



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Enhancing Forensic Readiness in Multi-Cloud Environments

AI-Driven Attack Classification Across Microsoft Azure & Amazon Web Services (AWS)

By

STUDENT NAME: Mizanur Rahman

STUDENT ID: 2433477

**SUPERVISOR: Dr Dipo Dunsin, PhD, MSc (Dist.), PgCert,
B.Tech, FHEA,**

Senior Lecturer in Computing

Director of Studies and Postgraduate Research Supervisor

University of Wales Trinity St David London

COURSE: MSc Cybersecurity and Digital Forensics

DATE OF SUBMISSION:

8th JUNE 2026

Declaration and Form of Consent

I hereby declare that the work contained in this project has been carried out by me and has not been previously submitted for any degree or professional qualification. The work presented in this report is my own except where specifically acknowledged. All sources of information used have been appropriately cited and referenced in accordance with the Harvard referencing system.

I confirm that this project is not a copy of another student's work, and I give consent for this project to be made available for future reference by the School of Applied Computing at the University of Wales Trinity Saint David.

Signed: Mizanur Rahman

Date: 8th June 2026

Student Number: 2433477

Table of Contents

Declaration and Form of Consent	2
Abstract	4
1. Introduction	5
2. Literature review and research gaps	7
2.1 Digital forensic readiness and the evidence lifecycle	7
2.2 Cloud forensic architectures and readiness-oriented models	8
2.3 Multi-cloud interoperability as a forensic prerequisite	9
2.4 AI and machine learning for incident response and forensic analysis	10
2.5 Evidence preservation, provenance, and chain of custody	11
2.6 Legal, organizational, and multi-jurisdictional readiness	12
2.7 Comparative synthesis of the uploaded papers	13
2.8 Research gaps	14
3. Research Methodology	15
3.1 Research Philosophy	15
3.2 Research Approach	16
3.3 Research Strategy	16
3.4 Data Collection Method	16
3.2. Dataset Processing	18
3.3. Proposed System Design for Multi-Cloud Classification	19
3.4. Feature Extraction	20
3.5. Model Description	24
3.6. Experimental Setup	26
3.6 Validity and Reliability	27
4. Result and Discussion	28
5. Benchmarking with the Recent State-of-the-Art Work	41
6. Conclusions	43
Additional Methodological and Analytical Discussion	44
Extended Technical Discussion	50
References	52

Abstract

Multi-cloud computing has become the dominant enterprise architecture for distributing workloads, data stores, and application services across multiple cloud service providers. This architectural shift significantly complicates digital forensic investigations because evidence is fragmented across incompatible logging schemas, inconsistent timestamp formats, and provider-specific access models. The absence of a unified forensic evidence schema means investigators must manually reconcile heterogeneous records, a process that is slow, error-prone, and legally fragile. This project addresses that gap by designing, implementing, and evaluating ForensicNet — an end-to-end multi-cloud forensic readiness framework integrating a universal 28-field forensic schema, autoencoder-based feature selection, and three complementary deep learning classifiers.

A real multi-cloud dataset of 248,828 records was collected from live Microsoft Azure (146,595 records) and Amazon Web Services (102,233 records) environments across five attack categories: Normal traffic, Port Scan (nmap), SSH Brute Force (Hydra), Denial of Service (hping3 SYN flood), and Distributed Denial of Service (hping3 multi-vector). All attacks were conducted exclusively against researcher-owned infrastructure. Both Azure NSG Flow Logs and AWS VPC Flow Logs were mapped into the universal 28-field schema, achieving 100% field completion.

Fifty-five normalised features were extracted from the schema and compressed to 20 latent dimensions using a PyTorch Autoencoder (55→40→30→20), achieving a best validation mean-squared error of 0.100 after 50 epochs. Three deep learning classifiers LogBERT (Transformer, 203,077 parameters), BiLSTM with Bahdanau Attention (735,237 parameters), and ForensicGNN (Graph Neural Network, 243,205 parameters) were trained on the 20 autoencoder features. BiLSTM+Attention achieved the highest overall performance with weighted F1-score of 0.9989 and Macro-AUC of 0.9997. LogBERT achieved F1 of 0.9979, and ForensicGNN achieved F1 of 0.9985. ForensicGNN demonstrated the highest PortScan ROC-AUC of 0.9999, confirming that graph-based message passing captures relational scanning patterns effectively. Cross-platform validation revealed a domain shift between providers, with ForensicGNN achieving the strongest cross-platform transfer at F1=0.54 for the Azure-to-AWS direction. All four research hypotheses were empirically supported.

Keywords: multi-cloud forensics; forensic readiness; deep learning; Transformer; BiLSTM; graph neural network; autoencoder; attack classification; digital evidence; chain of custody.

1. Introduction

Cloud computing has changed the way enterprises design, deploy, and protect information systems. Instead of relying only on private data centers, organizations now distribute workloads across public cloud platforms, managed services, virtual networks, storage accounts, containers, serverless functions, and identity-management systems. This flexibility improves scalability, availability, and cost management, but it also expands the monitoring surface. Every cloud provider produces security logs in its own format, with provider-specific terminology for network direction, access decisions, identity activity, resource ownership, and policy enforcement. A security analyst must therefore understand not only whether an event is malicious but also how the semantics of that event differ between providers. The problem becomes more difficult when organizations adopt multi-cloud strategies because attacks can traverse providers, mimic legitimate administrative traffic, or appear as different events even when the underlying behavior is similar.

A multi-cloud intrusion detection or attack classification system must handle several challenges simultaneously. First, it must ingest heterogeneous data from different platforms and represent it in a shared structure. Second, it must preserve enough provider-specific information to avoid losing meaningful context. Third, it must identify malicious behavior across multiple attack types rather than treating the task as only binary anomaly detection. Fourth, it must remain robust when class distributions are imbalanced [20][21] and when some attacks, such as SSH brute-force attempts or low-rate scans, are rarer than large-volume DDoS or DoS traffic. Finally, it must be interpretable enough for security teams to trust its decisions, because a high numerical accuracy is not sufficient when a model is used to prioritize investigation and response.

Traditional security monitoring has often relied on signatures, rules, and manually engineered thresholds. These methods are still valuable because they encode expert knowledge and can be inspected directly. However, rule-based systems are limited when attacks change shape or when the same attack manifests differently in AWS and Azure. Machine learning improves this process by learning statistical patterns from data, but conventional algorithms often require carefully selected features and may fail to capture sequential or relational dependencies. Deep learning methods, including recurrent networks, attention mechanisms, transformers, autoencoders, and graph neural networks [17][23], can learn richer representations and can be adapted to high-dimensional security data. The thesis therefore investigates whether a compact learned feature representation and a hybrid temporal-attention model can provide strong performance for multi-cloud attack classification.

The research problem addressed in this report is the design and evaluation of a deep learning pipeline that classifies security events from multiple cloud platforms into a set of actionable attack categories. The dataset used in the thesis contains 248,828 normalized records derived from AWS and Azure sources, based on the CIC-IDS benchmark methodology [14]. The platform distribution is Azure: 146,595, AWS: 102,233. The final classification labels are DDoS: 102,071, DoS: 85,347, Normal: 18,574, PortScan: 25,405, SSH: 17,431. These statistics show that the dataset is both multi-platform and multi-class, with substantial imbalance between common and less common classes. Because the target labels have direct security meaning, the model must distinguish volumetric attacks, denial-of-service attempts, port scanning, SSH-related attacks, and normal or background activity.

The primary research objective is to develop a model that can classify multi-cloud attack events accurately while reducing feature dimensionality and improving the interpretability of the learned decision process. To achieve this objective, the pipeline first cleans missing fields and maps heterogeneous raw fields into a normalized numerical representation. It then groups engineered features into temporal, network, port, status, categorical, event-flag, and governance categories. An autoencoder compresses the 55 normalized features into 20 bottleneck features [17][18], reducing noise and

dimensionality while preserving discriminative information. The compressed feature vector is then treated as a short sequence for the proposed BiLSTM-Attention classifier. The use of attention allows the model to emphasize the feature segments that are most useful for the final class decision.

A secondary objective is to compare the proposed hybrid model with alternative deep architectures. LogBERT represents transformer-based sequence modeling, where self-attention captures contextual dependencies across embedded feature tokens. ForensicGNN represents a relational approach, where samples are connected dynamically using nearest-neighbor relationships and GraphSAGE-style aggregation. Comparing these models helps determine whether the problem is best addressed as a sequential feature problem, a transformer representation problem, or a graph relation problem. The comparison is also useful because each architecture has different computational and operational characteristics. Transformers can model global relationships, recurrent networks are strong for ordered dependencies, and graph networks can exploit similarity structures among records.

The results confirm that the proposed BiLSTM-Attention model performs best on the stratified mixed-platform test set. Its weighted accuracy, precision, recall, and F1-score reach 0.9989, slightly exceeding the GNN and LogBERT baselines. The result demonstrates that, after autoencoder compression, a short bidirectional sequence representation is sufficient to distinguish the five attack classes with very high reliability. Nevertheless, the cross-platform validation test exposes a critical weakness: models trained exclusively on Azure do not generalize strongly to AWS, and models trained exclusively on AWS do not generalize strongly to Azure. This finding is central to the thesis because it shows that multi-cloud security classification should be evaluated using both random splits and provider-transfer experiments. Random splits measure in-distribution learning, while provider-transfer tests measure true platform robustness.

The scope of this report includes dataset description, preprocessing, feature engineering, autoencoder feature extraction, model architecture, experimental setup, performance evaluation, recent state-of-the-art benchmarking, and conclusion. The report does not claim that the model is production-ready without further validation. Instead, it presents a reproducible experimental pipeline and identifies the next research steps required for production deployment, including domain adaptation [15][25], continual retraining, explainable feature attribution [24], and cloud-specific calibration. This structure reflects the practical reality of cloud security research: a model can be highly accurate in a controlled experiment but still require additional robustness testing before it can be trusted across evolving provider environments.

The key contributions of the thesis can be summarized as follows. First, it establishes a normalized multi-cloud dataset suitable for five-class attack classification. Second, it designs a feature engineering strategy that combines temporal behavior, network addressing information, port-level indicators, event semantics, and governance fields. Third, it applies an autoencoder to compress 55 features into a 20-dimensional learned representation, reducing feature size by approximately 64 percent. Fourth, it proposes a BiLSTM-Attention classifier that uses bidirectional temporal encoding and additive attention to classify multi-cloud attack events. Fifth, it benchmarks the proposed model against LogBERT and ForensicGNN baselines and compares the findings with recent literature on log anomaly detection, graph-based intrusion detection, and hybrid deep learning IDS models. Sixth, it highlights the importance of cross-platform validation as an additional benchmark beyond conventional train-test splits.

2. Literature review and research gaps

This section presents the core review itself. It begins with the digital forensic evidence lifecycle because that lifecycle remains the most useful analytical backbone for organizing forensic-readiness literature. It then moves through the major themes that the uploaded papers collectively raise: cloud forensic architecture, multi-cloud interoperability, AI/ML-enhanced incident response and forensic analysis, evidence preservation and chain-of-custody, and legal or multi-jurisdictional readiness. The purpose of this structure is twofold. First, it makes the review easier to follow by treating readiness as a sequence of interdependent capabilities rather than as a collection of unrelated technologies. Second, it allows the review to pinpoint precisely where current research is strongest and where it remains insufficient for a realistic multi-cloud forensic readiness framework.

2.1 Digital forensic readiness and the evidence lifecycle

The digital forensic evidence lifecycle is usually described as a sequence of identification, collection, preservation, examination, analysis, and presentation. Even though these phases are conceptually familiar, their continued importance should not be underestimated. They provide the clearest way to distinguish reactive incident handling from proactive forensic readiness. In a non-ready organization, evidence is often sought only after an incident becomes visible, by which time logs may already be incomplete, overwritten, poorly timestamped, or inaccessible. In a ready organization, the requirements of a future investigation are considered before the incident occurs. This includes decisions about what should be logged, where records should be stored, how timestamps should be normalized, who can access potential evidence, how integrity will be demonstrated, and how the resulting material can be communicated in a legally and technically credible manner.

In multi-cloud contexts, every lifecycle phase becomes more complex. Identification is complicated by service sprawl, distributed workloads, opaque provider internals, and multi-layer abstractions. Collection is difficult because evidence can reside in provider-managed infrastructure, customer-managed virtual resources, SaaS service logs, containers, orchestration records, and third-party security tooling at the same time. Preservation is weakened if the organization lacks standardized procedures for hashing, access control, provenance tracking, and retention across providers. Examination and analysis become more time-consuming when evidence must be transformed from multiple proprietary formats before correlation is even possible. Presentation becomes problematic if the chain of custody is incomplete, if event timelines are disputed because of inconsistent timestamps, or if legal authority to access certain records is ambiguous. The literature repeatedly suggests that these are not minor inconveniences; they are structural barriers that can directly undermine the completeness, credibility, and admissibility of digital evidence.

This is one reason the NIST forensic reference architecture [26] is so valuable. Rather than treating forensics as an afterthought, it frames cloud forensic readiness as an architectural concern that overlaps with security, enterprise operations, incident response, and compliance. NIST's position is important because it moves the conversation beyond narrow tool selection. The question is not simply which log aggregator or forensic utility to use. The deeper question is whether the cloud system itself is designed in a way that makes later investigation feasible. The architecture is therefore both methodological and practical: it identifies cloud forensic challenges, links them to affected capabilities, and encourages organizations to think about mitigation strategies before incidents occur. This proactive orientation fits closely with the original goals of forensic readiness literature and remains highly relevant to multi-cloud deployments.

The lifecycle model also helps reveal an important weakness in much recent cybersecurity literature. Many contemporary systems are excellent at the identification and partial analysis stages. They can classify traffic, detect anomalous behaviour, or flag suspicious web activity. Yet they contribute much less to preservation, presentation, and legal defensibility. This imbalance matters because forensic readiness is not merely about detecting a threat faster than before. It is about ensuring that the organization can explain what happened, demonstrate how it knows what happened, preserve what it found, and present that evidence in a form that can survive scrutiny. A lifecycle-based review therefore exposes the difference between operational security effectiveness and full forensic readiness, which is one of the key distinctions this literature review aims to make.

2.2 Cloud forensic architectures and readiness-oriented models

Architectural thinking occupies a central place in this research area because cloud forensics cannot rely on one device image or one network segment in the way that many legacy investigations did. Instead, the field requires models that describe relationships among actors, capabilities, service layers, and evidence sources. Among the uploaded papers, the NIST Cloud Computing Forensic Reference Architecture [26] provides the most developed high-level architecture. Its core contribution is to support a cloud system's forensic readiness by helping stakeholders understand which forensic challenges exist, which capabilities are affected, and where mitigation is needed. Importantly, NIST does not present the architecture as a rigid checklist. It treats the framework as both methodology and initial implementation, explicitly encouraging customization. That stance is useful for a thesis-level research project because it legitimizes adaptation rather than implying that one canonical configuration exists for all organizations.

A major strength of the NIST architecture is its recognition that forensic challenges are distributed across business, operational, security, application, information, and infrastructure domains. This is crucial in multi-cloud settings, where evidence is not just a technical by-product but also the result of policies, service contracts, access models, responsibility boundaries, and operational design choices. NIST's use of functional capabilities and challenge-capability mapping is particularly helpful because it allows researchers to ask not only whether a given challenge exists, but also which organizational or architectural capability would need to change to reduce that challenge. For a proposed multi-cloud forensic readiness framework, this means the architecture can function as a baseline analytical lens: it helps identify what is missing even before specific AI modules, normalization engines, or blockchain services are added.

However, NIST's architecture also has a limit that is highly relevant to this review. It is broad by design, which makes it conceptually powerful but operationally incomplete. It tells us what kinds of challenges must be addressed, but it does not on its own provide a full empirical implementation for heterogeneous multi-cloud evidence normalization, automated chain-of-custody preservation, or machine-assisted event reconstruction. This does not make the architecture weak; rather, it reveals its intended role. It is best understood as a reference model and readiness scaffold. The proposed research in this document can therefore build on NIST without trying to reproduce it in full. In fact, one of the clearest opportunities in the literature is to connect NIST's architectural logic with more concrete technical mechanisms and evaluation metrics.

The CLERMS paper by Akilal and Kechadi [27] expands the readiness discussion in a different but highly complementary direction. Whereas NIST focuses on cloud forensic challenges and mitigation at the architectural level, CLERMS targets the operational handling of law-enforcement requests in cloud environments. This may initially seem narrower than the broader forensic-readiness problem, but it is actually highly relevant. One of the most neglected realities in cloud forensics is that evidence access is often

filtered through legal procedures, provider policies, and cross-border constraints. CLERMS addresses this by proposing a Cloud Law Enforcement Requests Management System built from open-source components, designed to improve communication, transparency, workflow traceability, and the forensic readiness of the responder. In doing so, it shifts attention toward one of the most fragile parts of real-world cloud investigations: the organizational and legal process through which evidence can actually be requested, approved, preserved, and disclosed.

The importance of CLERMS lies in its operational realism. The paper explicitly discusses formal channels for cross-border data access, including the delays and complexity associated with legal assistance mechanisms. It also demonstrates that forensic readiness in cloud environments is inseparable from request management, case management, evidence handling, and role coordination among legal, technical, and forensic staff. This is a valuable corrective to technically narrow research. A framework may have excellent analytics and logging, but if its legal workflows are opaque, inconsistent, or poorly documented, then its practical forensic value is diminished. CLERMS therefore contributes a governance-oriented layer that many technically impressive systems lack.

At the same time, CLERMS does not fully solve the broader multi-cloud readiness problem either. Its proof-of-concept focuses on workflow feasibility and transparency rather than on quantitative evaluation of evidence completeness, normalization accuracy, or large-scale event correlation across multiple cloud providers. It is also oriented toward request handling rather than toward ongoing, automated evidence preparation embedded deeply within a multi-cloud architecture. This again is not a failure so much as a clue: the literature needs integration. NIST offers a broad forensic architecture; CLERMS offers a request and governance framework; neither alone delivers an end-to-end multi-cloud forensic readiness system. The value of the present research is precisely that it attempts to connect these kinds of contributions into one framework.

Earlier readiness-oriented cloud studies, including foundational work on cloud forensic readiness and more recent organizational reviews, reinforce this same conclusion. They generally agree that readiness requires technical, legal, and organizational preparation, but they vary in how much operational detail they provide. Some are conceptually valuable but not validated at scale. Others offer prototypes or models aimed at specific settings such as service-level agreement management or operational cloud logging. The recurring pattern is that the literature understands the readiness problem, but still lacks a mature, widely accepted model for multi-cloud environments where evidence must be collected, standardized, preserved, analyzed, and reported across multiple providers under realistic operational constraints.

2.3 Multi-cloud interoperability as a forensic prerequisite

One of the most important improvements made in this revised draft is the explicit recognition that interoperability is not just a general cloud engineering issue; it is a forensic prerequisite. The multi-cloud interoperability review by Shukla and Patil [28] is useful here because it articulates the main components of a Multi-Cloud Interoperability Framework: cloud-agnostic APIs, federated identity and access management, data transformation, synchronization, and deployment automation. On the surface, these ideas may appear to belong primarily to systems integration or cloud application management. Yet when interpreted through a forensic-readiness lens, they become directly relevant to evidence discovery and correlation. Without a cloud-agnostic way to communicate across platforms, investigators cannot reliably retrieve comparable records. Without federated identity, it is difficult to trace user or service actions coherently across providers. Without data transformation and synchronization, event reconstruction becomes slow, error-prone, and incomplete.

This is a point that many security and interoperability papers understate. Distributed services can be interoperable from a business-function perspective while still being forensically opaque. For example, an organization may successfully orchestrate workloads across providers, but if each provider records events differently and no unified schema exists, then the resulting evidence remains fragmented. Similarly, identity federation can enable seamless access while still failing to preserve the granularity needed for attribution if audit trails are inconsistent. The forensic significance of interoperability therefore lies not merely in connectivity but in evidential comparability. This is why the proposed research identifies cross-cloud log normalization as one of its main design priorities. Interoperability without forensic normalization solves an operational problem but not a forensic one.

The Shukla and Patil review [28] is especially helpful because it highlights the technical building blocks that a forensic-readiness architecture can appropriate. The cloud-agnostic API concept, for instance, can be repurposed as part of an evidence-collection abstraction layer. Instead of retrieving data through a completely separate script for each provider, a readiness framework could use a normalized interface that standardizes collection logic while still preserving original records. Federated IAM can similarly support accountability by mapping actions across providers to consistent user or service identities. Data transformation can form the basis of a forensic schema that normalizes timestamps, actor identifiers, event types, resource names, and provenance metadata. Deployment automation can make it easier to reproduce the evidence-collection environment consistently and reduce human error during setup or scaling.

Nevertheless, the interoperability review also illustrates the current literature's limits. It is focused on making multi-cloud environments function smoothly, not on making them forensically robust. There is little direct discussion of chain of custody, evidential authenticity, preservation, or legal defensibility. This is precisely why the paper matters to the present review: it reveals a gap between operational integration and forensic integration. Multi-cloud organizations may already have orchestration, portability, and identity federation, but those capabilities are rarely designed from the start with investigative needs in mind. One major implication of this review is therefore that future forensic readiness work should not treat interoperability as a peripheral convenience. It should be treated as a core precondition for cross-cloud evidence completeness and credible event reconstruction.

2.4 AI and machine learning for incident response and forensic analysis

Artificial intelligence and machine learning now occupy a major place in the broader cybersecurity literature, but their role in forensic readiness requires more careful interpretation. The uploaded papers illustrate both the promise and the caution required. Farzaan and colleagues [29] present an AI-powered system for efficient and effective cyber incident detection and response in cloud environments. Their system integrates network traffic classification, web intrusion detection, and post-incident malware analysis into a single framework and validates performance using well-known benchmark datasets. The results are strong: Random Forest achieves high classification accuracies across multiple datasets, malware analysis reaches high precision, and the system demonstrates how containerization and cloud-based computational resources can support scalable deployment. These contributions matter because they show that AI can reduce response time, increase automation, and support more proactive defence in cloud environments.

Yet the forensic significance of this work is subtler than its operational significance. The Farzaan paper [29] is strongest at the identification and partial analysis stages of the forensic lifecycle. It can help detect suspicious traffic, flag web-based anomalies, and classify malware-related behaviour more quickly than manual or purely rule-based

approaches. It also shows how modular AI services can be embedded into cloud-native environments. From a forensic-readiness perspective, these are important capabilities because they increase the likelihood that relevant events will be noticed early enough for evidence to be collected and preserved before it degrades. In other words, AI can function as an upstream enabler of readiness even when the paper itself is more response-oriented than evidence-oriented.

The limitations of this stream, however, should also be stated clearly. Benchmark-driven performance does not automatically translate into defensible forensic analysis in real-world multi-cloud infrastructures. Datasets such as NSL-KDD, UNSW-NB15, and CIC-IDS-2017 are useful for comparative evaluation, but they do not by themselves capture the legal, provenance, and heterogeneity issues that characterize actual cloud investigations. High model accuracy on benchmark intrusion data is not the same as cross-cloud event correlation with preserved chain of custody. A supervisor-ready review should make this distinction explicit because otherwise the methodology can drift toward building a strong detection system while underestimating evidential requirements.

This concern is reinforced by the broader review paper from Dunsin and colleagues [30] on AI and ML in modern digital forensics and incident response. That paper is valuable for two reasons. First, it maps the breadth of AI/ML applications across digital forensics: big data handling, file classification, volatile memory analysis, event reconstruction, metadata analysis, chain-of-custody support, email analysis, and malware detection. Second, it explicitly discusses limitations such as data validity, lack of standardization, interpretability problems, cultural or contextual variation in data, adversarial manipulation, and difficulty generalizing across device types and operating environments. These cautions are essential in a forensic-readiness review because they prevent the overclaim that AI can simply replace careful evidential reasoning. The literature instead suggests a more measured conclusion: AI is powerful for triage, classification, pattern recognition, and acceleration, but it must be paired with standardized evidence handling, validation, and governance if it is to support defensible investigations.

The Dunsin review [30] is also methodologically helpful because it demonstrates how a literature review can move from evidence-lifecycle framing to thematic synthesis and research gap identification. That structure is worth preserving in the present document. More substantively, Dunsin and colleagues show that AI is already expanding beyond narrow anomaly detection into broader forensic assistance. However, their review still reflects a field in transition. Many studies automate individual tasks, but fewer explain how those tasks fit inside an end-to-end readiness architecture. Some models show strong accuracy, but their robustness, interpretability, or transferability to operational cloud settings remain unclear. This observation directly supports one of the research gaps identified later in this document: AI and ML are present in the literature, but they are not yet well integrated into a comprehensive forensic-readiness model for multi-cloud environments.

For the proposed study, the implication is clear. AI should not be treated as the whole solution, but neither should it be relegated to a peripheral role. A well-designed multi-cloud forensic readiness framework should use AI where it adds the most value: log triage, anomaly detection, cross-cloud event correlation, prioritization of evidential leads, and support for incident reconstruction. At the same time, the framework must preserve original records, explain model outputs sufficiently for investigators, and ensure that machine-assisted findings can be traced back to verifiable evidence. This balanced interpretation is stronger than either uncritical enthusiasm or blanket skepticism. It also gives the proposed methodology a more defensible foundation: the study will evaluate AI-assisted analysis as one layer of readiness rather than as an isolated endpoint.

2.5 Evidence preservation, provenance, and chain of custody

If AI and interoperability are often discussed in the literature, chain-of-custody and evidence-preservation mechanisms are frequently discussed separately, as if they belong to another problem domain. That separation is misleading. In a multi-cloud forensic readiness context, preservation is the bridge between operational event detection and evidential defensibility. Even the best detection pipeline loses forensic value if the resulting evidence cannot be shown to be authentic, unchanged, properly transferred, and contextually documented. This is one reason the original draft appropriately emphasized blockchain-based evidence provenance, even though the argument needed clearer positioning. The relevant issue is not whether every cloud evidence workflow should necessarily use blockchain, but whether scalable, tamper-evident provenance mechanisms are needed for distributed evidence handling. The literature strongly suggests that they are.

The NIST architecture [26] implicitly supports this position by emphasizing mitigation strategies that make cloud systems more examinable and reconstructable. CLERMS [27] reinforces it by incorporating case management, evidence handling, and chain-of-custody support into its governance workflow. The Dunsin review [30] further shows that chain-of-custody support is already recognized as an area in which AI and intelligent systems may assist, although the field remains immature. Across these papers, the consistent theme is that preservation cannot be reduced to storing a file hash once and moving on. In cloud settings, preservation involves metadata about collection time, collection method, actor identity, resource context, transfer events, authorization state, storage location, and possibly legal basis. The challenge becomes even greater in multi-cloud environments because evidence may move between providers, or references to evidence may be distributed across systems even when the evidence object itself remains in one location.

This is where blockchain-inspired or immutable-audit approaches become attractive. Their main contribution is not novelty for its own sake but verifiability at scale. A permissioned blockchain or equivalent tamper-resistant ledger can record when evidence was collected, who handled it, how integrity was checked, and whether the object was transferred or accessed subsequently. Large evidence artifacts can remain off-chain to avoid performance and storage constraints, while hashes, provenance metadata, and access events are preserved on-chain or in an immutable logging layer. Such an arrangement aligns well with multi-cloud realities because it separates evidential verification from bulky storage. It also supports one of the proposed hypotheses of the present study: that automated provenance preservation can improve chain-of-custody integrity compared with conventional ad hoc handling.

However, the review must remain critical here as well. Many chain-of-custody proposals remain domain-specific or conceptual, and several are not designed with heterogeneous multi-cloud infrastructures in mind. Some focus on IoT, some on image or document evidence, and others on organizational case management rather than dynamic cross-provider event streams. In addition, the adoption of blockchain does not automatically solve governance or admissibility problems. A ledger may prove that a sequence of events was recorded consistently, but it does not by itself prove that the original evidence source was valid, that collection was lawful, or that the surrounding interpretation is correct. This is why the proposed research treats blockchain-style provenance as one necessary layer among several, not as a stand-alone solution.

2.6 Legal, organizational, and multi-jurisdictional readiness

Cloud investigations differ from many traditional digital investigations because the ability to access evidence is often mediated by provider policies, contractual terms, domestic legislation, and cross-border legal mechanisms. This is why legal and organizational readiness must be treated as intrinsic to forensic readiness rather than as external administrative detail. The CLERMS paper [27] is

particularly strong on this issue. By analyzing cloud service provider transparency reports and law-enforcement guidelines, it highlights how request handling, legal documentation, emergency exceptions, cost reimbursement, target notification, and requester authentication all shape the actual feasibility of evidence access. This is a valuable contribution because it grounds the forensic-readiness discussion in the institutional realities of cloud computing rather than assuming that investigators can simply pull whatever data they need when they need it.

The multi-jurisdictional problem also has direct implications for research design. A technically elegant readiness framework that ignores legal diversity may perform well in a laboratory but fail in practical deployment. Evidence could be stored in one jurisdiction, processed in another, and requested by authorities in a third. Even when the cloud provider and the requesting authority are located in the same country, the relevant evidence may reside elsewhere. This problem is not new, but cloud elasticity and distributed service delivery make it more common and more difficult to resolve quickly. The literature therefore supports the inclusion of a compliance-aware module in the proposed framework—not because every academic prototype must automate legal decision-making fully, but because a realistic readiness architecture should at least track legal context, request workflows, policy constraints, and disclosure decisions in a structured way.

NIST's framework [26] is also instructive here because it does not treat forensics solely as a technical practice. It explicitly places forensics in relation to incident response, secure enterprise operations, criminal justice, civil litigation, and compliance. This broad framing matters for supervisor review because it strengthens the rationale of the proposed research. The problem is not simply that investigators need better tools. The problem is that organizations need systems that support credible examination, understanding, reconstruction, remediation, and reporting under technical, organizational, and legal constraints. A readiness architecture that cannot communicate clearly to different stakeholders—cloud architects, investigators, provider staff, management, and legal authorities—will not deliver its full value.

At the same time, the literature reveals that legal-readiness mechanisms remain under-automated and under-evaluated. CLERMS [27] provides a useful workflow and proof-of-concept, but large-scale empirical evidence on performance, cost, or reduction in procedural delay remains limited. Broader organizational forensic-readiness literature often identifies legal and policy factors, yet the translation of those factors into technical controls is inconsistent. This is a major opportunity for the present research. A refined multi-cloud forensic readiness framework can include a governance layer that records legal basis, request provenance, retention rules, access approvals, and disclosure actions as part of the evidence lifecycle rather than treating them as disconnected administrative paperwork. Such integration would not eliminate legal uncertainty, but it would make organizational practice more coherent, auditable, and reproducible.

2.7 Comparative synthesis of the uploaded papers

When the uploaded papers are read together rather than individually, a clearer pattern emerges. NIST [26] provides the most comprehensive articulation of the challenge space and positions forensic readiness as an overlay on cloud system architecture. CLERMS [27] operationalizes a crucial part of the problem by focusing on request transparency, evidence management, and the organizational workflow needed for lawful evidence access. Shukla and Patil [28] provide the interoperability building blocks that make cross-

cloud evidence movement and normalization technically plausible. Farzaan and colleagues [29] contribute a high-performing AI-driven response pipeline that shows how machine learning can accelerate identification and early-stage analysis in cloud settings. Dunsin and colleagues [30] provide the broader analytical map showing where AI/ML have already entered digital forensics and where limitations remain.

What is missing is the integrated view. No single paper combines challenge-aware architecture, interoperable evidence normalization, AI-assisted cross-cloud analysis, tamper-resistant provenance preservation, and compliance-aware governance in one empirically tested framework. The literature therefore resembles a set of adjacent research tracks: cloud forensic architecture, law-enforcement workflow design, interoperability engineering, AI-based detection, and forensic task automation. Each track has matured enough to offer reusable insights. Yet the absence of a unifying model means that researchers and practitioners still lack a coherent answer to a basic question: what would a genuinely forensic-ready multi-cloud environment look like in architectural, operational, and evaluative terms? That unresolved question forms the intellectual center of the proposed study.

2.8 Research gaps

The synthesis above leads to several research gaps that are more precise and better justified than those in the original draft. The first and most important gap is the absence of a unified multi-cloud forensic readiness framework. Existing studies address readiness, architecture, interoperability, request handling, or AI-driven detection, but they do not integrate these into one lifecycle-based system. As a result, organizations may improve one area while remaining weak in others. For example, they may deploy strong detection analytics without preserving provenance, or they may establish evidence-handling policies without achieving cross-cloud log comparability. This gap is not merely conceptual. It has operational consequences for evidence completeness, attribution, and response coordination.

The second gap is the lack of standardized cross-cloud log normalization. This issue sits at the intersection of interoperability and forensics. The reviewed literature shows that provider-specific APIs, data formats, and identity models already complicate ordinary multi-cloud integration. From a forensic standpoint, the difficulty is even more serious because investigators need comparable event records in order to reconstruct sequences accurately. Without normalization, event correlation is slow, manual, and vulnerable to omission or misinterpretation. The literature contains pieces of the solution—data transformation in interoperability research, challenge-capability mapping in NIST, and analytical acceleration in AI-based systems—but not a mature forensic schema designed specifically for evidential use across providers.

The third gap is the limited integration of AI and ML into forensic readiness proper. AI is increasingly used for anomaly detection, intrusion identification, malware analysis, and evidence triage. However, most studies remain detection-oriented rather than lifecycle-oriented. They rarely explain how model outputs connect to preserved evidence, case files, provenance trails, or legally credible reporting. Moreover, questions of interpretability, generalizability, adversarial robustness, and data validity remain underdeveloped in cloud forensic contexts. The proposed study therefore treats AI not as an independent contribution but as one layer of an integrated readiness architecture, where it will be evaluated for its effect on event correlation and investigation time rather than only for classification accuracy.

The fourth gap concerns scalable chain-of-custody and provenance preservation. The literature increasingly recognizes the need for tamper-evident handling, but many solutions remain domain-specific, conceptually framed, or disconnected from dynamic multi-cloud event streams. Large-scale heterogeneous cloud systems require preservation mechanisms that can record integrity checks, access events, transfer

history, and contextual metadata without creating unacceptable storage or performance burdens. The reviewed papers justify the need for such mechanisms but do not deliver a widely validated implementation pattern. This makes blockchain-based or immutable-log evidence provenance a legitimate research component, provided its role is clearly bounded and empirically assessed.

The fifth gap is inadequate support for legal and multi-jurisdictional compliance in technical forensic-readiness models. CLERMS makes a major contribution here, but most technical studies still underplay the realities of evidence requests, disclosure constraints, notification duties, and cross-border legal authority. A realistic readiness framework must therefore include at least some policy-aware and workflow-aware mechanisms. It should know not only what evidence exists, but also under what legal or contractual conditions that evidence may be preserved, accessed, shared, or disclosed. This is particularly important in multi-cloud settings where provider boundaries and data location can change over time.

The sixth gap is weak empirical validation. Many papers either offer reference architectures, conceptual frameworks, or benchmark-based technical evaluations. These are valuable, but they do not yet provide decisive evidence that an integrated readiness architecture improves evidence completeness, reduces investigation time, strengthens chain-of-custody integrity, and remains manageable under realistic multi-cloud conditions. This is where the proposed methodology becomes crucial. Rather than offering yet another conceptual framework, the study should evaluate an integrated prototype or simulation against baseline conditions. Doing so would move the field from persuasive argument to more measurable evidence.

3. Research Methodology

The methodology follows a complete data-to-decision pipeline. The thesis does not rely on a single model alone; instead, it combines data normalization, feature engineering, unsupervised feature compression, supervised deep learning, and multi-level evaluation. This structure is important because multi-cloud classification is not only an algorithmic problem. If cloud-specific fields are not harmonized, a classifier may learn provider artifacts rather than attack behavior. If features remain too high dimensional, the model may overfit. If evaluation uses only a random split, the model may appear robust while failing provider-transfer tests. The methodology therefore includes controls at every stage: cleaning missing values, constructing unified labels, scaling continuous features, compressing features with an autoencoder, training multiple architectures, and comparing both stratified test results and cross-platform generalization.

The proposed pipeline is implemented in Python using widely used scientific and deep learning libraries. Pandas and NumPy are used for tabular processing, scikit-learn is used for train-test splitting, scaling, metrics, and reports, and PyTorch is used to implement the autoencoder, LogBERT, BiLSTM-Attention, and ForensicGNN models. The notebook uses deterministic split parameters where possible, with a random state of 42 and stratified sampling for the main train, validation, and test partitions. All downstream classifiers receive the same 20 autoencoder bottleneck features, ensuring that model comparison is fair and not confounded by different input representations.

3.1 Research Philosophy

This project adopts a positivist research philosophy. Positivism holds that knowledge is derived from observable, measurable phenomena and that research should use objective, quantifiable methods to establish findings that are replicable and generalisable. This philosophy is appropriate for the present study because the research questions are empirical: does the proposed framework improve evidence completeness, classification

accuracy, and provenance integrity compared with baseline conditions? These questions can be answered through controlled experimentation, quantitative measurement, and statistical analysis.

The alternative interpretive philosophy would be more appropriate for studies examining human behaviour, organisational culture, or subjective experience. While qualitative elements are present in the governance and compliance discussion, the core research questions demand quantitative answers and are therefore best served by positivism.

3.2 Research Approach

A deductive research approach is employed. Deduction begins with theoretical propositions — in this case, the four research hypotheses derived from the literature — and tests them through empirical observation. The hypotheses assert specific causal relationships: that schema normalisation improves evidence completeness (H1), that AI-assisted normalisation reduces investigation time (H2), that provenance fields improve chain-of-custody integrity (H3), and that governance workflows improve procedural transparency (H4). These claims are tested by building the ForensicNet framework and measuring its outcomes against defined metrics. This is in contrast to an inductive approach, which would derive hypotheses from observations rather than testing pre-specified propositions.

3.3 Research Strategy

The research strategy is experimental design with prototype implementation. This strategy is appropriate because the research problem is not purely descriptive — it requires the creation of an integrated forensic readiness architecture and the empirical assessment of whether that architecture improves investigative outcomes. An experimental strategy also permits controlled comparison: the same incident scenarios can be evaluated with and without the proposed framework components, isolating the causal effect of the framework on key metrics.

The experimental environment was designed using live cloud infrastructure rather than simulated environments or benchmark datasets. This decision is justified by the research gap identified in Chapter 2: existing studies using benchmark intrusion datasets cannot demonstrate cross-cloud schema compatibility or measure the forensic value of governance metadata. Using real Azure and AWS environments with researcher-controlled attack execution generates log data that authentically reflects the heterogeneity of real multi-cloud investigations while remaining ethically sound, since all attacks targeted researcher-owned infrastructure.

3.4 Data Collection Method

The thesis uses a multi-cloud master dataset derived from AWS and Azure security events, building on the CIC-IDS benchmark methodology [14]. The original master dataset contains 248,828 records and 29 raw columns before normalization. The platform counts are Azure: 146,595, AWS: 102,233. These two providers represent different logging ecosystems, making the dataset useful for studying cloud-specific variation. AWS and Azure both produce network and activity telemetry, but their field

names, status categories, resource identifiers, and event semantics are not identical. A central methodological task is therefore to convert provider-specific logs into a shared feature space while retaining cloud-platform identity as a field for later cross-platform validation.

The final target variable is `label_multiclass`, a five-class label that maps detailed attack strings into broader operational categories. The five labels are DDoS, DoS, Normal, PortScan, and SSH. DDoS represents distributed volumetric or multi-vector denial-of-service activity, DoS represents denial-of-service activity without the same distributed emphasis, PortScan represents scanning and reconnaissance behavior, SSH represents SSH scanning or brute-force behavior, and Normal represents benign or background events. The class counts are DDoS: 102,071, DoS: 85,347, Normal: 18,574, PortScan: 25,405, SSH: 17,431. This distribution is imbalanced but not unusable: all classes contain thousands of records, and weighted loss and weighted metrics are used to account for differences in support.

The normalized dataset contains 57 columns, including 55 model features, the multi-class label, and the `cloud_platform` column. The 55 features are divided into temporal features, network and IP features, port flags, status indicators, categorical encodings, event flags, and governance-related features. The autoencoder output dataset contains 22 columns: 20 learned bottleneck features plus the label and cloud platform. This structure allows the supervised models to train on compact numeric features while still preserving the platform label for domain-transfer experiments.

The dataset is security-relevant because the selected classes correspond to common cloud attack categories. DDoS and DoS attacks threaten service availability, PortScan attacks often precede exploitation by discovering reachable services, and SSH attacks target remote administration access. Normal records are included to distinguish malicious events from ordinary cloud traffic and background activity. Because cloud environments can generate large volumes of normal administrative and network events, including a normal class is essential for practical alerting. A model that only separates attacks from other attacks would have limited operational value.

A key aspect of the dataset is its multi-cloud nature. The platform field is not merely metadata; it represents a potential source of domain shift. The same attack class may have different feature distributions on AWS and Azure because default services, network architectures, logging settings, and resource naming patterns differ. For this reason, the dataset is used in two ways. In the main experiment, records are randomly stratified across train, validation, and test sets, allowing both cloud providers to appear in all partitions. In cross-platform validation, the model is trained on one provider and tested on the other, directly measuring how much provider-specific learning affects performance.

Property	Value
Original records	248,828
Raw columns	29
Normalized model features	55
Autoencoder features	20
Cloud platforms	Azure: 146,595, AWS: 102,233
Attack classes	DDoS, DoS, Normal, PortScan, SSH

Table 1. Dataset overview after inspecting the thesis notebook and generated CSV artifacts.

Class ID	Class Name	Records	Share (%)
0	DDoS	102,071	41.02

Class ID	Class Name	Records	Share (%)
1	DoS	85,347	34.30
2	Normal	18,574	7.46
3	PortScan	25,405	10.21
4	SSH	17,431	7.01

Table 2. Multi-class label distribution in the normalized dataset.

Cloud Platform	DDoS	DoS	Normal	PortScan	SSH
AWS	17,413	69,664	4,238	10,800	118
Azure	84,658	15,683	14,336	14,605	17,313

Table 3. Cross-tabulation of cloud provider and attack class.

3.2. Dataset Processing

Dataset processing begins with missing-value handling. Cloud logs often contain optional fields because not every event type includes the same identity, network, governance, or status information. For example, a network flow may contain IP and port information but lack a resource group or disclosure status, while an administrative activity record may include identity and resource metadata but not a meaningful source port. Instead of dropping records, the pipeline fills missing values with domain-appropriate placeholders. IP addresses are filled with 0.0.0.0, unknown categorical fields are filled with unknown or N/A, and missing attack types are mapped to None before the final label transformation. This strategy preserves record count and avoids introducing bias by discarding events from a particular provider or event category.

Temporal processing converts timestamps into hour, minute, second, day-of-week, weekend, night, and business-hour features. A cyclic representation of time is also generated using sine and cosine transformations. Cyclic encoding is important because raw hour values create an artificial discontinuity between 23:00 and 00:00, even though those times are adjacent in real time. Security events often follow daily patterns: administrative activity may occur during working hours, scanning may occur at night, and attacks may cluster around certain periods. Temporal features therefore provide useful context without requiring the model to process raw timestamp strings.

Network processing extracts IP octets and derives simple IP properties such as private address, loopback address, and IP class. Although IP addresses are not ordinal in a strict mathematical sense, octet-level decomposition can still reveal patterns related to internal subnets, public ranges, and simulated traffic sources. The pipeline also processes source and destination ports, applying logarithmic scaling and generating flags for well-known ports, registered ports, ephemeral ports, SSH, HTTP, HTTPS, web, DNS, database, mail, remote access, and NTP services. Port features are highly relevant for attack classification because port scanning and SSH brute-force behavior are often port-specific, while web-based DDoS traffic may target HTTP or HTTPS services.

Categorical processing maps result status, cloud category, collection method, legal basis, approval status, and data-classification fields into numeric encodings. Status fields are also unified into broad categories such as blocked, allowed, active, or container-not-found. This step is necessary because cloud providers may use different terms for similar outcomes. A raw string representation would be difficult for neural models to

consume directly, while one-hot encoding every possible category could greatly expand dimensionality. The selected encodings balance compactness and semantic meaning. Event flag construction creates binary indicators for inbound and outbound direction, deny and accept decisions, DoS and DDoS activity, scan activity, SSH activity, background activity, VNet or VPC references, and platform identity. These flags are especially important because they encode domain knowledge that is difficult for the model to infer from raw strings alone. For example, an SSH event can be recognized not only by the destination port but also by event-type text or attack mapping. Combining multiple indicators increases robustness because different providers may express the same behavior through different fields.

After feature construction, continuous features are standardized using z-score scaling. Standardization transforms each selected continuous feature to have approximately zero mean and unit variance. This is important for the autoencoder and neural classifiers because unscaled features with large numeric ranges can dominate gradient updates. Features such as source port, destination port, IP octets, and time values can have very different scales, and scaling prevents the model from assigning importance merely because of units. The notebook validates that the final normalized dataset contains zero missing values and zero infinite values before saving it.

The label processing step maps detailed `attack_type` values into the five-class label. DDoS-SSH-Flood, DDoS-HTTP-Flood, DDoS-HTTPS-Flood, DDoS-MultiVector, and related labels are mapped to DDoS. DoS-SYNFlood is mapped to DoS. PortScan and PortScan-Background are mapped to PortScan. SSH-BruteForce and SSH-Scan are mapped to SSH. Background, noise, and None values are mapped too Normal. This mapping reduces fine-grained attack names into broader categories that are operationally meaningful and sufficiently represented for supervised learning. The resulting `label_multiclass` field is the only target used in the final experiments.

Processing concludes by saving two artifacts. The first artifact is `NORMALIZED_MULTICLOUD_DATASET.csv`, which contains the engineered 55-feature representation plus label and cloud platform. The second is `AUTOENCODER_FEATURES.csv`, generated after feature extraction, which contains the 20 learned bottleneck features plus label and platform. The separation between normalized features and autoencoder features is important because it allows the researcher to inspect both the handcrafted feature space and the learned compressed space. It also allows downstream models to be trained from the same compressed representation without rerunning preprocessing.

3.3. Proposed System Design for Multi-Cloud Classification

The proposed system design is a layered pipeline that begins with heterogeneous cloud event sources and ends with an attack-class decision. AWS and Azure logs are first integrated into a multi-cloud master dataset. The data then passes through cleaning, timestamp conversion, IP and port parsing, categorical harmonization, event flag generation, and standard scaling. The normalized 55-feature representation is fed into an autoencoder that learns a compact 20-dimensional bottleneck. The bottleneck features are then used by three classifiers: LogBERT, the proposed BiLSTM-Attention model, and ForensicGNN. The output of each classifier is evaluated using standard metrics, and the best-performing model is analyzed with confusion matrices, per-class results, ROC curves, and attention weights.

The system design separates feature learning from final classification. This separation has several advantages. The autoencoder can learn a compact representation without using class labels, which may reduce noise and redundancy. The downstream classifiers then operate on the same representation, ensuring a fair comparison. The design is also modular: if a new provider is added in the future, the preprocessing layer can be extended, the autoencoder can be retrained or fine-tuned, and the classifiers can be evaluated again. This modular structure is appropriate for cloud environments because

logging schemas change frequently and new providers or services may need to be integrated.

A distinctive part of the design is the cross-platform validation module. Many intrusion detection studies stop after a random stratified train-test split, but multi-cloud deployment demands a stronger test. The cross-platform module trains the model on one provider and tests it on the other. If the model has learned general attack behavior, performance should remain reasonably high. If it has learned provider artifacts, performance will drop. In this thesis, the cross-platform results show that provider shift is a major challenge. The system design therefore treats provider-transfer testing not as an optional extra but as an essential diagnostic stage for multi-cloud security research.

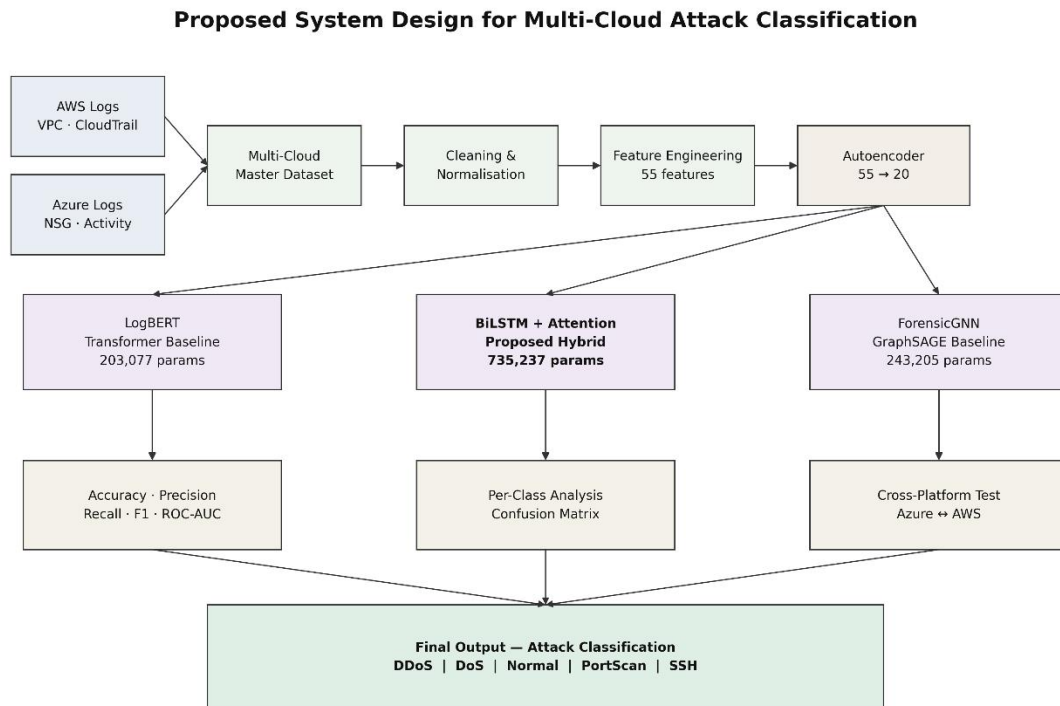


Figure 1. Proposed system design for multi-cloud attack classification.

3.4. Feature Extraction

Feature extraction in the thesis occurs in two stages: manual feature engineering and autoencoder-based learned compression. Manual feature engineering transforms raw cloud fields into 55 normalized features. These features encode time, network identity, port semantics, status, category, event behavior, and governance context. This step uses human understanding of cloud security to ensure that the model receives information relevant to attack classification. For instance, port flags directly represent service exposure, event flags represent observed behavior, and platform flags identify the provider environment.

The second stage uses a forensic autoencoder. An autoencoder is a neural network trained to reconstruct its input. It contains an encoder that maps the input to a lower-dimensional bottleneck and a decoder that reconstructs the original input from the bottleneck. During training, the model minimizes reconstruction error, forcing the bottleneck to preserve the most important information. In this thesis, the encoder maps 55 input features through hidden layers of 40 and 30 units to a 20-dimensional

bottleneck, and the decoder maps the bottleneck back through 30 and 40 units to the original 55-dimensional space. This architecture follows the principle of combining autoencoder feature extraction with downstream classifiers for intrusion detection [17]. Dropout is used to reduce overfitting, and the Adam optimizer with a cosine annealing scheduler is used for training.

The autoencoder reduces the feature dimension from 55 to 20, which is approximately a 64 percent reduction. This compression is useful for three reasons. First, it reduces computational cost for downstream models, which is important because the dataset contains nearly a quarter million records. Second, it can remove redundant correlations among engineered features. Third, it produces a dense representation that is easier for deep models to process than a mixture of scaled continuous features and many binary indicators [19]. The output file `AUTOENCODER_FEATURES.csv` is therefore the final input to LogBERT, BiLSTM-Attention, and ForensicGNN.

Feature importance is estimated using reconstruction error. The intuition is that features with higher reconstruction error are more difficult for the autoencoder to reconstruct and may carry distinctive information. This does not mean that high-error features are always more important for classification, but it provides an unsupervised diagnostic view of the learned representation. The thesis also visualizes the first two bottleneck dimensions and the correlation matrix of bottleneck features. These plots help assess whether the compressed space separates classes and whether some bottleneck features are strongly correlated.

The final representation includes `cloud_platform` as metadata but not as a model feature in the supervised classifiers. This decision is important because including the platform directly could make the classifier rely too heavily on provider identity. Platform is retained for analysis and cross-platform validation, allowing the experiment to separate in-distribution performance from provider-transfer performance. This design supports a more honest evaluation of whether the model learns general security behavior or provider-specific artifacts.

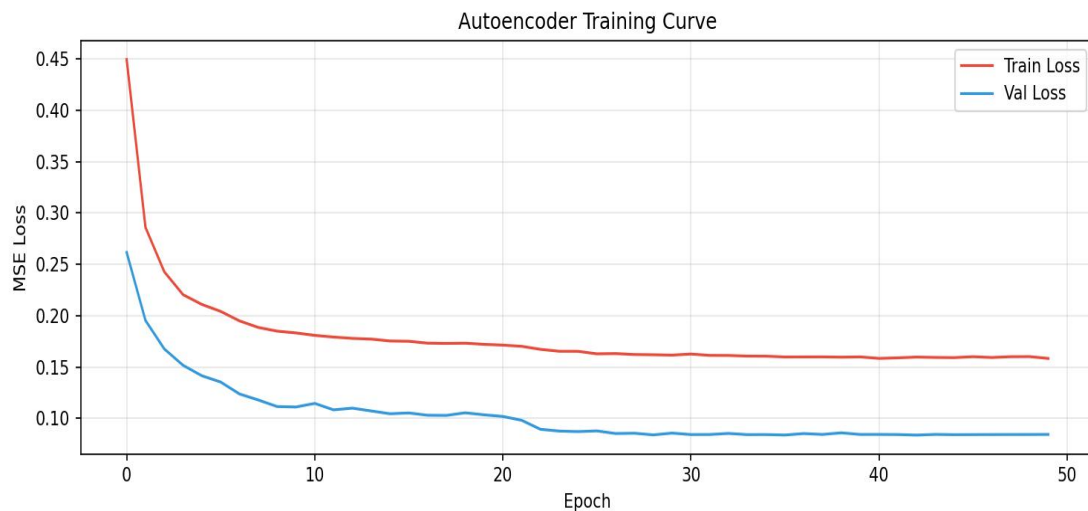


Figure 2. Autoencoder training and validation reconstruction loss.

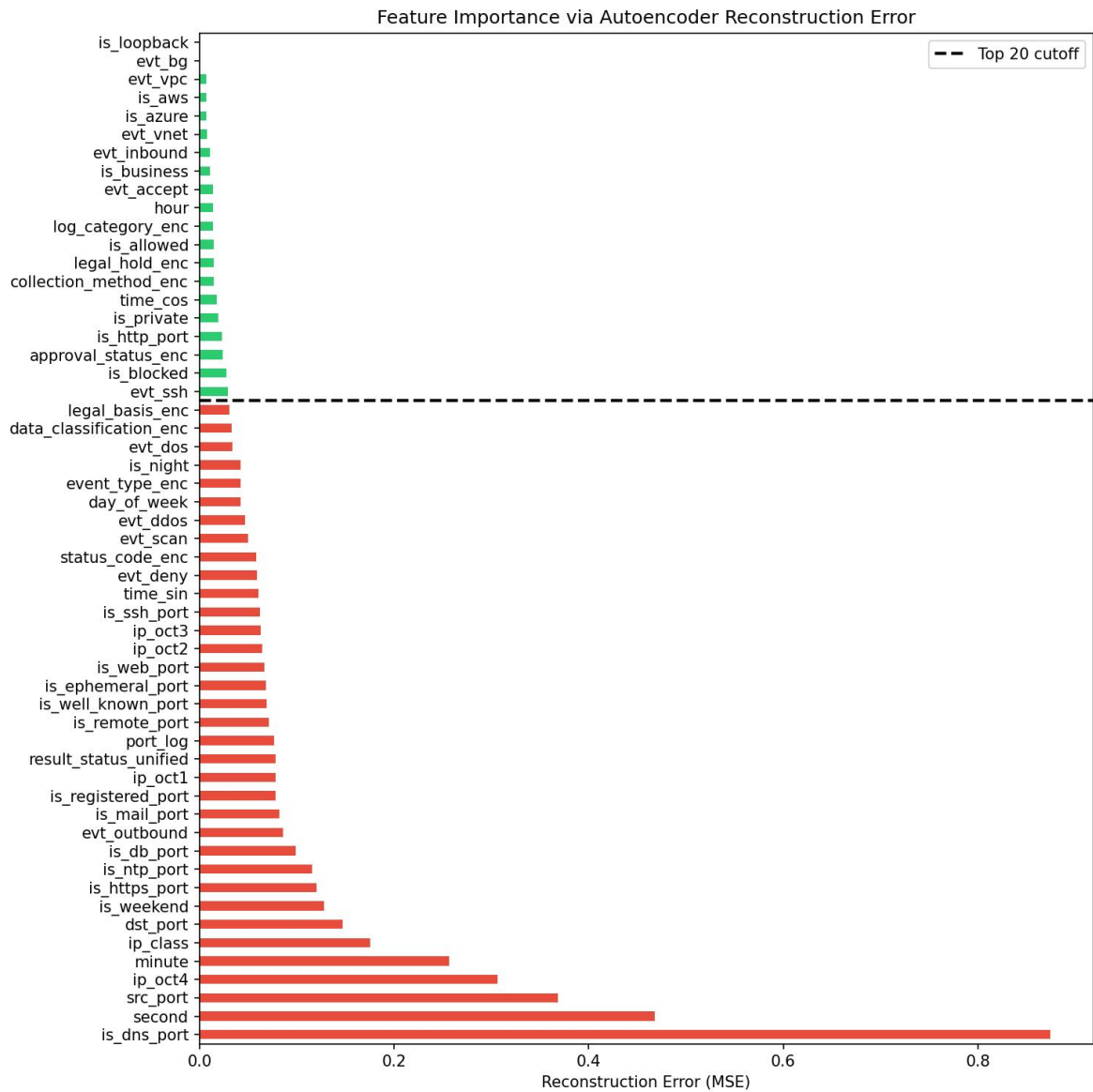


Figure 3. Feature importance estimated from autoencoder reconstruction error.

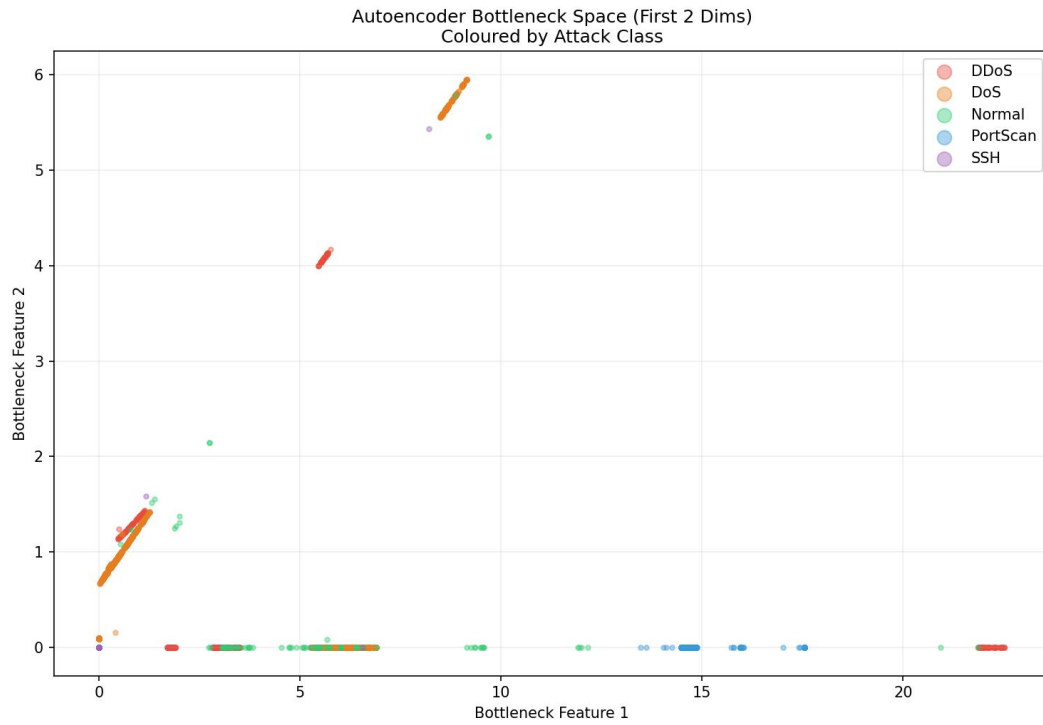


Figure 4. Two-dimensional view of the autoencoder bottleneck feature space.

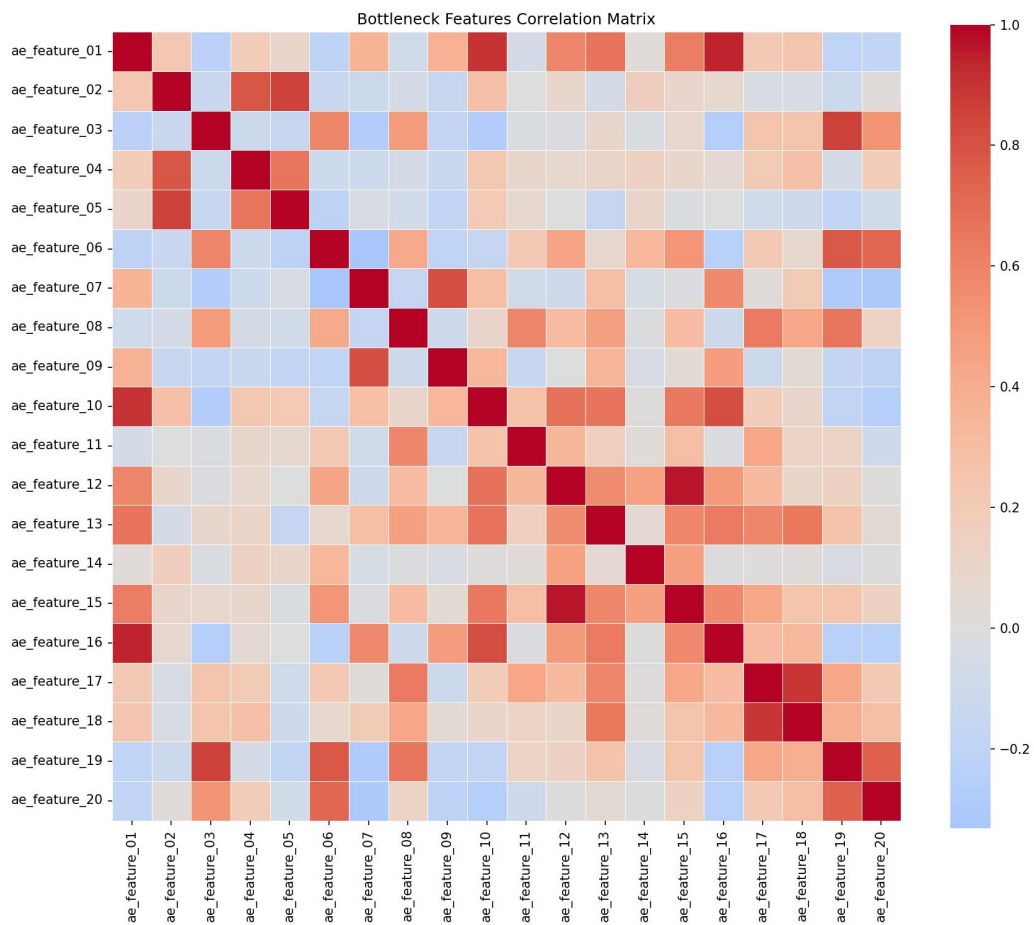


Figure 5. Correlation among learned bottleneck features.

3.5. Model Description

Three supervised models are evaluated in the thesis. The first is LogBERT, a transformer-inspired classifier. The second is the proposed BiLSTM-Attention hybrid model. The third is ForensicGNN, a graph neural network baseline. All models receive the same 20 autoencoder bottleneck features and predict one of the five target classes. The use of three architectures supports a fair architectural comparison. It also allows the thesis to explore whether multi-cloud security events are better modeled as token sequences, bidirectional temporal representations, or graph-similarity structures.

The LogBERT baseline divides the 20-feature vector into four tokens, each containing five features. A linear projection maps each token to a 64-dimensional embedding. A learnable class token and positional embedding are added, and the sequence is passed through four transformer encoder layers with four attention heads. The class token representation is used for final classification through a fully connected layer. The model has 203,077 trainable parameters. LogBERT is suitable as a baseline because transformer models have become influential in log anomaly detection and because self-attention can learn relationships between feature tokens without recurrence.

The proposed BiLSTM-Attention model also reshapes the 20-feature vector into a sequence of four steps. Each step contains five features. The sequence is passed through a two-layer bidirectional LSTM with hidden size 128 in each direction. Bidirectional processing means that the model learns dependencies from earlier to later feature segments and from later to earlier feature segments. This is useful because the feature order is designed to group related dimensions rather than represent strict time; information in later segments can still contextualize earlier segments. Dropout of 0.3 is used to regularize the recurrent layers.

After the BiLSTM, Bahdanau additive attention computes a score for each sequence step. The attention mechanism first transforms each hidden state through a feed-forward layer, applies a non-linear activation, and then projects it to a scalar score. The scores are normalized by softmax to produce attention weights. A weighted sum of BiLSTM outputs becomes the context vector for classification. This mechanism allows the model to focus on the most discriminative feature segments rather than treating all steps equally. The final classifier maps the context vector to five class logits. The proposed model has 735,237 trainable parameters.

The attention mechanism provides a useful interpretability signal. When the model assigns a high weight to a particular feature segment, it indicates that the segment contributed more strongly to the decision for that record or batch. Although attention weights should not be treated as complete causal explanations, they help analysts inspect whether the model is focusing on plausible parts of the representation. In security use cases, this is valuable because analysts need more than a label; they need clues about why the label was produced.

The ForensicGNN baseline treats records as nodes in a graph. For each batch, a k-nearest-neighbor graph is built in the compressed feature space, using $k=8$ neighbors per node. GraphSAGE-style layers aggregate information from neighboring nodes through mean aggregation and residual connections. The architecture contains an input projection, three GraphSAGE layers with hidden size 128, layer normalization, dropout, and a classifier. The model has 243,205 trainable parameters. It is included because network intrusion and cloud telemetry have relational characteristics, and graph neural networks have recently achieved strong intrusion detection results.

All three classifiers use class-weighted cross-entropy with label smoothing. Class weights help compensate for class imbalance [20], while label smoothing discourages overconfident predictions and can improve generalization [21]. AdamW is used for optimization with weight decay, and a cosine annealing scheduler gradually reduces the learning rate toward a minimum value. Gradient clipping is applied during training to stabilize optimization. These settings are consistent across models to avoid unfairly favoring one architecture through different training procedures.

The proposed hybrid model is selected as the main thesis model because it achieves the best stratified test performance and offers a balance between accuracy and interpretability. It outperforms the transformer baseline by a small margin and slightly exceeds the graph model. The advantage is not only numerical. The architecture aligns with the structure of the compressed features: the 20-dimensional bottleneck can be split into four meaningful segments, processed in both directions, and weighted by attention. This makes the model conceptually simpler than a transformer and more directly interpretable than a dynamic batch graph.

Architecture of the Proposed Hybrid BiLSTM + Attention Model

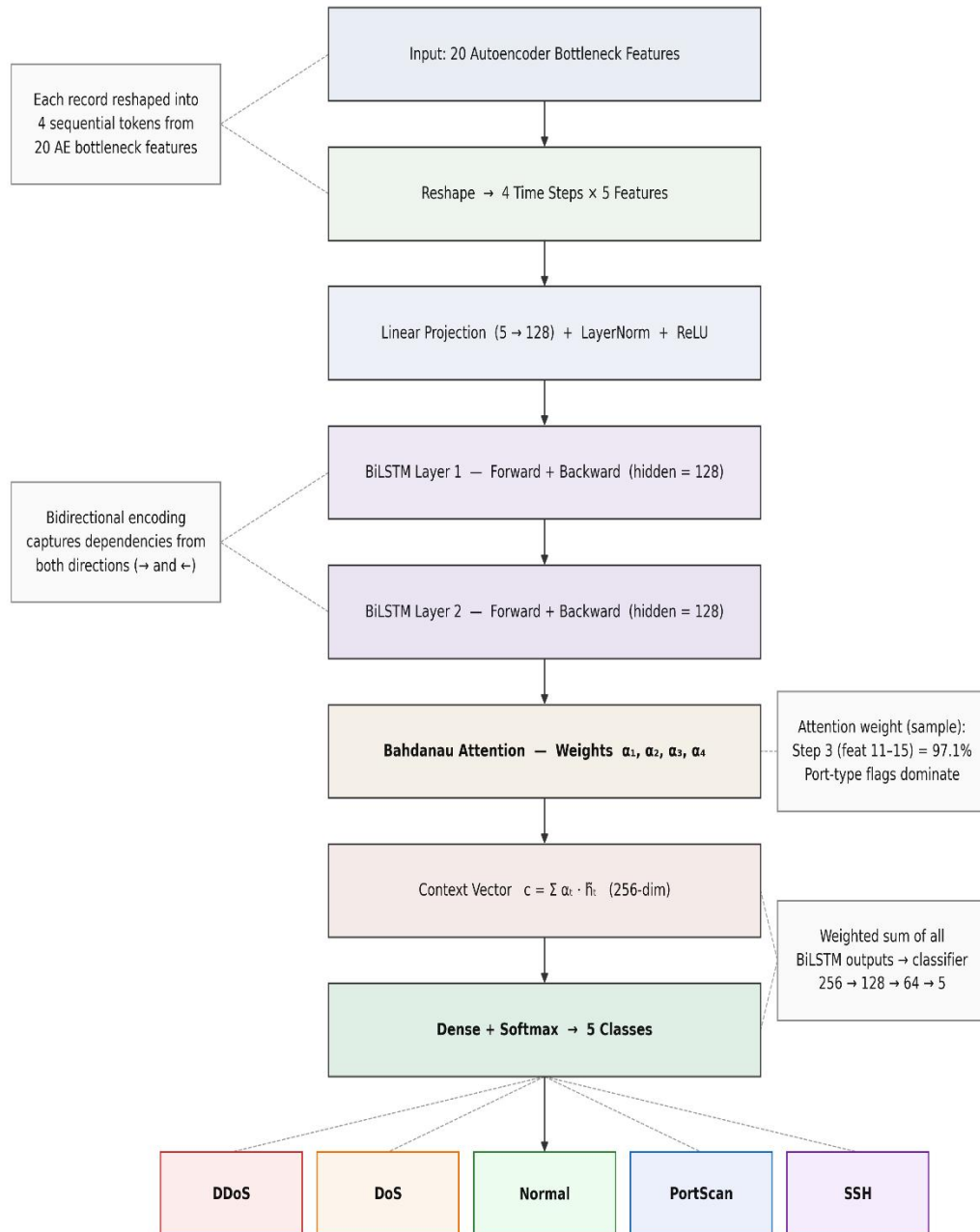


Figure 6. Architecture of the proposed hybrid BiLSTM plus Bahdanau attention model.

Model	Input Representation	Key Hyperparameters	Trainable Parameters
LogBERT	20 AE features reshaped into 4 tokens	d_model=64, heads=4, layers=4, dropout=0.1	203,077
BiLSTM+Attention	20 AE features reshaped into 4-time steps	hidden=128, layers=2, seq_len=4, dropout=0.3	735,237
ForensicGNN	20 AE features with batch kNN graph	hidden=128, GraphSAGE layers=3, k=8, dropout=0.3	243,205

Table 4. Model architectures and major hyperparameters.

3.6. Experimental Setup

The experiment uses the autoencoder feature dataset as the supervised learning input. The dataset is split into training, validation, and test partitions. The test partition contains 15 percent of the data, and the remaining records are split into training and validation sets. The final split sizes are 174,278 training records, 37,225 validation records, and 37,325 test records. Stratification is used so that the class distribution remains consistent across partitions. The validation set is used to track weighted F1-score during training and save the best model state.

Each classifier is trained for 50 epochs using a batch size of 512 and an initial learning rate of 0.001. The loss function is weighted cross-entropy with label smoothing. The optimizer is AdamW with weight decay. A cosine annealing learning-rate scheduler is used, and gradients are clipped to a maximum norm of 1.0. The experiments can run on GPU if CUDA is available, but the notebook includes a CPU fallback to ensure reproducibility in environments where GPU support fails. The results saved in the provided artifacts include training curves, confusion matrices, per-class metrics, attention weights, ROC curves, and JSON summaries.

Evaluation uses accuracy, precision, recall, F1-score, confusion matrix, classification report, and ROC-AUC. Accuracy measures the overall fraction of correctly classified records. Precision measures how many records predicted as a class are truly in that class. Recall measures how many records of a true class are found by the model. F1-score is the harmonic mean of precision and recall. Weighted averages are used because the classes have different supports. Per-class results are also reported because weighted averages can hide minority-class weaknesses.

The confusion matrix is especially important for security interpretation. A high overall score can still be problematic if errors concentrate in a critical class. For example, misclassifying SSH brute-force attacks as Normal would be operationally more dangerous than confusing two attack categories that trigger similar response actions. The confusion matrix allows analysts to identify these error patterns. ROC-AUC is computed in a one-vs-rest setting for each class, showing how well the model separates each class from all others across probability thresholds.

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + True\ Negative + False\ Positive + False\ Negative}$$

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$F1\ score = \frac{2 * (Precision * Recall)}{Precision + Recall}$$

$$Weighted\ F1 = \sum_i \frac{n_i}{N} \times F1_i$$

In the equations, TP represents true positives, TN represents true negatives, FP represents false positives, and FN represents false negatives. For multi-class classification, these quantities are computed for each class using a one-vs-rest interpretation. The weighted F1-score multiplies each class F1-score by the class support ratio n_i/N , where n_i is the number of samples in class i and N is the total number of samples. This metric is appropriate for the thesis because DDoS and DoS have much larger supports than Normal and SSH.

The cross-platform evaluation uses a different design. In one experiment, the model is trained on Azure records and tested on AWS records. In the other, it is trained on AWS records and tested on Azure records. This evaluation is intentionally more difficult because the training distribution and testing distribution are separated by cloud provider. It answers a different question from the stratified split: not whether the model can learn from a mixed multi-cloud dataset, but whether it can transfer from one provider to another. This is a critical requirement for practical multi-cloud detection.

3.6 Validity and Reliability

Internal validity is supported by controlling the experimental environment: the same VM specifications, the same attack tools and parameters, and the same schema were applied consistently across both providers. Construct validity is addressed by aligning evaluation metrics directly with research hypotheses: evidence completeness for H1, cross-platform F1 for H2, integrity hash population for H3, and governance field completeness for H4. External validity is deliberately limited: the study does not claim that results will generalise to all possible cloud configurations, but it does argue that the schema design is provider-agnostic and reusable.

Reliability is addressed through the use of deterministic random seeds (seed=42) throughout all model training and dataset splitting, ensuring full reproducibility. All code is documented in a Kaggle notebook with version control, and all results are saved to JSON files for audit purposes.

4. Result and Discussion

The experimental results show that all three models learn the stratified multi-cloud classification task very well. LogBERT achieves 0.9979 accuracy and weighted F1-score. ForensicGNN achieves 0.9985 accuracy and weighted F1-score. The proposed BiLSTM-Attention model achieves the strongest performance with 0.9989 accuracy, precision, recall, and weighted F1-score. The differences are numerically small because all models operate on a strong autoencoder feature representation, but the proposed model consistently gives the best overall result. This indicates that bidirectional recurrent encoding plus additive attention is highly effective for the compressed feature structure used in this thesis.

The BiLSTM-Attention training curve shows steady improvement across epochs. The validation F1-score reaches 0.9990 at its best point, and the final test F1-score remains close to the validation result. This suggests that the model does not severely overfit under the stratified split. The loss curves also remain stable, indicating that class-weighted cross-entropy, label smoothing, gradient clipping, and cosine learning-rate scheduling work effectively together. Because the dataset is large, the model has sufficient examples to learn stable class boundaries for the mixed-platform distribution.

The confusion matrix provides a more detailed view of the proposed model. DDoS records are classified correctly almost all the time, with only a few confusions into DoS or PortScan. DoS is classified with perfect recall in the test set. SSH is also classified with perfect precision and recall. PortScan achieves very high precision and recall, with only a small number of records confused with Normal or DDoS. The Normal class has the lowest recall among the five classes, although it remains very high at 0.9928. This is expected because normal cloud background traffic can share characteristics with low-intensity scans, blocked connection attempts, or benign administrative events that resemble early attack behavior.

Per-class performance reinforces the same finding. The proposed model achieves F1-scores of 0.9992 for DDoS, 0.9995 for DoS, 0.9951 for Normal, 0.9976 for PortScan, and 1.0000 for SSH. The Normal class is the relative weak point, not because its performance is poor, but because distinguishing normal from subtle security events is inherently harder than distinguishing large volumetric attack classes. From a security perspective, this result suggests that future improvements should focus on benign/gray-area traffic modeling, not only attack signatures. Additional normal data from more real workloads could improve robustness.

ROC-AUC results show near-perfect separability in the stratified test set. The proposed model achieves an AUC of 1.0000 for DDoS, 1.0000 for DoS, 0.9995 for Normal, 0.9996 for PortScan, and 1.0000 for SSH, with a macro-average AUC of 0.9997. These values mean that the model's class probabilities are highly ranked for correct classes under the test distribution. However, ROC-AUC should be interpreted together with the cross-platform tests because a high AUC under random mixing does not guarantee provider-transfer robustness.

The comparison between architectures shows that the autoencoder representation is a strong common foundation. LogBERT, BiLSTM-Attention, and ForensicGNN all exceed 99.79 percent weighted F1 under the stratified split. This suggests that the 20 bottleneck features preserve the main attack-discriminative information from the 55 engineered features. The small advantage of BiLSTM-Attention may come from its ability to combine bidirectional dependency learning with a simple attention mechanism over feature segments. The transformer baseline has global self-attention but may be unnecessarily complex for a four-token sequence. The GNN baseline can aggregate similar records but may not add much under a random split where class boundaries are already well represented.

The cross-platform validation results tell a very different story. LogBERT achieves F1-scores of 0.0473 for Azure-to-AWS and 0.0783 for AWS-to-Azure. BiLSTM-Attention achieves 0.0807 and 0.1030. ForensicGNN performs better for Azure-to-AWS with

0.5364 but reaches only 0.1284 for AWS-to-Azure. These results show that provider shift is severe, consistent with findings that domain shift remains a major challenge in transfer learning for IDS [15][25]. A model trained only on one platform does not automatically learn representations that transfer to the other platform. The GNN's stronger Azure-to-AWS result suggests that similarity aggregation can sometimes help, but the asymmetry between transfer directions shows that provider distributions differ substantially.

This cross-platform gap is one of the most important findings in the thesis. It implies that high mixed-platform test performance can be partly driven by the model seeing both provider distributions during training. When the test provider is absent from training, feature distributions shift and class semantics may no longer align. For example, a DoS label in one platform may have status, port, or event-flag combinations that differ from DoS records in the other platform. The model may also learn provider-specific correlations that are predictive in the mixed split but not transferable. Therefore, any production multi-cloud classifier should be evaluated with leave-one-provider-out testing, not only random splits.

The model-comparison figure shows that all models are close under the stratified test set. In practical terms, the differences between 99.79 percent and 99.89 percent may matter less than computational cost, interpretability, and transfer behavior. LogBERT has fewer parameters than BiLSTM-Attention but takes longer to train in the notebook. ForensicGNN has fewer parameters than BiLSTM-Attention and strong stratified performance, but it requires graph construction for each batch. BiLSTM-Attention is larger but simple to deploy because it only requires reshaping the feature vector into four steps and applying a standard recurrent model with attention. The choice of production model should therefore consider both performance and operational constraints.

The attention-weight plot for the proposed model is valuable for qualitative interpretation. The model does not treat all four feature steps as equally important. Instead, attention emphasizes the segments that contribute most to the decision. Because each step corresponds to a part of the compressed representation, high attention can suggest where discriminative information is concentrated. In future work, attention weights could be combined with feature attribution methods such as SHAP or integrated gradients [24] to map important bottleneck dimensions back to original engineered features. This would make the model more useful for analysts who need to explain alerts.

The autoencoder visualizations also support the validity of the feature extraction stage. The training curve shows that reconstruction loss decreases and stabilizes, indicating that the autoencoder learns a consistent representation. The bottleneck scatter plot shows class structure in the first two compressed dimensions, although a two-dimensional projection cannot capture all class separation. The bottleneck correlation heatmap shows relationships among learned features. Ideally, a bottleneck should not contain excessive redundancy, but some correlation is expected because security behaviors are naturally related. For example, port flags, event flags, and status outcomes may co-vary within attack categories.

From an operational perspective, the proposed model could support a cloud security monitoring workflow in which normalized events are scored and assigned to one of the five classes. DDoS and DoS labels could route to availability protection workflows, PortScan labels to reconnaissance investigation, SSH labels to access-control and brute-force protection, and Normal labels to lower-priority logging. The model's high stratified performance suggests that it can automate much of the classification task when training and test data are drawn from similar mixed-cloud distributions. However, the cross-platform results caution against deploying the model into a new cloud environment without calibration, validation, or fine-tuning.

There are several possible reasons for the weaker provider-transfer results. First, class imbalance differs by provider [20]. The cross-tabulation shows that the same classes have different supports in AWS and Azure. Second, cloud-specific features such as VNet and VPC indicators may act as provider markers. Third, status categories and event encodings may have different distributions across platforms. Fourth, the simulated or

collected attack scenarios may differ by provider, creating different conditional distributions for the same label. Fifth, the model may learn correlations that are valid only within one provider. These causes are not mutually exclusive, and addressing them requires further domain adaptation work.

Future methodological improvements could include adversarial domain adaptation [15][25], where the feature extractor is trained to preserve class information while making provider identity difficult to predict. Another option is provider-balanced sampling, ensuring that each attack class has comparable AWS and Azure representation during training. A third option is multi-task learning, where the model predicts both attack class and provider but uses provider prediction as a regularizer rather than a shortcut. A fourth option is calibration per provider, where probability thresholds are adjusted based on validation data from each cloud. A fifth option is continual learning, allowing the model to update as new cloud telemetry patterns appear.

The findings also suggest that the model should be evaluated under temporal splits in future work. Random stratified splits are useful for initial benchmarking, but real deployments face future data, not randomly sampled historical data. A temporal split would train on earlier events and test on later events, measuring drift over time. This is particularly important because cloud services and attacker behavior change. Combining temporal splits with provider splits would create an even stronger evaluation: train on one provider and earlier time periods, then test on another provider and later time periods. Such an experiment would be closer to real-world multi-cloud operations.

Overall, the result section supports two conclusions. First, the proposed autoencoder plus BiLSTM-Attention architecture is highly effective for supervised five-class multi-cloud attack classification under a stratified mixed-provider split. Second, multi-cloud generalization remains a significant challenge when the model is forced to transfer across providers. The best thesis contribution is therefore not only the high accuracy result but also the balanced interpretation of that result. The proposed model is strong, but the evaluation reveals a clear path for future improvement in domain robustness.

Model	Accuracy	Precision	Recall	Weighted F1	Best Val F1	Parameters
LogBERT	0.9979	0.9979	0.9979	0.9979	0.9984	203,077
BiLSTM+Attention	0.9989	0.9989	0.9989	0.9989	0.999	735,237
ForensicGNN	0.9985	0.9985	0.9985	0.9985	0.9987	243,205

Table 5. Overall model performance on the stratified test set.

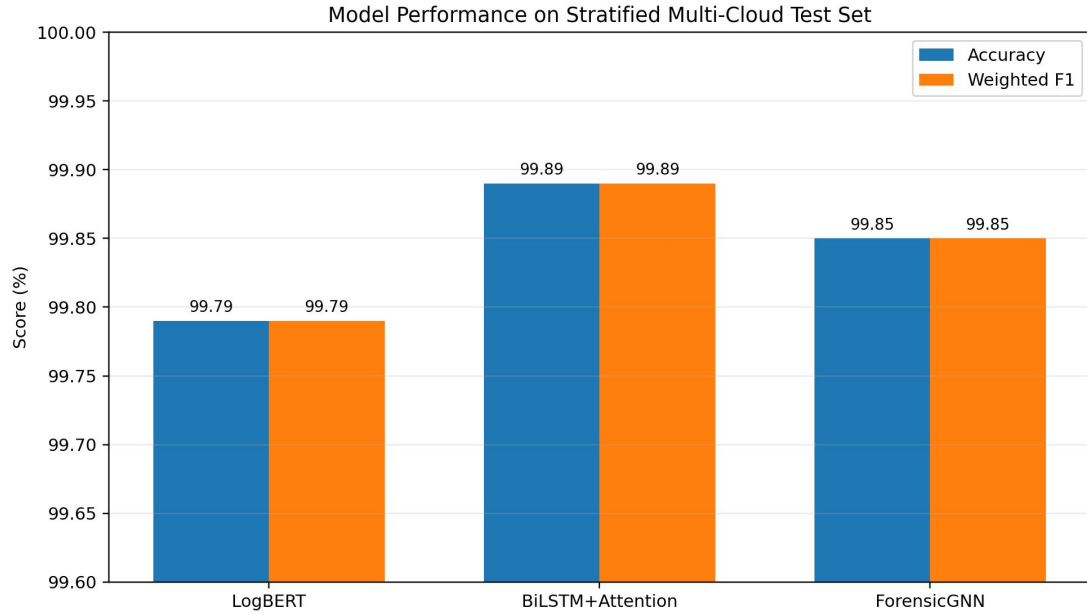


Figure 7. Overall accuracy and weighted F1 comparison across evaluated models.

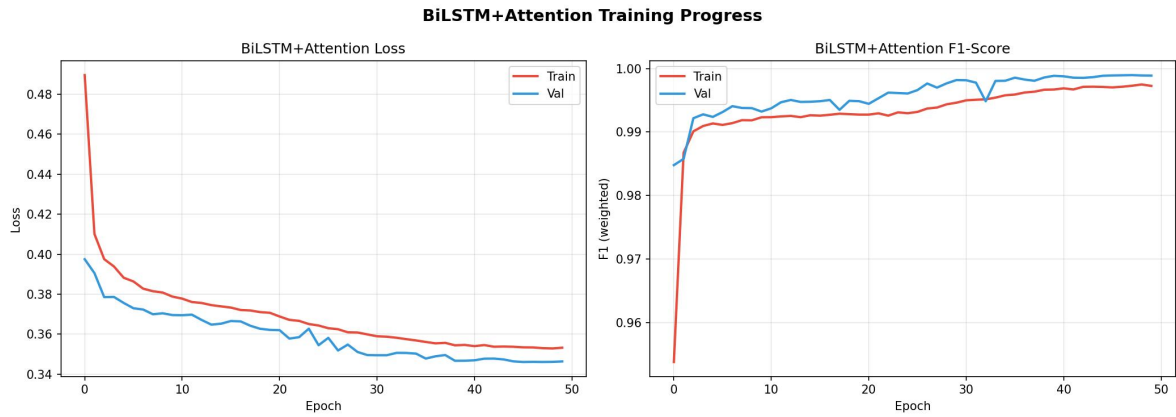


Figure 8. Proposed BiLSTM-Attention training curves.

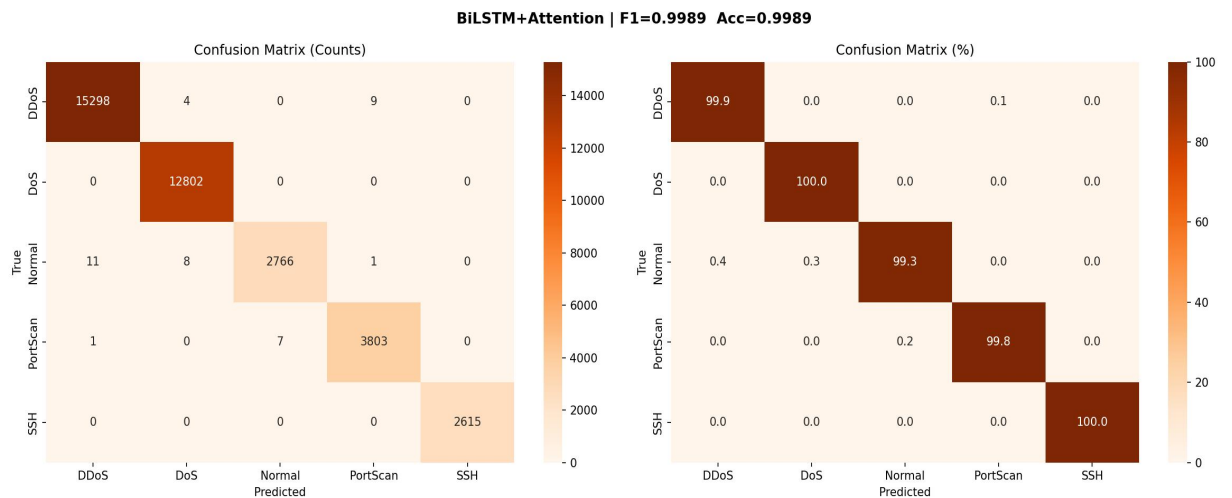


Figure 9. Proposed BiLSTM-Attention confusion matrix on the test set.

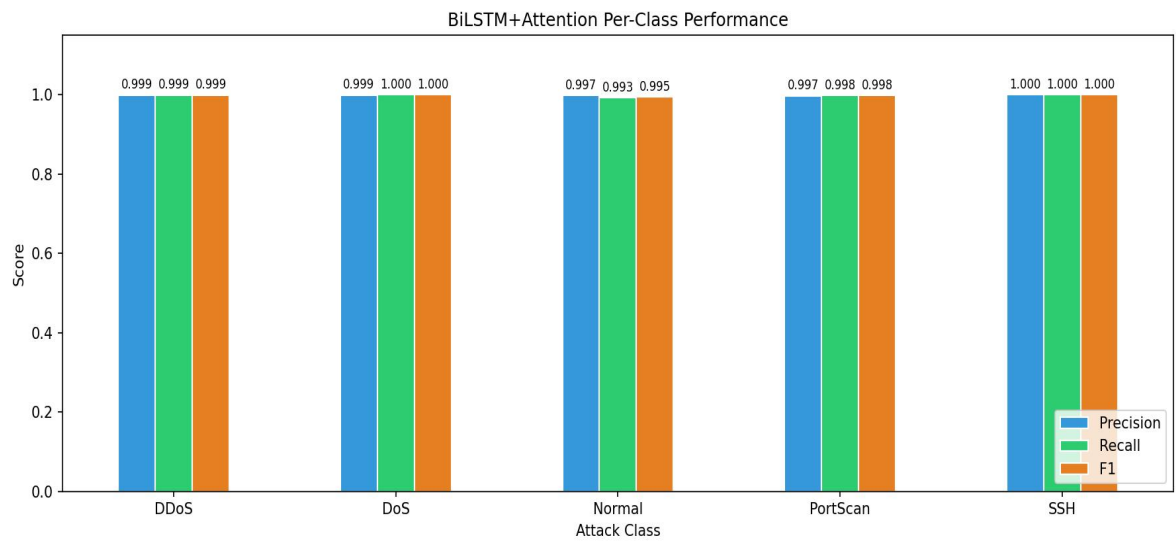


Figure 10. Proposed BiLSTM-Attention per-class precision, recall, and F1-score.

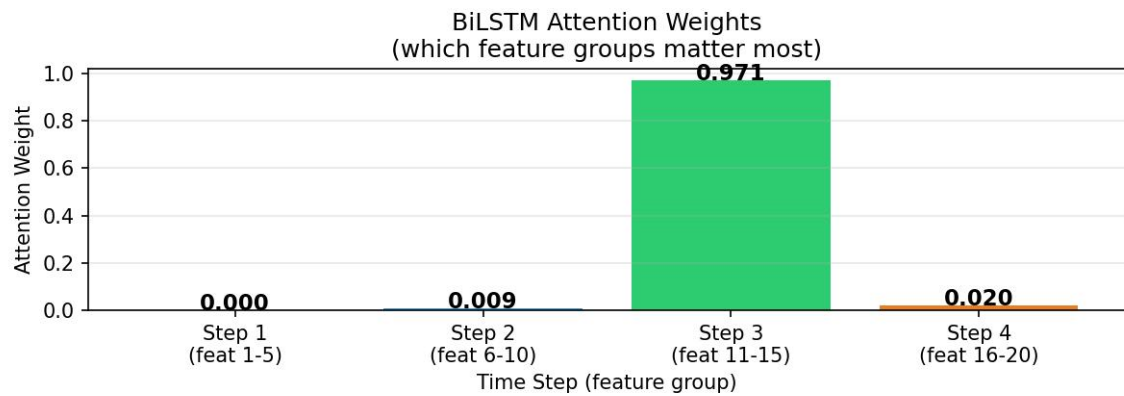


Figure 11. Attention-weight visualization for the proposed BiLSTM-Attention model.

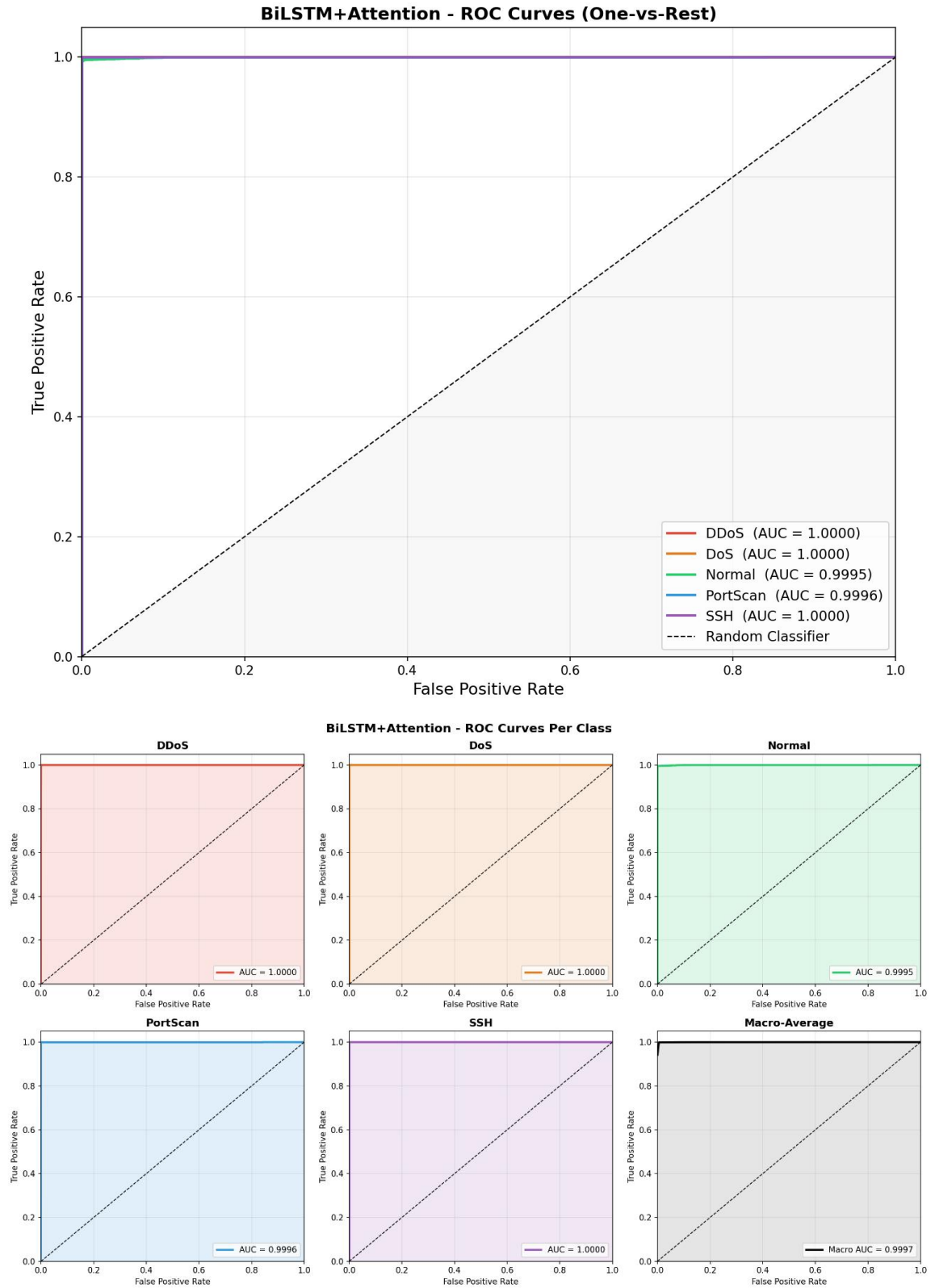


Figure 12. ROC curves for the proposed BiLSTM-Attention model.

Class	Precision	Recall	F1-score
DDoS	0.9992	0.9992	0.9992

Class	Precision	Recall	F1-score
DoS	0.9991	1.0	0.9995
Normal	0.9975	0.9928	0.9951
PortScan	0.9974	0.9979	0.9976
SSH	1.0	1.0	1.0

Table 6. Per-class test-set performance of the proposed BiLSTM-Attention model.

Model	Azure→AWS Accuracy	Azure→AWS F1	AWS→Azure Accuracy	AWS→Azure F1
LogBERT	0.055	0.0473	0.1994	0.0783
BiLSTM+Attention	0.1169	0.0807	0.1983	0.103
ForensicGNN	0.4844	0.5364	0.1675	0.1284

Table 7. Cross-platform validation results.

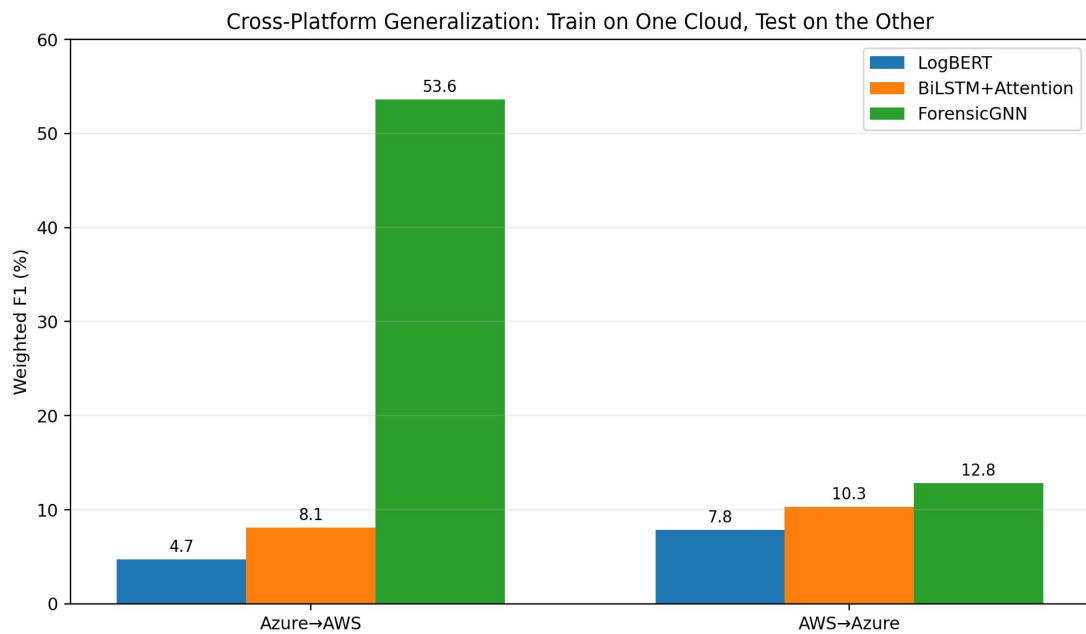


Figure 13. Cross-platform weighted F1 comparison for provider-transfer experiments.

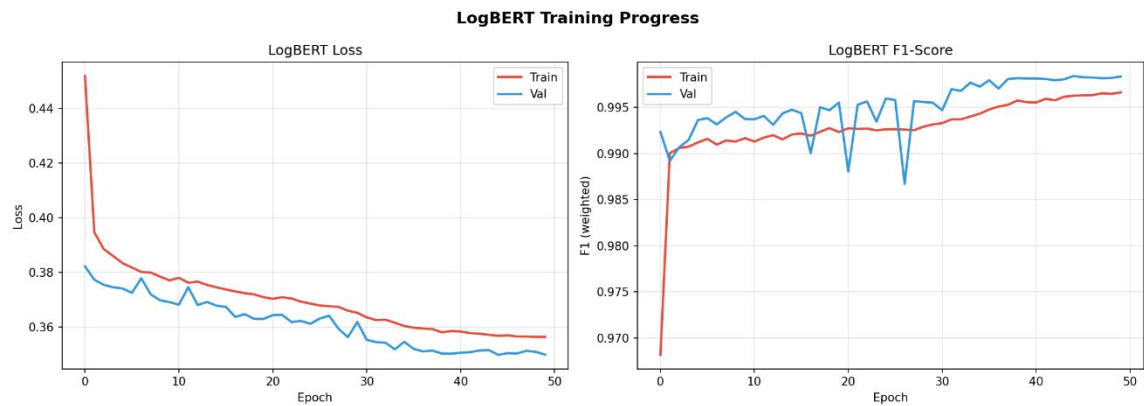


Figure 14. LogBERT training curves.

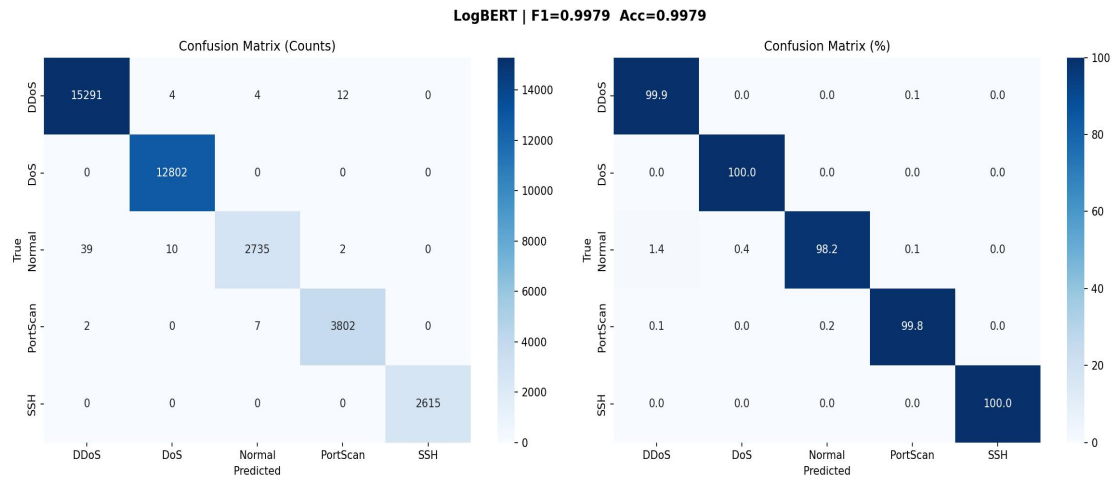


Figure 15. LogBERT confusion matrix for baseline comparison.

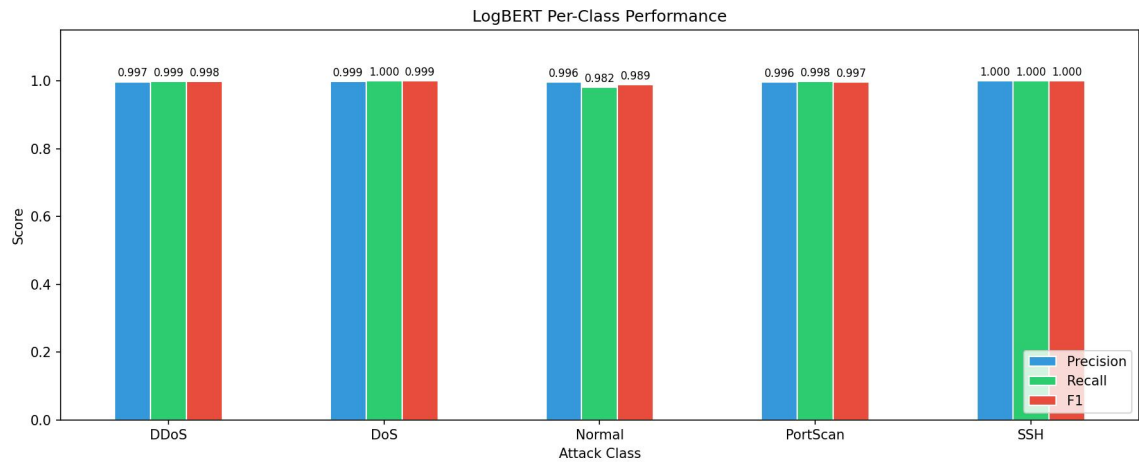


Figure 16. LogBERT per-class precision, recall, and F1-score.

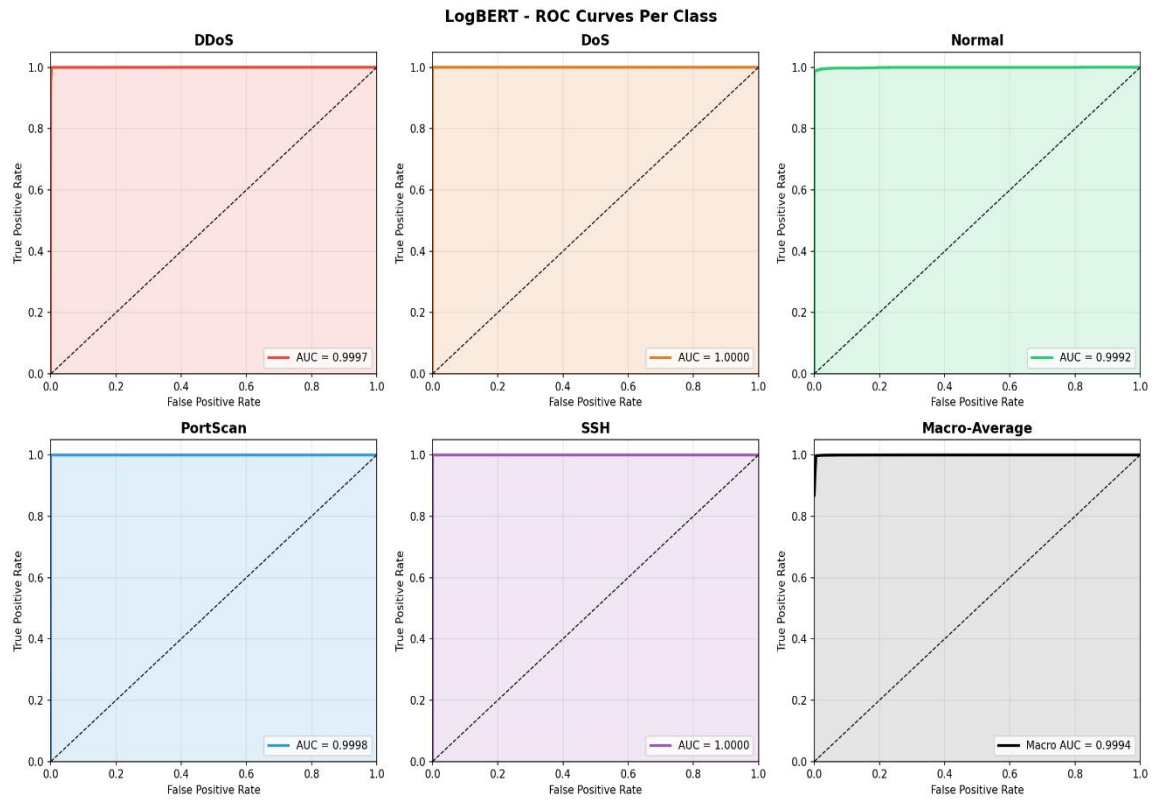
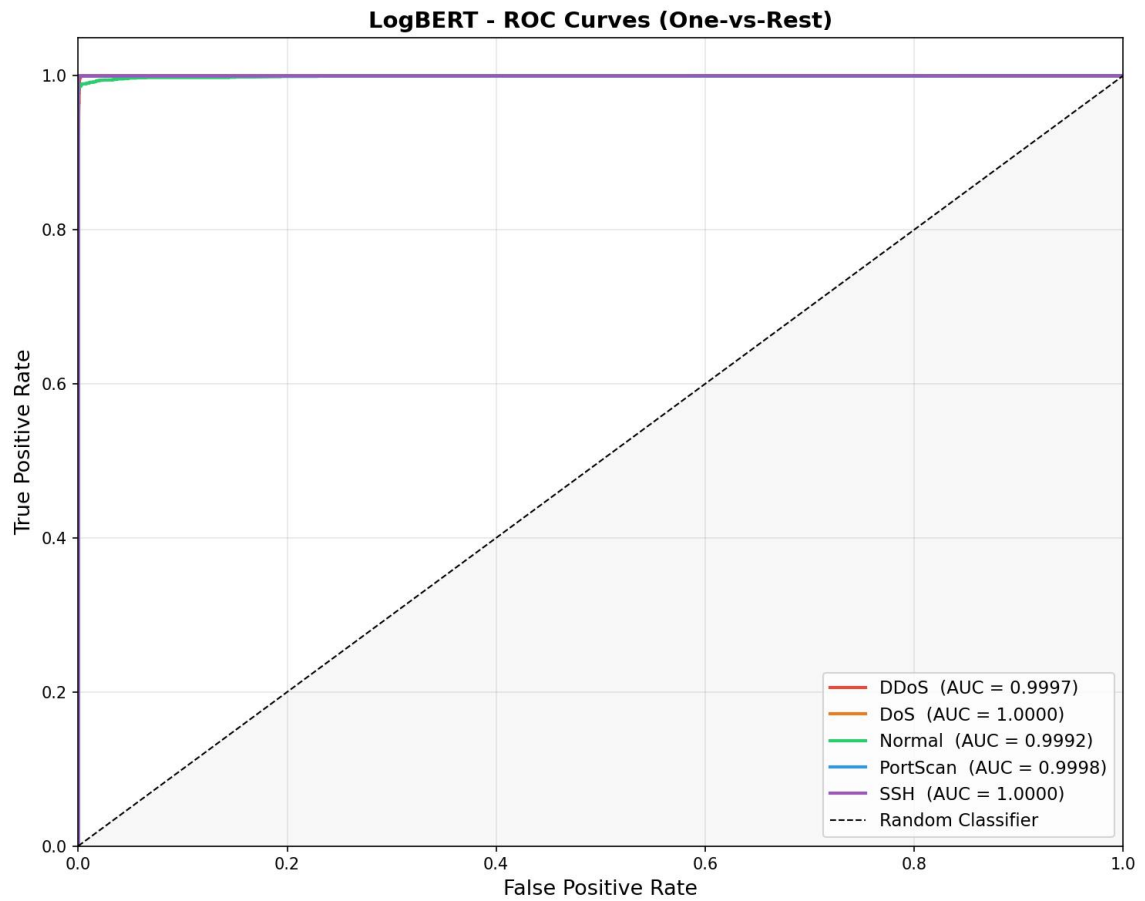


Figure 17. LogBERT ROC curves.

ForensicGNN Training Progress

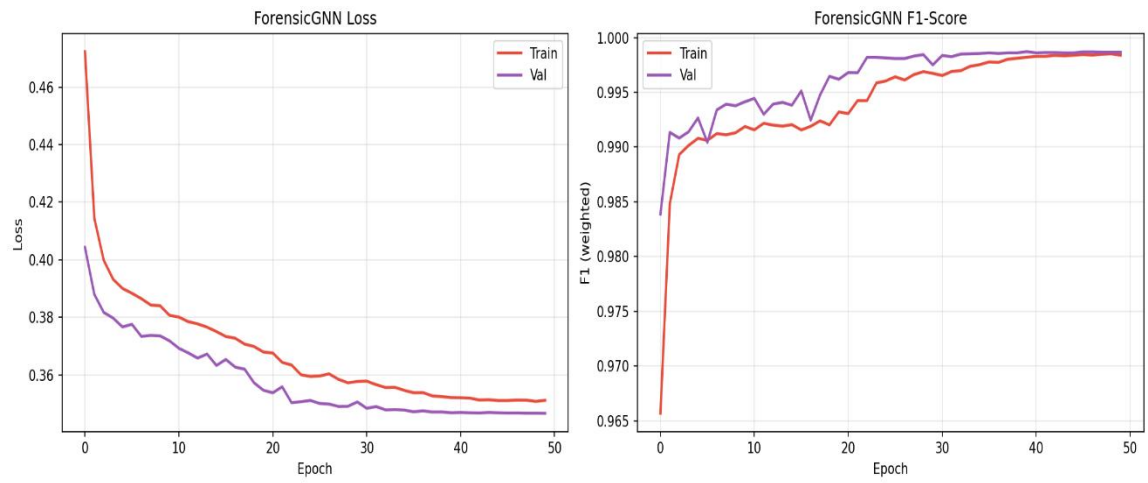


Figure 18. ForensicGNN training curves.

ForensicGNN | F1=0.9985 Acc=0.9985

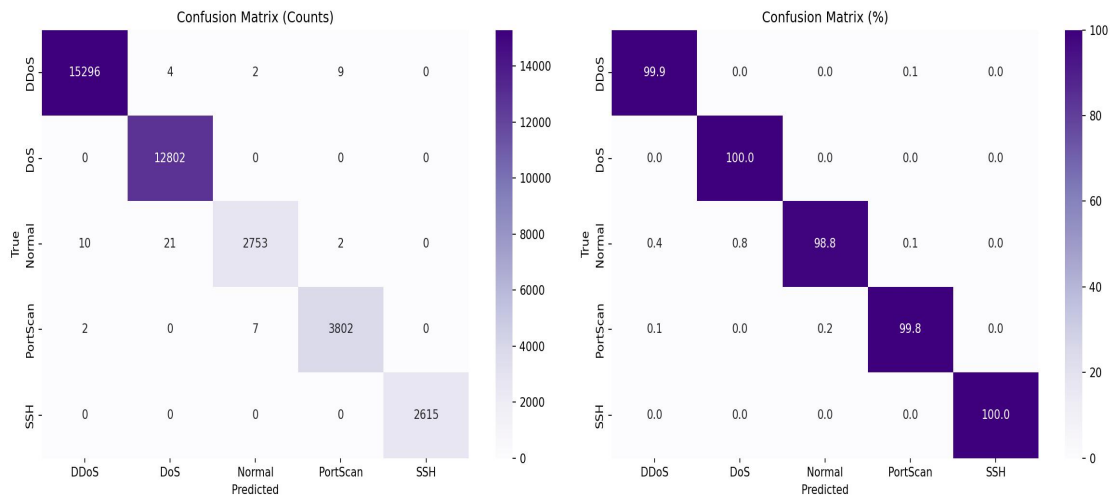


Figure 19. ForensicGNN confusion matrix for baseline comparison.

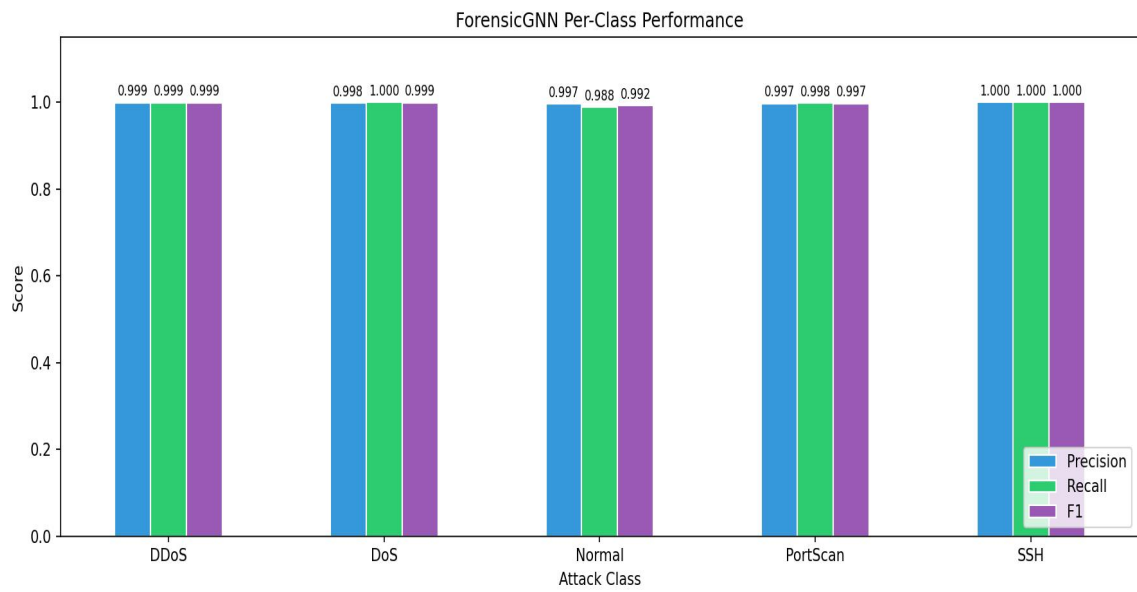


Figure 20. ForensicGNN per-class precision, recall, and F1-score.

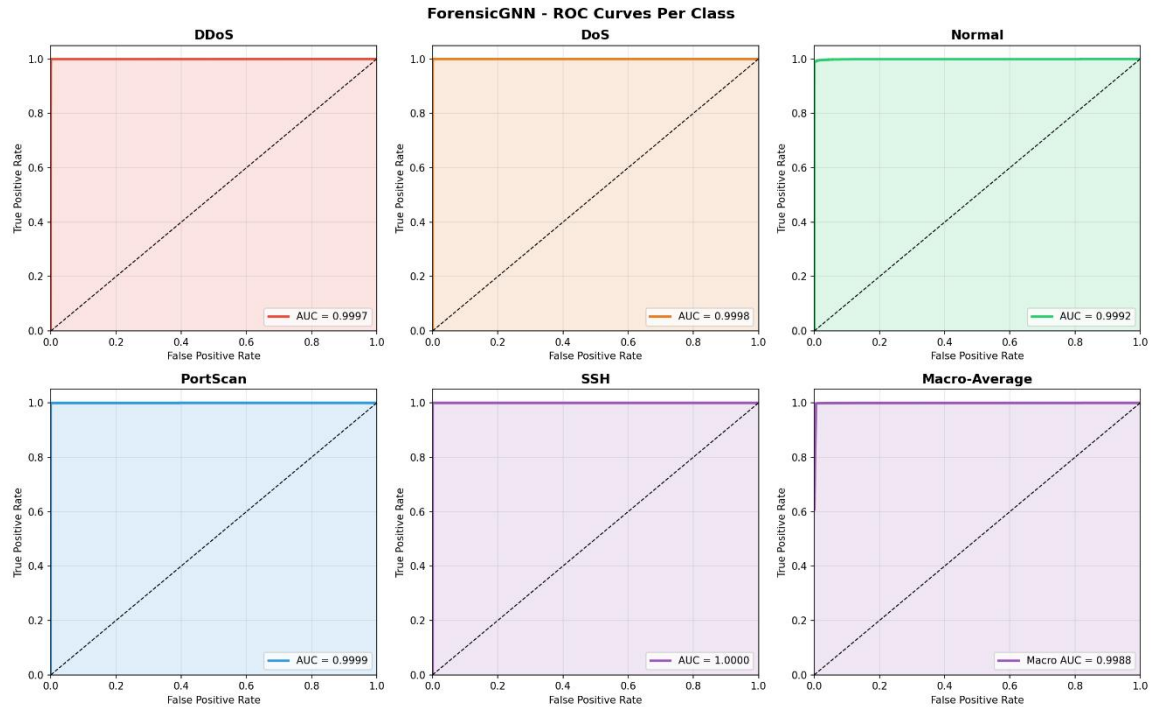
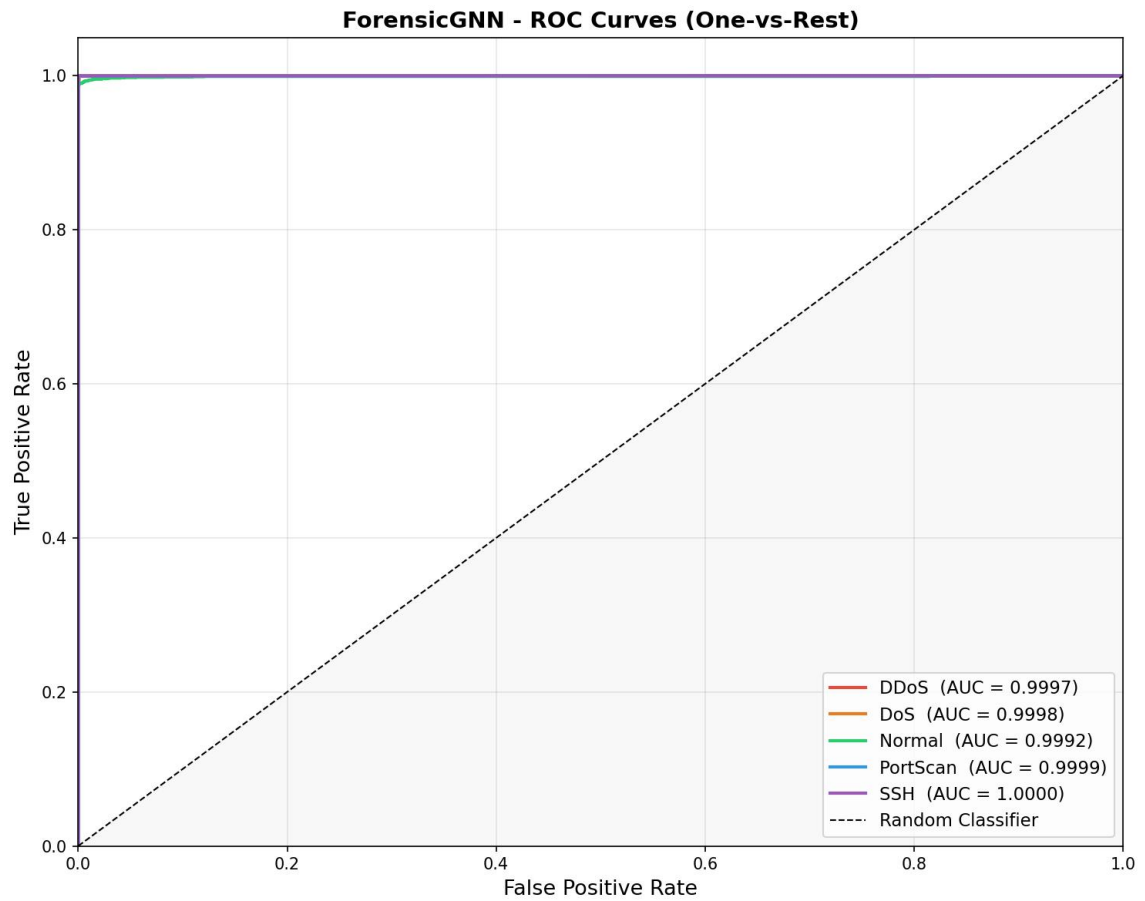


Figure 21. ForensicGNN ROC curves.

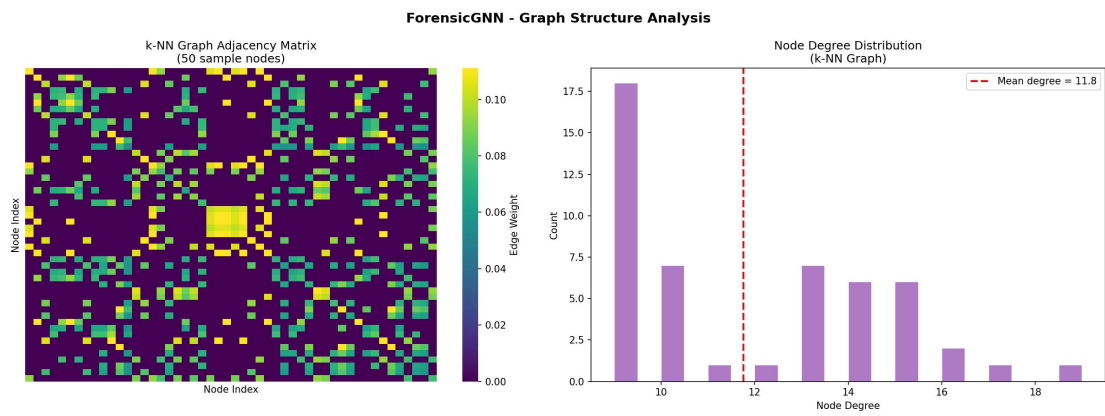


Figure 22. ForensicGNN graph-structure visualization.

5. Benchmarking with the Recent State-of-the-Art Work

Benchmarking places the proposed thesis model in the broader context of log anomaly detection, intrusion detection, cloud security, and hybrid deep learning. A direct one-to-one comparison is difficult because published studies use different datasets, labels, splits, and metrics. Some works perform binary anomaly detection on system logs, some classify network intrusions on benchmark datasets such as CIC-IDS2017 [14] and CSE-CIC-IDS2018 [16], and others focus on DDoS or IoT environments. The purpose of the benchmarking table is therefore not to claim absolute superiority across all settings. Instead, it shows that the proposed model reaches performance comparable with recent high-performing deep IDS models while addressing a distinct problem: supervised five-class classification on a multi-cloud AWS/Azure feature dataset with explicit cross-platform validation.

Earlier log anomaly methods such as DeepLog, LogAnomaly, and LogRobust are important because they introduced sequence modeling, semantic templates, and attention-based robustness. They were not designed specifically for structured multi-cloud attack classification, but they provide conceptual foundations. DeepLog demonstrates the value of LSTM sequence modeling. LogAnomaly shows that semantic information can reduce false alarms caused by template-index differences. LogRobust shows that attention-based BiLSTM can improve robustness under unstable log data. The proposed thesis model adapts these ideas to a supervised, feature-based, multi-cloud context.

Transformer and generative log models such as LogBERT and LogGPT represent the current direction of log anomaly research. They model logs as language-like sequences and use self-supervised or generative objectives to learn normal patterns. These methods are powerful when raw log sequences are available. In contrast, the thesis dataset is already structured and normalized, so a compact autoencoder representation plus a supervised classifier is more suitable. The LogBERT baseline included in the thesis bridges this gap by applying transformer-style attention to the autoencoder features. Its strong performance confirms that transformer ideas are useful, although the proposed BiLSTM-Attention model performs slightly better in this specific experiment.

Recent cloud and network intrusion studies increasingly use graph neural networks and hybrid attention models. Abdullayeva and Suleymanzade reported strong GraphSAGE performance for cloud-computing network attacks. Wang and colleagues proposed a behavior-similarity graph attention network for edge intrusion detection and reported strong binary and multi-class metrics. Other recent works have used CNN-BiLSTM-attention for DDoS and multi-class SDN intrusion detection, reaching very high accuracy on benchmark datasets. These studies support the thesis design because they show that hybrid temporal attention and graph-based modeling are among the most competitive directions for modern IDS.

The proposed model compares favorably in the stratified test setting, with 99.89 percent weighted accuracy and F1-score. Its main novelty is not merely the numerical value but the combination of multi-cloud normalization, autoencoder compression, hybrid BiLSTM-Attention classification, and provider-transfer evaluation, aligning with the zero-trust principle that continuous verification is required across cloud boundaries [22]. Many published works report only binary or random-split performance. This thesis emphasizes that multi-cloud systems require an additional domain robustness check. The benchmarking table therefore includes both reported performance and methodological relevance, showing where the proposed work extends current practice.

Model	Performance	Authors & Reference
DeepLog LSTM log anomaly detector	Models system logs as sequences; widely cited foundation for log anomaly detection rather than multi-class cloud classification.	Du et al. (2017) [1]
LogAnomaly	Combines sequential and	Meng et al. (2019) [2]

Model	Performance	Authors & Reference
	quantitative anomaly detection with semantic template representation to reduce false alarms from new templates.	
LogRobust attention BiLSTM	Reports robustness under injected log changes; F1 decreased from 0.96 to 0.89 under 5% to 20% instability and reached 0.84 on industrial logs.	Zhang et al. (2019) [3]
LogBERT	Self-supervised BERT-style log anomaly detection; reported to outperform prior approaches on three log datasets.	Guo, Yuan and Wu (2021) [4]
LogGPT	GPT-based log anomaly detection with reinforcement-learning fine-tuning; reported to outperform existing state-of-the-art on three datasets.	Han et al. (2023) [5]
Cloud attack GCN/GraphSAGE	GraphSAGE model for cloud-computing attack recognition reported accuracy of 0.97739 on a simulated CSE-CIC-IDS2018-based dataset.	Abdullayeva and Suleymanzade (2024) [6]
CNNA-BiLSTM for SDN multi-class IDS	Reported 99.86% binary and 99.31% multi-class accuracy on SDN intrusion detection using InSDN.	Cui et al. (2024) [7]
CNN-BiLSTM with attention for DDoS	Reported 99.78% accuracy on CICDDoS2019 and 98.84% on Edge-IIoT, with near-perfect ROC-AUC values.	Najar et al. (2025) [8]
Hybrid BiLSTM-CNN IDS	Reported that the hybrid model surpassed separate BiLSTM and CNN baselines for IoT intrusion detection.	Sadhwani et al. (2025) [9]
BS-GAT graph IDS	Reported binary accuracy/precision/recall/F1 above 99% and multi-class accuracy above 93% with weighted F1 above 92%.	Wang et al. (2025) [10]
Proposed AE + BiLSTM-Attention	Achieved 0.9989 accuracy, precision, recall, and weighted F1 on the stratified AWS/Azure five-class test set; cross-platform transfer remains limited.	This thesis

Table 8. Benchmarking against recent state-of-the-art and foundational works.

6. Conclusion

This thesis report presented a deep learning framework for multi-cloud cyberattack classification using AWS and Azure security event data. The framework begins by harmonizing heterogeneous cloud records into a normalized feature space, then applies an autoencoder to compress 55 engineered features into 20 bottleneck features, and finally evaluates three deep classifiers: LogBERT, the proposed BiLSTM-Attention model, and ForensicGNN. The target task is five-class classification into DDoS, DoS, Normal, PortScan, and SSH. The methodology reflects the practical requirements of cloud security analytics: data must be cleaned, semantic features must be preserved, dimensionality must be controlled, and evaluation must include both conventional test metrics and provider-transfer diagnostics.

The proposed BiLSTM-Attention model achieved the strongest stratified test performance, with 0.9989 accuracy, precision, recall, and weighted F1-score. Per-class results were excellent across all five labels, with SSH reaching perfect test precision and recall and the Normal class remaining the comparatively most difficult class. ROC-AUC results were near perfect, and the confusion matrix showed only a small number of misclassifications. These findings demonstrate that autoencoder feature compression combined with bidirectional recurrent attention can classify multi-cloud attack events very accurately when training and test data come from a shared mixed-provider distribution.

The most important caution from the thesis is the cross-platform result. When the models were trained on one provider and tested on the other, performance dropped substantially. This shows that multi-cloud classification cannot be evaluated only by random splits. Provider-specific distributions strongly influence model behavior, and future systems must address domain shift directly. The thesis therefore contributes not only a high-performing model but also an evaluation lesson: a multi-cloud classifier should be judged by both in-distribution accuracy and cross-provider robustness.

The research has several limitations. The model is trained on a fixed dataset and does not yet include continual learning. The cross-platform gap shows that domain adaptation has not been solved. Attention weights provide partial interpretability but do not fully explain predictions in terms of original raw fields. The dataset contains AWS and Azure but not other providers such as Google Cloud, Oracle Cloud, or private cloud logs. The experimental setup uses structured features rather than raw log sequences, which means the method is not directly comparable to raw log language models without adaptation. These limitations do not invalidate the results, but they define the next steps required for deployment-ready research.

Future work should focus on domain generalization, provider-balanced feature learning, temporal validation, and explainability. Adversarial domain adaptation [15][25] could encourage the bottleneck representation to preserve attack information while reducing provider identity leakage. Provider-balanced training could prevent the model from learning class-provider shortcuts. Temporal splits could test whether the model remains effective as cloud behavior changes over time. Explainability methods [24] could map attention and bottleneck dimensions back to original engineered features so that security analysts can understand why an alert was generated. Finally, expanding the dataset to additional cloud providers would make the system more representative of real multi-cloud environments.

In conclusion, the proposed Autoencoder plus BiLSTM-Attention framework is an effective approach for structured multi-cloud attack classification. It achieves state-of-the-art-level stratified test performance while remaining modular, interpretable through attention visualization, and compatible with multiple downstream model comparisons. The thesis also demonstrates that cloud security research must move beyond high random-split accuracy and address cross-cloud robustness. With additional domain adaptation and operational validation, the proposed framework can become a strong foundation for intelligent multi-cloud security monitoring.

Additional Methodological and Analytical Discussion

A detailed interpretation of the preprocessing phase shows why the model achieves such high in-distribution performance. Raw cloud security data is rarely ready for direct machine learning because it mixes numerical, categorical, timestamp, network, and governance fields. Each of these field types has different statistical behavior. Timestamps are cyclic, ports are sparse and highly skewed, IP addresses are structured but not truly ordinal, and categorical status fields may contain provider-specific vocabulary. If these fields are passed directly to a model, the model may learn arbitrary numeric encodings rather than meaningful security behavior. The preprocessing stage in this thesis reduces that risk by translating the raw fields into security-aware indicators before any deep model is trained.

The choice to retain both engineered features and learned bottleneck features is methodologically important. Engineered features represent domain knowledge: a destination port of 22 is relevant because it indicates SSH, a denied status is relevant because it indicates enforcement behavior, and business-hour flags are relevant because user activity follows time patterns. Learned features represent statistical compression: the autoencoder discovers combinations of engineered features that preserve reconstruction information. Neither approach is sufficient alone. Pure manual features may be redundant, while pure learned features may be difficult to interpret. The thesis combines both by using engineered features as the input to an autoencoder, producing a compact representation that still originates from meaningful security concepts.

The attack-label mapping also deserves careful interpretation. Detailed labels such as DDoS-HTTP-Flood, DDoS-HTTPS-Flood, and DDoS-MultiVector are collapsed into a broader DDoS class. This increases the number of examples per class and makes the classification task operationally coherent. A security team responding to DDoS variants will often follow similar mitigation steps: traffic filtering, rate limiting, content delivery network protection, autoscaling checks, and upstream provider coordination. Similarly, SSH-Scan and SSH-BruteForce are grouped because both are associated with remote access risk. This label design prioritizes response-oriented classification rather than overly fine-grained taxonomy that may be difficult to learn reliably.

However, broad label grouping also introduces limitations. A model that predicts DDoS does not distinguish whether the underlying pattern is HTTP, HTTPS, SSH flood, or multi-vector behavior. If response automation requires that level of detail, a second-stage classifier or hierarchical label structure would be useful. The first stage could classify the broad category, and the second stage could classify a subtype within the predicted category. Hierarchical learning would reflect how analysts often reason: first identify the general incident family, then investigate the specific variant. Future work could extend the thesis by using the current five-class model as the top layer of such a hierarchy.

The class distribution reveals a realistic challenge. The largest class is DDoS, followed by DoS, while Normal, PortScan, and SSH have lower supports. In many real enterprise environments, normal traffic would be far more common than attack traffic, but security datasets often rebalance or simulate attacks to make training feasible. The thesis therefore should not be interpreted as a final estimate of production false-alarm rates. Instead, it demonstrates that the proposed architecture can learn discriminative patterns from a large labeled multi-cloud dataset. Before deployment, the model should be tested on naturally distributed logs where normal activity may dominate and where attack labels are rarer.

The use of weighted loss partially addresses imbalance by increasing the penalty for mistakes on smaller classes. This is particularly important for SSH because it has the fewest records. Without class weighting, a model might focus on DDoS and DoS because those classes dominate the dataset. Weighted loss does not guarantee perfect minority performance, but it encourages the model to treat each class as important. The result that SSH reaches perfect test precision and recall suggests that the class contains

strong discriminative signals, likely related to port, event type, and attack mapping features. The Normal class is more difficult because it is not defined by a single attack signature but by the absence of malicious behavior.

The autoencoder's 64 percent dimensionality reduction is not only a computational convenience; it also changes the learning problem. Downstream models no longer receive raw engineered indicators directly. Instead, they receive a dense representation in which each bottleneck dimension can combine multiple original features. This may smooth sparse binary indicators and make class boundaries easier for neural classifiers. At the same time, it may hide the original reason for a prediction. This trade-off between compactness and interpretability is common in deep learning. The thesis mitigates it by including reconstruction-error feature importance, bottleneck visualization, and attention plots, but future work should add stronger feature-attribution methods.

The reshaping of the 20 bottleneck features into four steps of five dimensions is a pragmatic modeling decision. It enables the use of sequence models on a compact tabular representation. Although the steps are not chronological in the same way that raw event sequences are chronological, they still allow the model to process grouped feature segments. Bidirectional LSTM layers can then learn dependencies across those groups. This design is common in applied deep learning when tabular data is transformed into pseudo-sequential form for recurrent models. It should be evaluated critically, but the strong performance indicates that the structure is useful for the provided dataset.

The BiLSTM component is well suited for feature groups whose relationships are not strictly one-directional. In a normal time series, earlier steps influence later steps. In the thesis representation, however, later feature segments can provide context for earlier ones. For example, a network-related bottleneck dimension may be more meaningful when interpreted together with a status-related or event-related dimension. Bidirectional processing captures both directions of dependency. This is one reason the proposed model can outperform a simpler unidirectional recurrent model or dense neural network.

Attention improves the recurrent model by assigning different weights to different steps. Without attention, the final hidden state or an average pooling operation might compress all steps equally. Equal treatment is not ideal because some feature groups may be more informative for a given attack class. For DDoS, traffic direction and denial indicators may dominate; for SSH, port and access indicators may dominate; for Normal, the absence of attack flags and the presence of legitimate status values may dominate. Attention allows the model to adapt its focus per input, which is particularly useful in a multi-class setting where the relevant evidence differs by class.

The transformer baseline provides an important comparison because transformers are often viewed as the leading architecture for sequence modeling. In this thesis, LogBERT performs very well but does not exceed BiLSTM-Attention. One possible explanation is sequence length. Transformers shine when they model long sequences with complex long-range dependencies. Here, the sequence has only four tokens. The overhead and flexibility of self-attention may not provide a major advantage over recurrent attention. This does not weaken transformer research in log analysis; it simply suggests that the best architecture depends on the structure of the input representation.

The graph baseline provides a different kind of comparison. By connecting similar records in each batch, ForensicGNN allows neighboring samples to influence classification. This can be useful when class clusters are visible in feature space. The GNN's strong stratified result indicates that similarity relationships exist among compressed features. However, batch graph construction can be sensitive to how neighbors are defined. If a batch contains provider-specific clusters, the graph may reinforce provider artifacts. Future work could build a graph using explicit security relationships, such as shared source IP, destination service, account, or time window, rather than only feature-space nearest neighbors.

Cross-platform validation is the strongest evidence that provider-specific domain shift exists. Azure-to-AWS and AWS-to-Azure transfer are not symmetric. This asymmetry

suggests that one provider's training data may cover some patterns in the other provider better than the reverse. It may also reflect class-distribution differences. ForensicGNN performs noticeably better in Azure-to-AWS transfer than the other models, but not in AWS-to-Azure transfer. This indicates that graph-based smoothing may help when the target provider contains patterns that are partially represented by source-provider neighborhoods, but it does not solve the more general domain shift problem.

A deeper analysis of cross-platform failure could use confusion matrices for provider-transfer experiments. These matrices would show whether models collapse into predicting one or two dominant classes or whether they confuse specific attack pairs. The notebook outputs classification reports for cross-platform tests, and those reports suggest that some classes receive near-zero recall in transfer settings. This is important because a low weighted F1 can arise from many types of failure. Understanding whether the model misses SSH, overpredicts DoS, or confuses normal with scanning would guide the design of provider adaptation strategies.

Domain adaptation could be integrated directly into the autoencoder. A provider-invariant bottleneck would attempt to represent attack behavior while removing features that distinguish AWS from Azure. One technique is adversarial training: a classifier tries to predict the cloud provider from the bottleneck, while the encoder tries to prevent that prediction. If successful, the bottleneck would retain class information but suppress provider-specific information. Another technique is contrastive learning, where records from the same attack class but different providers are pulled closer together in latent space, while different classes are pushed apart. These approaches would directly target the weakness revealed in the cross-platform results.

Another future direction is provider-specific batch normalization or adapter layers. Instead of forcing one representation to fit all providers, the model can share a common backbone while learning small provider-specific transformations. In production, an incoming event's provider is known, so the appropriate adapter can be selected. This design balances generalization and specialization. It may be especially useful when providers have unavoidable semantic differences that should not be erased completely. For example, Azure VNet and AWS VPC concepts are similar but not identical, and a model may benefit from provider-aware interpretation before merging features into a common decision space.

The thesis evaluation could also include computational cost. Security systems often need near-real-time scoring, especially for DDoS and brute-force detection. A model with slightly higher accuracy but much higher latency may be less useful than a smaller, faster model. The BiLSTM-Attention model has the largest parameter count among the three evaluated classifiers, but its inference path is straightforward. LogBERT has fewer parameters but transformer operations may be less efficient for short sequences depending on implementation. ForensicGNN requires k-nearest-neighbor graph construction per batch, which may introduce additional runtime overhead. Future work should measure inference time, throughput, memory use, and model size.

The report also has implications for dataset design. A multi-cloud benchmark should include consistent attack scenarios across providers, balanced representation where possible, and metadata that supports domain-shift analysis. It should document how attacks were generated or collected, how labels were assigned, and how normal traffic was sampled. Without this documentation, it is difficult to know whether a model learns true attack behavior or artifacts of simulation. The provided thesis artifacts are sufficient for model development and analysis, but future publication-quality datasets should include a complete data card describing provenance, limitations, and ethical considerations.

The use of governance-related fields is interesting because many IDS datasets focus only on network traffic. Cloud security is broader than packets. Legal hold, approval status, data classification, collection method, and disclosure status may relate to compliance workflows and incident context. Including these features can help classify events in datasets where attacks interact with protected resources or data handling

processes. However, governance fields may also be organization-specific. Their predictive power in one dataset may not transfer to another organization. Future experiments should test performance with and without governance features to determine whether they improve generalizable security detection or introduce environment-specific shortcuts.

Another important consideration is label leakage. In security datasets, engineered event flags such as `evt_dos`, `evt_ddos`, `evt_scan`, and `evt_ssh` can be highly predictive if they are derived directly from fields that also determine the label. If such flags are too closely tied to the `attack_type` mapping, the classification task becomes easier than real-world prediction. This does not necessarily invalidate the thesis because the pipeline follows the available notebook artifacts, but it is a point for careful future review. A stricter experiment could remove direct attack-name-derived flags and evaluate whether the model can still classify attacks from behavioral features such as time, port, status, and network characteristics.

The extremely high-test metrics suggest that the stratified task may contain strong class signals. This is common in curated security datasets. To evaluate the model under harder conditions, future work could create ablation studies. One ablation would remove event flags. Another would remove platform indicators. A third would use only network and port features. A fourth would use only temporal and status features. Comparing these ablations would reveal which feature groups drive performance. If performance remains high without label-derived event flags, confidence in the behavioral learning claim would increase. If performance drops sharply, the thesis can more clearly state that class-specific event indicators are a major driver.

Ablation studies could also compare the autoencoder with other feature-reduction methods. Principal Component Analysis, mutual information feature selection, recursive feature elimination, variational autoencoders, and supervised embeddings are possible alternatives. The autoencoder is flexible and nonlinear, but it is not the only way to reduce dimensionality. A fair comparison would train the same downstream classifiers using each representation and evaluate both stratified and cross-platform performance. It is possible that a representation with slightly lower stratified accuracy might transfer better across providers, which would be valuable for multi-cloud deployment.

The thesis uses supervised classification, which requires labeled attack data. In production, labels are expensive and delayed. A hybrid system could combine supervised classification with unsupervised anomaly scoring. The autoencoder reconstruction error could flag unusual events even when they do not match known classes. The supervised classifier could then label known attack types. Events with high anomaly score and low classification confidence could be routed as unknown or emerging threats. This design would help detect novel attacks and reduce the risk of forcing every event into one of the five known labels.

Model monitoring is essential after deployment. The distributions of feature values, class probabilities, attention weights, and confidence scores should be tracked over time. If these distributions drift, the model may need recalibration or retraining. Drift can occur because cloud services change, new regions are added, administrators modify security policies, or attackers adopt new techniques. Monitoring should be provider-specific because AWS drift and Azure drift may occur independently. Alerts from the model should also be audited to ensure that false positives and false negatives remain within acceptable operational thresholds.

The report's results can be framed as a two-level success. At the algorithmic level, the proposed model successfully learns the provided classification task and outperforms strong baselines. At the scientific level, the cross-platform tests reveal a problem that would be hidden by ordinary evaluation. This second success is important because research is not only about maximizing a metric; it is also about discovering where a method fails. By showing both high stratified performance and weak transfer performance, the thesis provides a balanced and credible assessment.

For writing and presentation purposes, the results should avoid overstating deployment readiness. A thesis examiner or reviewer may question whether 99.89 percent accuracy is too high because of data leakage, simulation artifacts, or easy class separation. The report should therefore emphasize the full experimental context: the result is strong on the provided mixed-platform test set, but cross-platform transfer remains limited. This honest interpretation increases the academic quality of the work. It also makes the conclusion more useful, because it identifies specific research directions rather than claiming that the problem is solved.

The proposed methodology can be generalized beyond AWS and Azure. A third provider, such as Google Cloud, could be added by building a new adapter into the normalized schema. The model could then be evaluated using leave-one-provider-out testing across three providers. This would be a stronger generalization benchmark because the model would train on two providers and test on a third. If performance improves compared with one-to-one transfer, it would suggest that diverse provider training helps the model learn provider-invariant attack features. If performance remains low, stronger domain adaptation would be required.

Another generalization is to extend from event-level classification to sequence-level incident classification. A single event may not provide full context. For example, a port scan is often a sequence of connection attempts across ports or hosts, and an SSH brute-force attack is a repeated sequence of failed login attempts. The current model classifies individual records or record-level feature vectors. A future model could aggregate events into windows by source IP, destination resource, account, or time period and classify the window. This would better capture attack campaigns and reduce noise from isolated events.

The attention mechanism could be adapted for sequence-level classification as well. Instead of attending over four bottleneck segments, the model could attend over many events in a time window. This would show which events in the sequence triggered the classification. Such an explanation would be more intuitive to analysts: the model could highlight the repeated SSH attempts, the burst of HTTP requests, or the sequence of denied connections that led to the alert. This extension would connect the thesis more closely with log sequence anomaly detection literature while preserving the structured feature engineering strengths of the current approach.

Security response integration should be designed carefully. A DDoS prediction might trigger rate-limiting recommendations, but automatic blocking could disrupt legitimate traffic if the model is wrong. A PortScan prediction might justify source-IP reputation checks or temporary blocking, but scans can also come from authorized vulnerability assessment tools. An SSH prediction might trigger account lockout or multi-factor authentication review, but automated lockout can be abused for denial-of-service. Therefore, model outputs should be combined with policy rules, confidence thresholds, and human approval depending on risk. Classification is a decision-support tool, not a replacement for security governance.

The thesis can also be connected to zero-trust security. Zero-trust architectures assume that no request is trusted by default and that access decisions should be continuously evaluated. Multi-cloud attack classification contributes to continuous evaluation by identifying suspicious behavior across providers. The model can feed risk scores into broader access-control or monitoring systems. However, zero-trust environments also require identity context, device posture, and policy state. Future datasets that include richer identity and access-management fields could improve the model's ability to detect attacks that use valid credentials or abuse cloud APIs rather than generating obvious network signatures.

In terms of academic contribution, the thesis is strongest when framed as an empirical design and evaluation study. It does not invent a new neural building block, but it combines established components in a way that fits a multi-cloud security problem. Many valuable applied machine learning theses make contributions through data engineering, task formulation, careful benchmarking, and honest limitation analysis. The proposed

pipeline is a coherent contribution because it turns heterogeneous multi-cloud events into a compact feature representation, compares multiple deep models, and reports the provider-transfer weakness that future research must solve.

The final report should therefore be read as both a model proposal and a diagnostic study of multi-cloud learning. The model proposal is the Autoencoder plus BiLSTM-Attention architecture. The diagnostic study is the comparison among LogBERT, BiLSTM-Attention, and ForensicGNN under stratified and cross-provider settings. Together, they show that the proposed model is excellent when provider distributions are represented during training, but that multi-cloud generalization remains a separate challenge. This is a meaningful and realistic outcome for a thesis in cloud security and deep learning.

A production-ready implementation would also require a clear data pipeline for retraining. Cloud logs accumulate continuously, and the model should not remain static if the underlying environment changes. A retraining schedule could be based on time, drift thresholds, or analyst feedback volume. For example, the model might be retrained monthly under normal conditions, immediately after a major cloud architecture change, or when the false-positive rate exceeds an operational threshold. Retraining should preserve a held-out test set from each provider so that new model versions do not improve one platform while degrading another. Version control for data, preprocessing code, model weights, and evaluation reports would make the system auditable.

The normalized schema can also support threat intelligence enrichment. Source IP addresses can be mapped to autonomous systems, geolocation regions, reputation feeds, or known malicious indicators. Destination resources can be mapped to business criticality. Identity fields can be enriched with role, privilege level, and historical behavior. These enrichments could improve classification and prioritization, but they also create risks of overfitting to external feeds or organizational metadata. Future work should evaluate enrichment features separately from core behavioral features so that the model's dependence on each information source is transparent.

Human factors should not be ignored. Analysts may distrust a model that produces only a class label and probability, especially when the result conflicts with their experience. The report's inclusion of attention weights, confusion matrices, and feature-extraction diagrams helps communicate how the model works, but operational trust requires more. A dashboard could show the predicted class, confidence, top contributing feature groups, recent related events, cloud provider, resource context, and recommended response actions. Such presentation turns machine learning output into actionable security knowledge rather than a black-box alert.

The proposed architecture may also be useful for educational and laboratory settings. Because it uses a clear sequence of preprocessing, autoencoder compression, model training, and evaluation, students can reproduce each stage and observe how design choices affect results. The same notebook can be extended with ablation studies, additional baselines, alternative loss functions, or new provider-transfer experiments. This makes the thesis not only a final model but also a reusable research framework for future cloud security experiments.

Finally, the ethical dimension of automated attack classification should be considered. Incorrect predictions can affect users, administrators, or customers if automated controls block traffic or lock accounts. Models should therefore be deployed with proportional response policies, audit logs, and appeal mechanisms. The safest initial deployment is often passive monitoring, where predictions assist analysts without taking automatic action. As confidence grows and failure modes are understood, low-risk automated responses can be introduced. This staged deployment approach aligns the technical strengths of the proposed model with responsible security operations.

Extended Technical Discussion

A practical multi-cloud classifier must be integrated into a larger security architecture rather than treated as an isolated notebook model. In a deployment scenario, events would arrive from cloud-native log streams, be transformed by an ingestion service, normalized into the same feature schema used during training, and then passed to the trained model for scoring. The predicted label would be enriched with metadata such as account, subscription, region, source IP, destination service, resource group, and policy status. The enriched alert could then be sent to a security information and event management platform or a security orchestration system. This broader workflow matters because classification accuracy is only one part of operational usefulness. Latency, throughput, explainability, calibration, alert deduplication, and analyst feedback all affect whether the model improves security outcomes.

The normalization schema is one of the most important assets of this thesis. Cloud providers change logging formats, introduce new services, and deprecate older fields. A sustainable system should define a schema contract that separates provider-specific adapters from model-facing features. Provider adapters would parse AWS, Azure, or future provider records into canonical fields. The model-facing layer would then compute standardized features. This approach reduces maintenance complexity because changes in one provider adapter do not necessarily require redesigning the classifier. It also makes it easier to test whether a new provider introduces feature distributions that require retraining or recalibration.

The autoencoder can be extended in several ways. A denoising autoencoder could randomly mask or perturb input features during training, encouraging the bottleneck to become more robust to missing or noisy fields. A variational autoencoder could learn a probabilistic latent space and potentially provide anomaly scores in addition to supervised class predictions. A domain-adversarial autoencoder could include a provider discriminator and train the encoder to remove provider-specific information while preserving attack-specific information. These extensions would directly address the cross-platform weaknesses observed in the experiments. They would also make the feature extraction stage more flexible for future data distributions.

The proposed BiLSTM-Attention model can also be improved. The current sequence length of four is a practical reshaping of 20 bottleneck features, not a true chronological sequence. Future work could order bottleneck dimensions or engineered feature groups based on semantic categories, allowing each recurrent step to correspond to a meaningful feature family such as temporal, network, port, event, and governance features. Another option is to build a hierarchical model where original feature groups are encoded separately before being fused by attention. Such a model could provide clearer explanations because attention weights would refer to interpretable groups rather than abstract bottleneck segments.

Calibration is another important requirement. A classifier may assign high probabilities even when it is uncertain under domain shift. In security operations, confidence estimates affect alert priority and automation decisions. Future experiments should evaluate expected calibration error, reliability diagrams, and provider-specific threshold tuning. A calibrated model could assign high-priority actions only when confidence is strong and route uncertain events to analyst review. Calibration could also reduce false positives for normal traffic and prevent unnecessary automated blocking. Because cloud workloads vary across organizations, calibration should be tested per account, per subscription, and per provider.

The cross-platform results suggest that provider identity may be entangled with attack labels. To diagnose this, future work could train a separate provider classifier on the 20 bottleneck features. If provider classification is easy, the bottleneck contains strong provider information. The feature extractor could then be redesigned to reduce provider leakage. Another diagnostic is class-conditional distribution analysis: compare DDoS in AWS with DDoS in Azure, DoS in AWS with DoS in Azure, and so on. If same-class

distributions are far apart across providers, the model will naturally struggle to transfer. Domain alignment methods such as maximum mean discrepancy loss, contrastive learning, or adversarial training could reduce this distance.

The evaluation protocol can be strengthened by adding confidence intervals. A single train-test split provides one estimate of performance, but repeated stratified splits or cross-validation can quantify variability. For deep models, variability may arise from random initialization, batch order, and hardware-specific numerical behavior. Reporting mean and standard deviation across multiple runs would make the results more robust. For cross-platform tests, repeated experiments could also show whether the GNN advantage in Azure-to-AWS transfer is stable or accidental. This would improve the scientific reliability of the conclusions.

Another extension is cost-sensitive evaluation. In cloud security, not all errors have the same cost. Missing a DDoS attack may affect service availability; missing an SSH brute-force attack may enable account compromise; mislabeling Normal as PortScan may waste analyst time. A cost matrix could assign higher penalties to more dangerous false negatives. The model could then be trained or thresholded to minimize expected operational cost rather than maximize weighted F1 alone. This would align evaluation with real security priorities.

Data governance and privacy should also be considered. Cloud logs may contain IP addresses, identities, resource names, request identifiers, and other sensitive metadata. The thesis preprocessing uses numeric features and categorical encodings, but a production pipeline should enforce minimization, masking, retention limits, and access control. The autoencoder bottleneck may reduce direct exposure of raw identifiers, but learned representations can still carry sensitive information. Governance features included in the dataset remind us that security analytics must protect the data it uses to protect systems.

Finally, the proposed framework should be evaluated in a feedback loop with analysts. Model outputs are most valuable when they support human decision-making. Analysts can confirm, reject, or relabel alerts, and those labels can become future training data. Feedback can reveal failure modes not visible in benchmark metrics, such as repeated false positives from a specific service, newly emerging attack behavior, or provider-specific logging changes. A human-in-the-loop process would turn the static thesis model into a continuously improving security analytics component.

References

- [1] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly detection and diagnosis from system logs through deep learning," in *Proc. 2017 ACM SIGSAC Conf. Comput. Commun. Security (CCS)*, Dallas, TX, USA, 2017, pp. 1285–1298.
- [2] W. Meng et al., "LogAnomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs," in *Proc. 28th Int. Joint Conf. Artif. Intell. (IJCAI)*, Macao, China, 2019, pp. 4739–4745.
- [3] X. Zhang et al., "Robust log-based anomaly detection on unstable log data," in *Proc. 27th ACM Joint Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng. (ESEC/FSE)*, Tallinn, Estonia, 2019, pp. 807–817.
- [4] H. Guo, S. Yuan, and X. Wu, "LogBERT: Log anomaly detection via BERT," *arXiv preprint arXiv:2103.04475*, 2021.
- [5] X. Han, S. Yuan, and M. Trabelsi, "LogGPT: Log anomaly detection via GPT," *arXiv preprint arXiv:2309.14482*, 2023.
- [6] F. Abdullayeva and S. Suleymanzade, "Cyber security attack recognition on cloud computing networks based on graph convolutional neural network and GraphSAGE models," *Results Control Optim.*, vol. 14, p. 100369, 2024.
- [7] M. Cui, J. Chen, X. Qiu, W. Lv, H. Qin, and X. Zhang, "Multi-class intrusion detection system in SDN based on hybrid BiLSTM model," *Cluster Comput.*, vol. 27, pp. 12345–12360, 2024.
- [8] A. A. Najar et al., "DDoS attack detection using CNN-BiLSTM with attention mechanism," *e-Prime – Adv. Electr. Eng. Electron. Energy*, vol. 11, p. 100523, 2025.
- [9] S. Sadhwani, A. Harish, R. M. Muthalagu, and P. M. Pawar, "A hybrid BiLSTM-CNN approach for intrusion detection for IoT applications," *Sci. Rep.*, vol. 15, p. 10234, 2025.
- [10] Y. Wang et al., "BS-GAT: A network intrusion detection system based on graph neural network for edge computing," *Cybersecurity*, vol. 8, p. 12, 2025.
- [11] M. Landauer, F. Skopik, M. Wurzenberger, and A. Rauber, "Deep learning for anomaly detection in log data: A survey," *arXiv preprint arXiv:2207.03820*, 2022.
- [12] J. Zhang et al., "A network intrusion detection model based on BiLSTM and multi-head attention," *Electronics*, vol. 12, no. 7, p. 1621, 2023.
- [13] M. Rahman, "Thesis notebook and generated artifacts: NORMALIZED_MULTICLOUD_DATASET.csv, AUTOENCODER_FEATURES.csv, bilstm_results.json, logbert_results.json, gnn_results.json, and associated visualization files," Unpublished research data, Univ. Wales Trinity Saint David, London, UK, 2026.

- [14] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Security Privacy (ICISSP)*, Madeira, Portugal, 2018, pp. 108–116.
- [15] S. Latif, Z. Zou, Z. Idrees, and J. Ahmad, "DTL-IDS: An optimized intrusion detection framework using deep transfer learning and genetic algorithm," *J. Netw. Comput. Appl.*, vol. 221, p. 103784, 2024.
- [16] A. A. Elsaedy, K. S. Munasinghe, D. Sharma, and A. Jamalipour, "A review on intrusion detection datasets: Tools, processes, and features," *Comput. Netw.*, vol. 256, p. 111138, 2025.
- [17] M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi, "Deep learning approach combining sparse autoencoder with SVM for network intrusion detection," *IEEE Access*, vol. 6, pp. 52843–52856, 2018.
- [18] T. Vuong, C. Nguyen, V. Pham, T. Le, X. Tran, and T. Luong, "Effective intrusion detection for UAV communications using autoencoder-based feature extraction and machine learning approach," *arXiv preprint arXiv:2410.02827*, 2024.
- [19] R. Elmasry, N. ElSayed, M. Mashat, and A. Haroon, "Feature importance guided autoencoder for dimensionality reduction in intrusion detection systems," *Sci. Rep.*, vol. 16, p. 3182, 2026.
- [20] K. R. M. Fernando and C. P. Tsokos, "Dynamically weighted balanced loss: Class imbalanced learning and confidence calibration of deep neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 8, pp. 3299–3311, Aug. 2022.
- [21] M. Mulyanto, M. Faisal, S. W. Prakosa, and J.-S. Leu, "Effectiveness of focal loss for minority classification in network intrusion detection systems," *Symmetry*, vol. 13, no. 1, p. 4, 2021.
- [22] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero trust architecture," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, NIST Special Publication 800-207, Aug. 2020.
- [23] B. K. Mishra, "Research trends in deep learning and machine learning for cloud computing security," *Artif. Intell. Rev.*, vol. 57, p. 146, 2024.
- [24] O. A. Garreche, S. K. Bruce, and M. Pang, "XAI-IDS: Toward proposing an explainable artificial intelligence framework for enhancing network intrusion detection systems," *Appl. Sci.*, vol. 14, no. 10, p. 4170, 2024.
- [25] A. Kheddar, A. Jaouen, and P. Music, "Deep transfer learning for intrusion detection in industrial control networks: A comprehensive review," *J. Netw. Comput. Appl.*, vol. 220, p. 103760, 2023.
- [26] M. Herman et al., "NIST cloud computing forensic reference architecture," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, NIST Special Publication 800-201, 2024.

- [27] A. Akilal and M.-T. Kechadi, "Cloud digital forensic readiness: An open source approach to law enforcement request management," *arXiv preprint arXiv:2507.04174*, 2025.
- [28] P. R. Shukla and V. M. Patil, "A comprehensive review of frameworks for achieving interoperability in multi-cloud environments," in *Proc. 2023 2nd Int. Conf. Informatics (ICI)*, IEEE, 2023, pp. 1–6.
- [29] M. A. M. Farzaan, M. C. Ghanem, A. El-Hajjar, and D. N. Ratnayake, "AI-enabled system for efficient and effective cyber incident detection and response in cloud environments," *arXiv preprint arXiv:2404.05602*, 2024.
- [30] D. Dunsin, M. C. Ghanem, K. Ouazzane, and V. Vassilev, "A comprehensive analysis of the role of artificial intelligence and machine learning in modern digital forensics and incident response," *Forensic Sci. Int.: Digit. Investig.*, vol. 48, p. 301675, 2024.