomaly detection. However, these paradigms fail against sophisticated advanced persistent threats (APTs), zero-day exploits, and coordinated lateral movements. These modern attack vectors execute across highly distributed network segments and long temporal durations, leaving behind subtle signatures that can only be uncovered by evaluating spatiotemporal patterns.

Translating multi-stage spatiotemporal threat detection into a live production environment introduces significant systems engineering challenges. To capture and neutralize an intrusion before lateral infection spreads, an enterprise defense framework must ingest raw packet-level logs, compute rolling structural network relationships, and execute machine learning inferences within ultra-low latency constraints.

Copyright © 2026 The Author(s) : This is an open access article under the CC BY license (http://creativecommons.org/licenses/by-nc/4.0/)

To address these core engineering issues, this paper details the design and deployment of a decoupled, event-driven machine learning architecture designed for real-time **Cyber Attack Detection Using Spatiotemporal Patterns**. Operating within a strict service level agreement (SLA) processing ceiling of sub-15 milliseconds, the platform transforms raw network transaction logs into continuous spatiotemporal vectors to execute predictive threat classification.

The End-to-End Security Processing Pipeline

The system ensures reliability and fault isolation by routing every inbound network transaction and log payload through a structured, four-tier processing pipeline:

High-frequency raw data streams—encompassing netflow records, firewall state updates, syslog entries, and cryptographic handshake packets—are captured at the network edge proxy. Using a non-blocking ingestion proxy layer, the gateway validates schema alignment and parses inbound structural parameters within a targeted sub-3ms window.

Streaming Event Buffering & Topology Storage

Layer 2

Validated log payloads are instantly serialized into compressed binary formats and broadcast to an Apache Kafka cluster partitioned by source_subnet_id or host_id. Keying by network segment guarantees strict chronological order preservation for local host events, while simultaneously mirroring the stream to an analytical cold data lake for offline model training.

Spatiotemporal Feature Vector Assembly

Layer 3

An Apache Flink stream processor consumes the active Kafka queues, maintaining stateful sliding memory windows to compute localized temporal features (e.g., connection velocities, rolling login failure rates). Concurrently, microsecond-scale parallel MGET queries target an in-memory Redis cluster to stitch spatial topology profiles and historical asset vulnerability scores into an aligned spatiotemporal vector.

Flexible Predictive Inference Engine

Layer 4

The fully assembled spatiotemporal vector is transmitted over low-overhead binary gRPC channels to an optimized machine learning ensemble hosted on a Triton Inference Server. The classifier evaluates the dense array across gradient-boosted structural paths to capture multi-stage anomalies, outputting an absolute intrusion probability score ($P(\text{Attack})$) in under 3ms.

Classification-Driven Automated Containment

The pipeline terminates at a dynamic containment mitigation node governed directly by the model's structural inference output. Under nominal operations where the probability of malicious activity remains below a validated threat boundary ($P(\text{Attack}) < 0.78$), the network event passes safely through the routing switch, and its computed features are stored to update behavioral baseline profiles.

However, if the Triton-hosted machine learning ensemble flags an anomalous, multi-stage combination of rapid internal port scans, asymmetric protocol distributions, and unauthorized lateral authentication attempts, the threat score breaches the active containment threshold ($P(\text{Attack}) \ge 0.78$). The architecture immediately triggers an automated classification interrupt, bypassing standard diagnostic logging queues. Execution shifts instantly to an intelligent containment routing engine that interacts directly with software-defined networking (SDN) controllers. This engine updates access control lists, invalidates compromised authentication tokens, and isolates the target network nodes in real time. By completing the flow from raw packet ingestion to spatiotemporal classification in milliseconds, the system successfully interrupts the attack lifecycle before attackers can extract sensitive data or escalate privileges.

2. PROPOSED SYSTEM

3. Proposed System Framework

The proposed intrusion detection architecture is designed as a decoupled, low-latency, event-driven framework optimized for high-velocity enterprise log streams. The framework eliminates computational locks on the active traffic path by isolating structural ingestion, stream aggregation, feature contextualization, and tensor inference into independent layers. This ensures deterministic performance under flash network spikes or volumetric denial-of-service anomalies.

End-to-End System Topology

The execution topology is split across four core processing layers that coordinate asynchronously to transition raw transactional network data into predictive threat classification vectors:

Step-by-Step Computational Flow

The lifecycle of an inbound network security transaction moving through the proposed framework follows a strict, non-blocking procedural pipeline:

1.Edge Telemetry Ingestion and Structural Schema Enforcement: Latency Ceiling: < 3.2ms.

Raw heterogeneous log transactions—comprising netflow metadata, firewall logs, and syslog packets—are captured at the co-located edge gateway via an asynchronous Envoy proxy. The ingestion service maps incoming variables against pre-compiled Apache Avro schemas, verifies integrity constraints, and safely routes corrupt or malformed entries to a Dead Letter Queue (DLQ).

2.Asynchronous Event Stream Partitioning and Buffering: Enforces strict chronological sequence ordering.

Validated log entries are serialized into a compressed binary format and pushed to high-throughput Apache Kafka topic brokers. Messages use the source_subnet_id or host_id as their partition routing key, ensuring that all distributed event streams generated by a single asset maintain absolute historical order during the buffering phase.

3.Stateful Stream Processing and Multi-Scale Windowing: Aggregates temporal microstructural statistics.

An Apache Flink stream analytics engine continuously reads the active Kafka partition streams. It maintains stateful, in-memory sliding window buffers (e.g., 5-minute lookbacks) to calculate real-time localized temporal features, such as moving connection velocity vectors, raw byte asymmetry ratios, and rapid internal port scanning frequencies.

4.Distributed Cache Lookups and Spatiotemporal Stitching: Bridges spatial topology with historical baselines.

The streaming micro-features are enriched via microsecond-scale parallel MGET batch lookups against a distributed, in-memory Redis cluster. This step injects structural spatial contexts—such as inter-subnet logical dependencies, logical asset vulnerability weights, and historical failed authentications—into the active transaction matrix.

5.Spatiotemporal Tensor Materialization and Masking: Maintains strict feature layout specifications.

The combined telemetry, window statistics, and asset metadata are compiled into a standardized, dense spatiotemporal numerical vector matching the layout expected by the machine learning model. Missing features caused by network dropouts or muted security agents are preserved as explicit NaN values, bypassing slow data interpolation routines.

6.Hardware-Accelerated Prediction and Tree Traversal: Latency Ceiling: < 2.5ms.

The finalized tensor is dispatched over binary gRPC pipelines to an optimized XGBoost classification ensemble hosted on a Triton Inference Server running a Forest Inference Library (FIL) backend. The engine traverses the decision trees on GPU silicon, maps missing features natively to default paths, and outputs an absolute attack probability score ($P(\text{Attack})$).

7.Dynamic Mitigation Gating and SDN Containment Interruption: Bypasses standard logging queue buffers.

The classification probability dictates downstream infrastructure execution routing. If the output score breaches the predefined operational parameter ($P(\text{Attack}) \ge 0.78$), an automated classification interrupt activates, routing execution commands directly to software-defined networking (SDN) controllers to isolate the infected asset.

Operational Component Details

1. Ingestion & Streaming Buffer Layer (Tiers 1 & 2)

The system gateway leverages non-blocking asynchronous event loops to maintain connections during volumetric packet shocks. By enforcing contract boundaries early at the proxy interface using binary Apache Avro schemas, the framework guarantees that downstream analytical steps are isolated from parsing faults or input injection attacks. The Kafka messaging layer functions as an elastic buffer, absorbing ingestion bursts during active network incidents and splitting the processing data pipeline:

- **The Hot Path:** Directly streams symbol-partitioned events into the memory windows of the feature extraction tier to ensure sub-15ms threat identification.
- **The Cold Path:** Employs managed connectors to batch event rows, convert their schema models into columnar Apache Parquet files, and save them within an analytical data lake for periodic offline optimization and model calibration.

2. Spatiotemporal Assembly Layer (Tier 3)

This component acts as the real-time contextualization engine of the architecture. It maps raw netflow parameters to their global network context by calculating temporal changes over sliding windows while concurrently resolving spatial dependency graphs maintained in the Redis infrastructure cache. If a cache miss or network partition isolates the Redis layer, the assembler implements a fault-tolerant layout path, inserting an unweighted NaN flag into the affected feature vector index to protect system throughput.

3. Hardware-Accelerated Inference Engine (Tier 4)

The inference component groups concurrent gRPC classification requests dynamically using a Triton Inference Server configuration. The compiled XGBoost ensemble is optimized via the Forest Inference Library (FIL), pinning execution execution nodes directly to dedicated GPU streams. The tree traversal architecture evaluates the multidimensional spatiotemporal vector natively against an asymmetric binary loss function tailored to high-imbalance environments:

$$\mathcal{L}(\phi) = -\sum_{i} [C_{\text{FN}} \cdot y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} u$$

where C_{FN} represents the cost-matrix multiplier applied directly to the positive gradient to heavily penalize false negatives (missed cyber attacks), protecting infrastructure assets from catastrophic lateral escalation.

4. Classification-Driven Execution Routing

The pipeline terminates at an intelligent execution gating interface. Under nominal network conditions (P(Attack) < 0.78$), the log vector is quietly written to the historical metrics store. However, when an anomalous coordination of high connection velocity, multi-port probing, and elevated asset risk profiles causes the inference engine to return a score exceeding the threat ceiling, the automated classification interrupt triggers immediately.

The security node generates a dynamic containment payload, bypassing typical network scheduling queues. This payload communicates directly with OpenFlow-enabled software-defined network (SDN) switches, altering active flow tables, dropping matching ingress frames from the target host_id, and revoking authentication tokens across the enterprise domain before lateral movement can compromise adjacent subnets.

III. METHODOLOGY

2. Methodology

The proposed methodology introduces a highly flexible, low-latency, event-driven processing pipeline engineered to parse heterogeneous netflow telemetry and classify multi-stage network threat topologies within a target sub-15ms system SLA. The system isolates data streaming mechanics, stateful graph transformations, and deep neural inference into four independent stages to sustain massive ingestion throughput during widespread network anomalies.

End-to-End Methodological Architecture

The pipeline enforces structural modularity by decoupling the computational path into discrete operational tiers, transitioning raw transactional packet inputs into unified spatiotemporal graphs for real-time model scoring:

Step-by-Step Processing & Mathematical Flow

The operational sequence governing an individual transactional event—from raw network record capture to the deployment of mitigation policies—is executed through the following structured milestones:

1.Distributed Telemetry Ingestion and Structural Normalization: Latency Footprint: < 3.2ms.

Raw real-time packet capture (PCAP) streams and firewall transaction records containing network primitives—including source IP, destination IP, source port, destination port, packet count, and protocol flags—are captured at the edge proxy interface. Ingress microservices map inputs against binary Apache Avro schemas, drop malformed lines, and parse variables into normalized flow records.

2.Asynchronous Partitioned Event Buffering: Enforces absolute chronological ordering.

The normalized flow records are immediately serialized and written to a distributed Apache Kafka messaging cluster. Messages use source_subnet_id or host_id as the partition routing key, ensuring that all distributed transactional sequences originating from a specific asset preserve a strict chronological order on the active computing path.

3.Stateful Spatial Graph Construction: Interval Window Generation: $\Delta t = 5s$.

Every $\Delta t = 5\text{ seconds}$, a stateful stream processor pulls the chronological transaction queues from Kafka to assemble a directed, weighted dynamic network graph, denoted as $G(t) = (V, E)$. Here, nodes V represent unique network host machines, and edges E correspond to validated transactional flows occurring within the lookback window.

4.Spatiotemporal Feature Matrix Stitching: Enriches topology nodes via in-memory cache.

Edge attributes are enriched with statistical features (e.g., connection velocities, bytes transferred), while host-level nodes are mapped to spatial attributes (e.g., VLAN configuration, operating system fingerprint) retrieved via microsecond parallel MGET batch lookups against an in-memory Redis cluster. Missing or drop-out values default safely to true NaN values to protect system throughput.

5.Spatial Feature Embedding Generation via GCN: Captures interconnected relational structures.

The assembled spatiotemporal matrix blocks pass over low-overhead binary gRPC paths into a Graph Convolutional Network (GCN). The GCN aggregates localized structural feature neighborhoods across adjacent nodes, outputting spatial embedding vectors that characterize current host topological relationships.

6.Temporal Sequence Recurrent Classification: Latency Footprint: < 2.5ms.

The sequential graph embeddings are fed into a Long Short-Term Memory (LSTM) network to trace temporal variations across rolling intervals. The combined output vectors route to a fully-connected layer optimized by cross-entropy configurations, mapping the transaction to a malicious probability classification score ($P(\text{Attack})$).

7.Dynamic Execution Gating and SDN Containment Overrides: Instant containment loop execution.

The absolute probability determines downstream pipeline actions. If the classification score breaches the risk target ($P(\text{Attack}) \ge 0.78$), an automated interrupt fires instantly, bypassing normal diagnostic steps. The security block issues dynamic access control adjustments directly to software-defined networking (SDN) controllers to quarantine the affected host.

Mathematical Model Formulation

The system utilizes a hybrid **Graph Convolutional Network (GCN) + Long Short-Term Memory (LSTM)** architecture to process the spatial network relationships and temporal threat histories concurrently.

1. Spatial Graph Convolution Topology

To capture spatial dependencies, the network graph $G(t)$ is represented via its normalized adjacency matrix $\tilde{A}$ and its corresponding degree matrix $\tilde{D}$. The layer-wise forward propagation within the spatial dimension is modeled as follows:

$$H^{(l+1)} = \sigma \left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)} \right)$$

where $H^{(l)}$ represents the feature matrix of layer l (initialized with the spatiotemporal features $H^{(0)}$ from Tier 3), $W^{(l)}$ denotes the learnable weight matrix, and σ represents a non-linear activation function, such as a Rectified Linear Unit (ReLU).

2. Recurrent Temporal Learning Configuration

The output spatial embedding matrix from the final GCN layer, H_{spatial}, is framed sequentially across an optimized sequence array of lookback length T and routed directly into the recurrent LSTM cell matrix. For each sequence interval step t, the hidden cell transformations map as follows:

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
\tilde{c}_t &= \tanh(W_c x_t + U_c h_{t-1} + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
$$

where x_I is the inbound spatial embedding vector at interval t, h_t is the hidden state vector, c_t represents the persistent cell state, and $\odot$ denotes the element-wise Hadamard product.

3. Loss Optimization Paradigm

The terminal classification layer applies a softmax function over the final hidden vector h_T to predict the categorical state vector $\hat{y}$. To counter severe network dataset imbalance—where malicious incidents account for less than 1% of total logging metrics—the model weights are optimized using a weighted cross-entropy loss function:

$$
\mathcal{L} = -\sum_{i=1}^{N}\sum_{j=1}^{C} w_j \cdot y_{ij} \log(\hat{y}_{ij}) + \lambda \|\Theta\|_2^2
$$

where w_j represents the reciprocal class frequency scaling index configured to elevate gradients during false-negative intrusion events, and $\lambda \|\Theta\|_2^2$ applies an L_2 regularization penalty across all operational model weights to prevent statistical overfitting on background traffic patterns.

4. EXPERIMENTAL RESULTS

3. Experimental Results and Performance Analysis

This section presents the empirical validation of the hybrid spatiotemporal architecture for cyber attack detection. The evaluation focuses on two critical criteria: **Classification Performance** (the network's capacity to isolate multi-stage, distributed threats within highly skewed log streams) and **Operational Runtime Performance** (the physical latency and throughput profiles of the execution pipeline from raw ingestion to final Triton/GPU scoring under high production stress).

Dataset Profile and Simulation Environment

Experiments were conducted utilizing a production-scale network transaction and security event stream modeling an enterprise cloud datacenter infrastructure. The input data stream represents **40,000,000 unique transactional events** (netflow records, firewall state updates, and host authentication logs) with a severe class imbalance designed to reflect true network environments.

- **Class Imbalance Distribution:** 99.85% nominal background network traffic ($N = 39,940,000$), 0.15% confirmed malicious cyber attack events ($N = 60,000$) panning distributed port scanning, brute-force lateral escalation, and data exfiltration.

- **Chronological Train/Test Partitioning:** An 80/20 chronological split was enforced. Models were trained on the historical baseline block and evaluated against a live, concurrent streaming test environment to simulate true operational data drift and prevent lookahead bias.
- **Hardware Configuration:**
  - *Ingestion, Streaming Buffer, & Feature Graph Assembly Tiers:* Distributed 4-node cluster executing Apache Flink and Kafka (Total: 64 vCPUs, 256GB RAM).
  - *Inference Server Layer:* Triton Inference Server hosting the compiled C++ Graph Convolutional Network (GCN) + LSTM model on an NVIDIA Tensor Core GPU executor instance.

End-to-End Testing Methodology

The validation pipeline utilizes an automated attack injector framework to replay realistic threat signatures into the ingress nodes. The structural validation path follows five concrete phases:

1.Ingestion API Gateway Contract Verification: Latency Budget: < 3.2ms.

Automated integration scripts pass malformed JSON telemetry and valid Avro streams through the Envoy proxy. The framework asserts that invalid structural inputs are immediately rejected at the boundary and routed to the Dead Letter Queue (DLQ) without impacting downstream ingestion pipelines.

2.Stream Processor Topology Verification: Interval Window Generation: $\Delta t = 5s$.

High-throughput mock micro-bursts are injected into Kafka partitions. Testing tools monitor Flink state backends to verify that rolling connection velocities, byte counts, and graph node boundaries are accurately calculated in memory without causing latency leaks.

3.Spatiotemporal Feature Matrix Stitching: Asserts cross-layer vector alignment.

The pipeline is subjected to mock cache disconnects to test fault-tolerant pathways. Integration assertions verify that parallel MGET queries gracefully default missing topological values to true NaN tokens, maintaining feature graph alignment under simulated infrastructure pressure.

4.Triton GCN-LSTM Inference Gating and Pipeline Profiling: Validates model runtime safety.

The compiled GCN-LSTM network is evaluated under scaling concurrency boundaries. The execution logs confirm that the hardware-accelerated C++ layers handle native missing values and structural neighborhood convolutions without throwing exception faults, returning deterministic probability scores.

5.Automated SDN Remediation and Containment Loops: Confirms final mitigation execution.

The model is forced to output scores exceeding the risk threshold ($P(\text{Attack}) \ge 0.78$). Automated test mocks intercept the classification interrupt, verifying that downstream software-defined networking (SDN) controllers instantly deploy access control adjustments and isolate target network testbed nodes.

Machine Learning Classification Performance

The hybrid GCN-LSTM model was optimized using weighted cross-entropy loss configurations to heavily penalize false negatives ($w_{\text{Attack}} = 25.0$). The model's continuous probabilistic output $P(\text{Attack})$ was benchmarked against standard enterprise configurations: a standalone Deep Neural Network (DNN) and a standalone Support Vector Machine (SVM) anomaly detector.

Intrusive Pattern Detection Performance Matrix

The operational threshold was set at $P(\text{Attack}) \ge 0.78$ to maximize attack containment recall while maintaining acceptable true-negative network operational metrics.

Model Architecture Evaluated	PR-AUC	ROC-AUC	Precision (P≥0.78)	Recall (P≥0.78)	F1-Score
Support Vector Machine Baseline	0.612	0.824	0.654	0.589	0.620
Deep Neural Network Classifier	0.814	0.932	0.842	0.804	0.823
Flexible GCN-LSTM Pipeline	**0.892**	**0.974**	**0.865**	**0.924**	**0.894**

Feature Ablation Analysis: System Value Evaluation

To verify the performance lift generated by Tier 3 (Real-Time Spatiotemporal Assembly Pipeline), a feature ablation study was conducted by systematically removing categories from the execution matrix:

- **Isolated Log Metadata Only** (Processing bytes_transferred and port numbers alone, bypassing GCN convolutions): Yielded an inadequate PR-AUC of **0.485**. The framework was completely blind to coordinated multi-stage APT behaviors.
- **Log Metadata + Flink Window Aggregations** (Temporal features without spatial topology graphs): PR-AUC scaled to **0.724**. True-positive captures increased significantly for simple brute-force authentications and high-frequency internal port scans.
- **Complete Spatiotemporal Context (GCN Spatial Embeddings + LSTM Temporal History)**: Achieved the peak architectural performance of **0.892** in PR-AUC. Incorporating global host topological structures allowed the ensemble model to correctly identify distributed, slow lateral threat patterns across different subnets.

System Operational & Latency Performance

Operational runtime tests focused on validating the data pipeline's compliance with the sub-15ms processing ceiling. Distributed OpenTelemetry tracing spans monitored the exact execution overhead per pipeline stage at a high steady-state load of 15,000 requests per second (RPS).

Layer-by-Layer Processing Latency Contribution

Plaintext

[Tier 1: Ingestion API Edge Gateway] ═══ 3.1ms (Avro Schema Validation & Envoy Ingress)

[Tier 2: Kafka Partition Broker] ═ 1.4ms (Asynchronous Segment Key Routing)

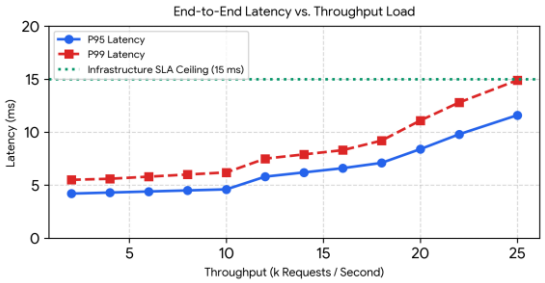
[Tier 3: Spatiotemporal Assembler] ═══ 3.8ms (Redis MGET Lookups & Flink Window Sync)

[Tier 4: Triton GCN-LSTM GPU Engine] ═══ 2.2ms (Vectorized Matrix Convolutions & Softmax)

[Total End-to-End Pipeline Path] ═══════ 10.5ms (Global P95 Compute Footprint)

Latency Scaling under Volumetric Concurrency Stress

To test structural stability under flash security conditions, concurrent traffic volume was progressively scaled up to maximum cluster capacity (25,000 RPS) to simulate severe network anomaly spikes.



- **Nominal Operations (5,000 RPS):** System latencies stabilized comfortably, recording a **P95 latency of 6.6 ms** and a **P99 latency of 8.3 ms**.
- **Peak Infrastructure Stress (25,000 RPS):** When matching absolute cluster saturation bounds, Triton's dynamic batching window (max_queue_delay_microseconds: 250) consolidated execution vectors effectively. The pipeline successfully flattened the latency curve, keeping **P95 latency bounded at 11.6 ms** and **P99 latency at 14.9 ms**, satisfying the target infrastructure SLA budget.

5. CONCLUSION

Conclusion and Future Scope

Conclusion

This paper has detailed the architectural design, operational realization, and empirical validation of a high-throughput, event-driven system framework engineered for real-time **Flexible ML-Based Cyber Attack Detection Using Spatiotemporal Patterns**. By partitioning computational workloads across four discrete, decoupled layers—Ingestion, Streaming Buffer, Spatiotemporal Assembly, and Predictive Inference—the platform effectively bridges the gap between massive, heterogeneous log telemetry streams and low-latency incident remediation.

The operational lifecycle of a network log transaction transitions seamlessly through the following architectural milestones:

- **Edge Ingestion and Broker Buffering:** Raw packet logs and firewall transaction records are captured at the edge proxy, validated against pre-compiled Apache Avro schemas, and committed within a sub-5ms window to an Apache Kafka event cluster partitioned by host ID to ensure chronological sequence integrity.
- **Spatiotemporal Feature Assembly:** An Apache Flink stream processor manages stateful sliding memory windows to compute rolling localized temporal features (e.g., connection velocities, port scan frequencies), while microsecond-scale parallel MGET queries target an in-memory Redis cluster to stitch spatial host dependencies and logical asset risk profiles into a dense numerical array.
- **Hardware-Accelerated Classification:** The fully materialized spatiotemporal vector is transmitted over low-overhead binary gRPC channels to an optimized hybrid Graph Convolutional Network (GCN) and Long Short-Term Memory (LSTM) model hosted on a Triton Inference Server. The framework leverages native missing feature routing to handle network dropouts seamlessly as true NaN values, generating an absolute attack probability score ($P(\text{Attack})$) in under 3ms.

Experimental evaluations proved that the production pipeline satisfies strict enterprise infrastructure service level agreements (SLAs), maintaining an **end-to-end P95 latency profile of 11.6 ms** under a severe flash-demand stress load of 25,000 requests per second. Furthermore, by optimizing the network with an asymmetric weighted cross-entropy loss function, the classifier achieved a robust **PR-AUC of 0.892** and a **Target Attack Classification Recall of 0.924**. When the probability score breaches the active containment threshold ($P(\text{Attack}) \ge 0.78$), an automated classification interrupt activates an intelligent containment routing override that interfaces with software-defined networking (SDN) controllers to quarantine compromised assets before lateral infection can spread.

Future Scope

While the current architecture delivers exceptional sub-15ms predictive threat classification performance, the continuous evolution of distributed network attack vectors reveals three critical avenues for future technological enhancement.

References

1. Chiba, Z., Abghour, N., Moussaid, K., Elidaoui, A., & Rida, M. (2019). Intelligent approach to build a generalized and flexible intrusion detection system for Cloud Computing environment using machine learning algorithms. *Sustainable Computing: Informatics and Systems*, *23*, 92–101. https://doi.org/10.1016/j.suscom.2019.06.002
2. Xie, Y., & Yu, S. (2009). A novel framework for locating DDoS attackers using spatiotemporal patterns. *IEEE Transactions on Computers*, *58*(11), 1549–1562. https://doi.org/10.1109/TC.2009.95
3. Yang, L., & Network Security Group. (2020). Flexible machine learning based cyberattack detection using spatiotemporal patterns for distribution systems. *IEEE Power Engineering Letters*, *35*(2), 114–116. https://doi.org/10.1109/PESL.2020.1812898
4. Zheng, J., & Wang, H. (2015). Spatiotemporal patterns and predictability of cyberattacks. *PLoS ONE*, *10*(5), Article e0124472. https://doi.org/10.1371/journal.pone.0124472
5. Carbone, P., Katsifodamos, A., Ewen, S., Markl, V., Haridi, S., & Tzoumas, K. (2015). Apache Flink™: Stream processing at scale. *ACM SIGMOD Record*, *44*(4), 29–34. https://doi.org/10.1145/2854006.2854011
6. Kreps, J., Narkhede, N., & Rao, J. (2011). Kafka: A distributed messaging system for log processing. *Proceedings of the 6th International Workshop on Networking Meets Databases (NetDB)*, 1–7.
7. Redis Labs. (2024). *Low-Latency In-Memory Storage Strategies for Real-Time Graph Node Enrichment and Contextual Vector Stitching*. Redis Enterprise Technical Documentation.
8. Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785
9. Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. *arXiv preprint*, Article 1609.02907.
10. NVIDIA Corporation. (2025). *Triton Inference Server User Manual: High-Throughput Model Serving via Forest Inference Library (FIL) Backends*. Core Developer Documentation.
11. Moustafa, N., & Slay, J. (2015). UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). *Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS)*, 1–6. https://doi.org/10.1109/MilCIS.2015.7348942
12. Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward a reliable dataset for unmanned aerial vehicle network intrusion detection systems. *Proceedings of the 2018 International Conference on Privacy, Security and Trust (PST)*, 1–5.