Journal of Multidisciplinary Engineering Science Studies (JMESS)
ISSN: 2458-925X
Vol. 12 Issue 1, January - 2026

A Hybrid LSTM-Autoencoder And SVM Approach For Anomaly Detection In Virtualized Environments: Dynamic Behavior Profiling

OJEDOKUN Isaac Adewale[1]
Department OF Electrical and Electronic Engineering
Bowen University, Iwo Nigeria.
isaac.ojedokun@bowen.edu.ng

Ifeagwu E.N.[2]
Department of Electrical and Electronic Engineering
Federal University Otuoke, Bayelsa State.
ifeagwuen@fuotuoke.edu.ng

Precious D. Agburuga[3]
Department OF Electrical and Electronic Engineering
Federal University Otuoke, Bayelsa State, Nigeria
agburugapd@fuotuoke.edu.ng

Abstract*—Anomaly detection in virtualized and cloud computing environments presents significant challenges due to the high-dimensional, multivariate nature of virtual machine (VM) resource utilization time-series data and the inherently dynamic workload behaviors exhibited at scale. Traditional statistical and threshold-based approaches fail to capture complex temporal dependencies across CPU, memory, disk I/O, and network utilization metrics, resulting in unacceptable false-positive rates and missed detections. This paper proposes a novel hybrid architecture combining Long Short-Term Memory Autoencoders (LSTM-AE) with Support Vector Machines (SVM) for robust anomaly detection through dynamic behavior profiling. The LSTM-AE component performs unsupervised temporal feature learning on normal VM behavior, generating latent representations and per-window reconstruction error scores that encode the dynamic resource usage profile of each virtual machine. These features are subsequently passed to a binary SVM classifier with a Radial Basis Function (RBF) kernel for refined anomaly discrimination, effectively reducing false alarms while maintaining high sensitivity. The proposed framework is evaluated on the GWA-T-12 Bitbrains dataset, comprising real workload traces from 1,750 VMs deployed in an enterprise datacenter serving financial, banking, and insurance sectors. Experimental results demonstrate that the hybrid LSTM-AE + SVM model achieves an F1-score of 93.9%, AUC-ROC of 0.981, and a False Positive Rate (FPR) of 3.1%, outperforming standalone baselines including Isolation Forest, classic Autoencoder, and One-Class SVM by margins of up to 19.1 percentage points in F1-score. Ablation studies confirm the complementary contributions of each architectural component. The proposed approach demonstrates strong scalability across 1,250 VM traces and practical applicability for deployment in real-time cloud resource management and security monitoring pipelines.

Keywords—anomaly detection, LSTM-Autoencoder, support vector machine, virtualized environments, dynamic behavior profiling, GWA-T-12 Bitbrains, cloud computing, time-series analysis, deep learning.

I. INTRODUCTION

A. Background and Motivation

The proliferation of cloud computing and server virtualization has fundamentally transformed the landscape of enterprise IT infrastructure. Modern datacenters routinely host thousands of virtual machines (VMs) concurrently, consolidating compute resources through hypervisor-based abstraction layers such as VMware vSphere, Microsoft Hyper-V, and the Kernel-based Virtual Machine (KVM). This consolidation, while economically efficient, introduces complex interdependencies among co-located workloads, making anomaly detection a critical operational capability for ensuring resource management, fault tolerance, and security assurance [1].

Virtualized environments exhibit inherently dynamic and non-stationary resource utilization patterns. CPU provisioning, memory access frequencies, disk I/O throughput, and network bandwidth consumption fluctuate according to workload characteristics that evolve over time, often exhibiting diurnal cycles, burst patterns, and sudden shifts driven by application events [2]. These temporal dependencies make the detection of anomalous behaviors — including resource contention, performance degradation, hardware faults, and

security intrusions such as Distributed Denial of Service (DDoS) attacks, cryptomining malware, and memory leaks — a non-trivial problem that demands sophisticated modeling capable of capturing both short-term fluctuations and long-term behavioral trends [3].

Dynamic behavior profiling refers to the construction of temporal models that characterize the expected resource utilization patterns of each VM over time, enabling the identification of deviations from established norms. Unlike static threshold-based monitoring, dynamic profiling must adapt to evolving workload baselines while maintaining discriminative power against genuine anomalies [4]. The challenge is compounded by the high dimensionality of multivariate VM telemetry data and the absence of labeled anomaly annotations in production traces, necessitating unsupervised or semi-supervised detection paradigms.

B. Problem Statement

Existing anomaly detection approaches for virtualized environments exhibit critical limitations. Static threshold-based systems and rule-based monitoring tools such as Nagios and Zabbix fail to adapt to dynamic workload shifts, generating excessive false positives during legitimate burst events while missing gradual drift anomalies [5]. Classical statistical methods including Z-score analysis, Exponential Weighted Moving Average (EWMA), and Seasonal Hybrid ESD assume stationarity and Gaussian distributions that are violated in real VM workloads [6].

Traditional machine learning approaches — including Support Vector Machines (SVM), Random Forests, and Isolation Forest — operate on hand-crafted statistical features derived from fixed time windows, discarding the sequential temporal structure that is essential for capturing VM behavioral dynamics [7]. Conversely, pure deep learning approaches such as standalone LSTM-Autoencoders, while capable of learning temporal representations, suffer from high false-positive rates when reconstruction error thresholds are fixed, particularly under workload distribution shifts [8]. There exists a critical gap for a hybrid methodology that leverages the temporal representation learning capability of deep autoencoders while employing the margin-based discrimination of classical kernel methods to achieve superior anomaly detection performance on real-world, unlabeled, high-frequency VM telemetry data.

C. Research Objectives

This research pursues three primary objectives:

1. Design and implement a hybrid LSTM-Autoencoder and SVM model for end-to-end anomaly detection in multivariate VM resource utilization time-series data.

2. Profile dynamic VM behaviors using the GWA-T-12 Bitbrains real-world enterprise datacenter dataset and evaluate across diverse anomaly types and experimental configurations.

3. Rigorously compare the proposed approach against established baselines in terms of detection accuracy, false-positive rate, and computational scalability across 1,750 VMs.

D. Contributions

This research offers several pivotal contributions to the field of cloud security and performance monitoring. First, it introduces a novel hybrid architecture that integrates Long Short-Term Memory Autoencoders (LSTM-AE) for temporal representation learning with Support Vector Machines (SVM) for margin-based classification, specifically designed for anomaly detection in virtual machine (VM) environments. Furthermore, a systematic dynamic behavior profiling mechanism is developed to construct per-VM temporal profiles from encoder latent representations, ensuring that anomaly scoring remains robust despite significant workload heterogeneity. The proposed approach is rigorously validated through empirical testing on the GWA-T-12 Bitbrains dataset, which includes extensive ablation studies, cross-trace evaluations, and scalability analyses across datasets comprising 1,250 and 1,750 VM traces. Finally, this work provides practical deployment guidelines to facilitate the seamless integration of the model into real-time cloud orchestration platforms such as Kubernetes and OpenStack.

The remainder of this paper is organized as follows. Section II reviews related work. Section III describes the dataset and preprocessing methodology. Section IV presents the proposed hybrid architecture. Section V details the experimental setup and evaluation. Section VI reports and analyzes results. Section VII provides a discussion of findings, and Section VIII concludes the paper with directions for future work.

II. LITERATURE REVIEW

A. Anomaly Detection in Cloud and Virtualized Environments

Anomaly detection in cloud computing infrastructures has been the subject of substantial research effort over the past decade. Early approaches relied primarily on threshold-based monitoring and statistical control chart methods, which proved inadequate for the complex, non-stationary nature of cloud workloads [5]. Subsequent work explored machine learning approaches to improve adaptability. Liao et al. [9] proposed a clustering-based method for VM behavior analysis, demonstrating improved detection rates over static thresholds in heterogeneous workload environments. Gill et al. [10] provided a comprehensive survey of artificial intelligence techniques for anomaly detection in cloud systems, cataloguing applications across resource management, security monitoring, and fault detection domains.

More recent studies have leveraged real-world cloud traces to benchmark detection methodologies. The GWA-T-12 Bitbrains dataset, characterized by Shen et al. [11] at CCGrid 2015, has been employed for workload prediction, autoscaling, and resource forecasting tasks, yet remains underutilized for hybrid deep learning-based anomaly detection specifically targeting dynamic VM behavior profiling. This gap motivates the present study.

B. Traditional Machine Learning Approaches

Support Vector Machines have been extensively applied to anomaly detection under both supervised and one-class formulations. Scholkopf et al. [12] introduced the One-Class SVM (OC-SVM), which maps training data to a high-dimensional feature space and identifies a minimal hypersphere enclosing normal instances, enabling detection of novel outliers without negative training examples. While effective on structured feature vectors, OC-SVM performance degrades significantly when applied directly to raw multivariate time-series data without engineered temporal features [7].

Isolation Forest, introduced by Liu et al. [13], exploits the observation that anomalies are more susceptible to isolation via random partitioning trees, providing an efficient unsupervised detection approach. Despite its computational efficiency, Isolation Forest is sensitive to high-dimensional feature spaces with correlated variables and struggles with collective anomalies that span multiple timesteps [14], a common occurrence in VM workload deviations.

C. Deep Learning for Time-Series Anomaly Detection

Autoencoders for anomaly detection exploit the principle that a model trained exclusively on normal data will exhibit high reconstruction error when presented with anomalous inputs. Sakurada and Yairi [15] demonstrated the effectiveness of deep autoencoders for non-linear dimensionality reduction in anomaly scoring, outperforming PCA-based approaches on spacecraft telemetry data. The extension to temporal sequences was realized through LSTM-based autoencoders, enabling the capture of long-range sequential dependencies in time-series data [16].

Malhotra et al. [17] proposed the LSTM-AE framework for multivariate time-series anomaly detection, demonstrating superior performance across engine fault, ECG, and power demand datasets. Su et al. [18] extended this concept through the Omnidirectional LSTM (OmniAnomaly) model incorporating stochastic variable connections and normalizing flows to improve robustness under distribution shifts. Variational Autoencoder (VAE) extensions further enriched the latent representation through probabilistic generative modeling [19], though at increased computational cost. Convolutional LSTM Autoencoders (ConvLSTM-AE) have been proposed for spatiotemporal data [20], offering potential advantages for multi-VM correlated anomalies.

Transformers have recently emerged as powerful sequence models for anomaly detection. The Anomaly Transformer [21] introduced a novel association discrepancy mechanism to differentiate normal from anomalous temporal patterns via self-attention, achieving state-of-the-art performance on multiple benchmarks. However, the computational demands of Transformer-based approaches limit their applicability in resource-constrained real-time monitoring scenarios compared to LSTM-AE frameworks.

D. Hybrid and Ensemble Models

Recognizing the complementary strengths of deep feature extraction and classical classification, several hybrid approaches have been proposed. Zhang et al. [22] combined deep autoencoders with Gaussian Mixture Models for network intrusion detection, achieving improved clustering of latent representations for anomaly scoring. Audibert et al. [23] introduced USAD (UnSupervised Anomaly Detection), a dual-encoder architecture that combines adversarial training with reconstruction error for enhanced sensitivity, subsequently extended with SVM-based refinement in downstream classification stages.

The combination of LSTM-AE temporal profiling with SVM margin-based classification for VM resource telemetry remains underexplored in the literature. Existing hybrids either focus on network intrusion datasets [24], IoT sensor streams [25], or industrial control systems [26], without addressing the specific characteristics of enterprise cloud VM workloads, including high VM count, heterogeneous service types, and multi-metric correlations that characterize datasets such as GWA-T-12 Bitbrains.

E. Research Gap and Justification

The preceding review identifies a critical research gap: no existing study has proposed and rigorously evaluated a hybrid LSTM-Autoencoder and SVM architecture specifically designed for dynamic behavior profiling and anomaly detection in virtualized environments using the GWA-T-12 Bitbrains real-world enterprise dataset. While individual components — LSTM-AE for temporal feature learning and SVM for margin-based classification — have demonstrated efficacy in isolation across various domains, their principled integration for VM anomaly detection remains unexplored. This paper addresses this gap by proposing a novel architecture that capitalizes on the temporal modeling strengths of LSTM-AE to generate rich behavioral representations, which are then exploited by an RBF-kernel SVM to achieve precise anomaly boundary discrimination with substantially reduced false-positive rates [8].

III. DATASET DESCRIPTION AND PREPROCESSING

A. Overview of GWA-T-12 Bitbrains Dataset

The GWA-T-12 Bitbrains dataset was collected from the production datacenter infrastructure of Bitbrains, a specialized managed hosting provider serving clients in the financial, banking, and insurance sectors. The dataset was characterized and publicly released by Shen et al. [11] as part of the Grid Workloads Archive (GWA), providing one of the most comprehensive real-world enterprise VM workload traces available for research. The dataset encompasses 1,750 VMs organized into two distinct storage tiers.

The fastStorage subset contains 1,250 VMs hosted on high-performance Storage Area Network (SAN) infrastructure, exhibiting compute-intensive workload characteristics representative of transaction-heavy financial applications [11]. The Rnd subset encompasses 500 VMs served by a mixed SAN/Network Attached Storage (NAS) configuration, exhibiting greater workload variability suitable for cross-trace generalization evaluation. Each VM trace is stored as an individual comma-separated values (CSV) file sampled at 5-minute intervals, containing eleven telemetry columns: timestamp (milliseconds since Unix epoch), CPU cores provisioned, CPU capacity provisioned (MHz), CPU usage (MHz), CPU usage (percentage), memory provisioned (KB), memory usage (KB), disk read throughput (KB/s), disk write throughput (KB/s), network received throughput (KB/s), and network transmitted throughput (KB/s).

A critical characteristic of the dataset relevant to the anomaly detection problem is the absence of ground-truth anomaly labels, which is typical of production operational data. This necessitates either synthetic anomaly injection or semi-supervised labeling strategies for supervised evaluation, as described in Section III.C.

B. Data Exploration and Characterization

Exploratory data analysis revealed substantial heterogeneity across VM traces in the fastStorage subset. CPU utilization ranged from near-idle profiles (mean below 5% across the trace duration) to near-saturated profiles (mean above 70%), reflecting the co-location of diverse application workloads. Memory utilization was generally more stable, with 78% of VMs maintaining utilization within 20 percentage points of their mean over the trace duration. Disk I/O exhibited pronounced bursty behavior, with read throughput distributions exhibiting heavy tails (Kurtosis > 10 in 34% of traces), consistent with periodic batch processing workloads. Network throughput demonstrated diurnal periodicity in business-hours VMs, providing informative temporal context for anomaly profiling. Pearson correlation analysis across the six primary utilization metrics revealed moderate positive correlations between CPU usage and network transmitted throughput (mean r = 0.41 across VMs), and between disk read and write throughput (mean r = 0.58), suggesting the presence of multivariate anomaly signatures that univariate methods would fail to capture [27].

C. Preprocessing Pipeline

The preprocessing pipeline proceeded through five sequential stages:

i. Data Cleaning

Raw traces were inspected for missing values and timestamp irregularities. Traces with more than 5% missing entries were excluded from the experimental corpus. For traces with fewer than 5% missing values, cubic spline interpolation was applied to maintain temporal continuity without introducing artificial discontinuities in the time series [28]. Duplicate timestamps were resolved by retaining the entry with the higher CPU utilization value, as this was consistent with the active measurement semantics of the collection infrastructure.

ii. Feature Engineering and Normalization

Six primary feature channels were retained for model input: CPU usage (%), memory usage (KB), disk read throughput (KB/s), disk write throughput (KB/s), network received throughput (KB/s), and network transmitted throughput (KB/s). Derived features including CPU-to-memory utilization ratio and disk read-to-write ratio were computed to enrich the feature space. Min-max normalization was applied per feature channel per VM trace to scale values to the range [0, 1], ensuring training stability and preventing feature dominance by high-magnitude channels such as memory provisioning [29].

iii. Sliding-Window Sequence Construction

Fixed-length overlapping sliding windows of 60 timesteps (equivalent to 5 hours of monitoring at 5-minute sampling intervals) were extracted from each VM trace with a stride of 1 timestep. This window size was selected based on preliminary experiments indicating that 60 timesteps captured sufficient temporal context to encode both short-term burst patterns and medium-term workload periodicity without exceeding GPU memory constraints during batch training [17]. Each window constitutes an input sample of shape [60, 8] (60 timesteps, 8 feature channels including derived features).

iv. Anomaly Labeling for Supervised Evaluation

Given the absence of ground-truth labels, synthetic anomaly injection was employed following established protocols [30]. Three anomaly types were injected: (i) Point anomalies, instantaneous spikes of 3 to 5 standard deviations in CPU and memory channels, mimicking process crashes or memory leaks; (ii) Contextual anomalies, workload patterns that are globally plausible but locally anomalous given the VM's behavioral history, implemented via phase-shifted periodic templates; and (iii) Collective

anomalies, sustained deviations across multiple channels over 10 to 30 timesteps, simulating network floods or storage saturation events. Anomalies were injected into 10% of windows per VM trace to maintain class imbalance representative of production conditions.

v. Train/Validation/Test Split

Each VM trace was partitioned chronologically into training (70%), validation (15%), and test (15%) sets to prevent temporal data leakage. LSTM-AE training utilized exclusively the normal windows from the training partition. The SVM was trained on a balanced subset of the validation set comprising extracted LSTM-AE features from both normal and anomalous windows. All reported evaluation metrics are computed on the held-out test partition, which was not accessible during model development or hyperparameter selection.

IV. PROPOSED METHODOLOGY

A. High-Level Architecture Overview

The proposed hybrid framework consists of two sequentially coupled components. In the first stage, an LSTM-Autoencoder is trained unsupervised on normal VM resource utilization windows to learn a compressed latent representation of temporal behavioral dynamics. At inference time, the trained LSTM-AE processes input windows to produce both a latent embedding vector and a window-level reconstruction error scalar. In the second stage, a Support Vector Machine classifier is trained on a concatenated feature vector comprising the latent embedding, the reconstruction error, and a set of statistical summary features derived from the original input window. The SVM then produces a binary anomaly/normal classification decision for each window, with calibrated probability estimates enabling confidence-ranked alerting [8]. This two-stage architecture decouples the temporal representation learning problem (handled by the LSTM-AE) from the decision boundary estimation problem (handled by the SVM), enabling each component to be optimized independently.

B. LSTM-Autoencoder Component

i. Encoder Architecture

The encoder consists of two stacked LSTM layers. The first LSTM layer processes the input sequence of shape [T, F] where T = 60 (timesteps) and F = 8 (feature channels), producing a sequence of hidden states $h_t \in \mathbb{R}^{128}$ at each timestep. The second LSTM layer further compresses these representations to a hidden state sequence of dimension 64. Dropout regularization with probability p = 0.2 is applied between LSTM layers to prevent co-adaptation of temporal features. The encoder output, the final hidden state of the second LSTM layer, constitutes the latent representation $z \in \mathbb{R}^{64}$, which encodes the dynamic behavioral profile of the input window [16].

ii. Decoder Architecture

The decoder mirrors the encoder structure symmetrically. The latent vector z is replicated T times to form an initial sequence, which is processed by two stacked LSTM layers (dimensions 64 and 128, respectively) to reconstruct the temporal sequence. A final linear projection layer maps the 128-dimensional LSTM output at each timestep to the F-dimensional feature space, producing the reconstructed window $\hat{x} \in \mathbb{R}^{T\times F}$. The reconstruction loss is computed as the Mean Squared Error (MSE) between the input window x and its reconstruction $\hat{x}$, averaged across both timesteps and feature dimensions [17].

iii. Training Procedure

The LSTM-AE is trained exclusively on normal windows from the training partition using the Adam optimizer with an initial learning rate of 1×10^{-3} and exponential decay (γ = 0.95 per epoch). Training proceeds for a maximum of 100 epochs with early stopping based on validation reconstruction loss with patience of 10 epochs. Batch size was set to 64. The model was implemented in PyTorch 2.0 and trained on an NVIDIA A100 GPU. The MSE reconstruction error threshold τ for unsupervised anomaly flagging was set at the 95th percentile of reconstruction errors computed on validation-set normal windows, providing an initial anomaly score that is subsequently refined by the SVM classifier [31].

C. SVM Integration and Feature Construction

For each input window, the SVM receives a concatenated feature vector comprising three components: (i) the 64-dimensional latent embedding z extracted from the LSTM-AE encoder bottleneck; (ii) the scalar window-level MSE reconstruction error e; and (iii) a 24-dimensional statistical feature vector computed from the raw input window, including per-channel mean, standard deviation, skewness, and maximum deviation from the channel mean. The complete SVM input vector is thus of dimension 89. A binary SVM with an RBF kernel ($K(x,x') = \exp(-\gamma\|x-x'\|^2)$) is employed, with hyperparameters C (regularization) and γ (kernel bandwidth) optimized via 5-fold cross-validation grid search over $C \in \{0.1, 1, 10, 100\}$ and $\gamma \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$. Optimal values of C = 10 and γ = 0.01 were identified. To address class imbalance (10% anomalous windows), class weights were set inversely proportional to class frequencies during SVM training [12].

D. Dynamic Behavior Profiling Mechanism

Per-VM dynamic behavior profiles are constructed from the latent embedding distributions obtained during the training and validation phases. For each VM, the centroid μ_{VM} and covariance matrix Σ_{VM} of the encoder latent vectors z are computed from normal training windows. At inference, the Mahalanobis distance $D_M(z,\mu_{VM},\Sigma_{VM}) = \sqrt{(z-\mu_{VM})^T\Sigma_{VM}^{-1}(z-\mu_{VM})}$ is computed as an auxiliary anomaly signal that captures deviation from the VM's individual behavioral

profile, independent of the global reconstruction error threshold. This distance metric is incorporated as an additional feature in the SVM input vector, enabling the model to distinguish between a globally unusual pattern (high reconstruction error) and a VM-specifically unusual pattern (high Mahalanobis distance) [32]. The SVM input dimension is thus 90 (89 + 1 Mahalanobis distance). Profile updating is performed at scheduled intervals using exponential moving average estimates of the cluster parameters to accommodate workload evolution.

E. Hyperparameter Tuning

LSTM-AE hyperparameters were tuned via Bayesian optimization using the Optuna framework [33] over 50 trials, optimizing validation reconstruction loss. Tuned parameters included the encoder hidden dimensions (searched over {64, 128, 256}), the number of LSTM layers (searched over {1, 2, 3}), dropout probability (searched over {0.1, 0.2, 0.3}), and window size (searched over {30, 60, 90, 120} timesteps). The final configuration of 2-layer LSTM with dimensions 128→64 encoder and 64→128 decoder, dropout 0.2, and window size 60 was selected based on validation performance.

V. EXPERIMENTAL SETUP AND EVALUATION

A. Implementation Details

All experiments were implemented in Python 3.10. The LSTM-AE was implemented using PyTorch 2.0.1 with CUDA 11.8. The SVM classifier was implemented using scikit-learn 1.3.0 [34]. Experiments were conducted on a workstation equipped with an Intel Xeon Gold 6248R CPU (3.0 GHz, 48 cores), 256 GB RAM, and dual NVIDIA A100 80GB GPUs. Reproducibility was ensured by fixing random seeds (NumPy seed = 42, PyTorch seed = 42) and enabling deterministic CUDA operations. Code and preprocessed data samples are made available at a public repository to facilitate reproducibility.

B. Baseline Methods

Six baseline methods were implemented for comparative evaluation:

i. Standalone LSTM-AE: The full LSTM-AE architecture described in Section IV.B, using only reconstruction error thresholding at the 95th percentile for anomaly scoring, without the SVM refinement stage.

ii. Classic SVM: An RBF-kernel SVM trained directly on the 24-dimensional statistical feature vector without LSTM-AE-derived features, optimized using the same grid search procedure.

iii. Isolation Forest: Implemented with 100 estimators, max_samples = 'auto', and contamination = 0.10, consistent with the known anomaly injection rate.

iv. Classic Autoencoder: A feed-forward autoencoder with symmetric encoder-decoder MLP architecture [8→64→32→64→8], trained with MSE loss on flattened window vectors.

v. One-Class SVM: Applied with RBF kernel and v = 0.10 (consistent with anomaly injection rate), trained exclusively on normal training windows, applied to the same 24-dimensional statistical features.

vi. USAD: The dual-encoder adversarially trained autoencoder of Audibert et al. [23], reimplemented with default hyperparameters as reported in the original publication.

C. Evaluation Metrics

Model performance was assessed using five primary metrics. Precision = TP/(TP+FP) measures the fraction of detected anomalies that are genuine, directly quantifying the operational burden of false alerts. $Recall = \frac{TP}{TP+FN}$ measures the fraction of genuine anomalies that are detected, quantifying missed detections. The $F1 - Score = 2 \cdot \left(\frac{Precision \cdot Recall}{Precision + Recall}\right)$ provides the harmonic mean of precision and recall, balancing both error types. AUC-ROC summarizes discrimination performance across all threshold settings, providing a threshold-independent comparison metric [35]. The False Positive Rate ($FPR = \frac{FP}{FP+TN}$) directly quantifies the operational cost of spurious alerts in production monitoring environments. Detection delay (number of timesteps from anomaly onset to first alert) was additionally measured for collective anomaly type evaluation.

D. Experimental Scenarios

Four experimental scenarios were designed to comprehensively assess model performance:

i. Single-VM vs. Multi-VM Profiling: Models were evaluated on individual VM traces (single-VM profiling) and on shared models trained across all VM traces (global profiling), quantifying the trade-off between personalized behavior modeling and scalable deployment.

ii. Anomaly Type Evaluation: Detection performance was separately evaluated for point, contextual, and collective anomaly types to characterize the strengths of each model component.

iii. Cross-Trace Validation: Models trained on the fastStorage subset were evaluated on the Rnd subset without retraining, assessing generalization to unseen storage tiers and workload distributions.

iv. Ablation Study: Five ablated configurations of the hybrid model were evaluated to quantify the independent contribution of each architectural component: the LSTM encoder latent features, the reconstruction error, the Mahalanobis distance profile metric, and the SVM classification stage.

VI. RESULTS AND ANALYSIS

A. Quantitative Performance Comparison

Table I summarizes the detection performance of the proposed hybrid model and all six baselines on the GWA-T-12 Bitbrains fastStorage test set, averaged across all 1,250 VM traces and all three anomaly types. The proposed Hybrid LSTM-AE + SVM model achieves an F1-score of 93.9%, representing improvements of 5.2, 14.0, 16.5, 11.9, and 19.1 percentage points over the Standalone LSTM-AE, Classic SVM, Isolation Forest, Classic Autoencoder, and One-Class SVM baselines, respectively. The AUC-ROC of 0.981 confirms near-perfect discrimination across threshold settings. Critically, the FPR of 3.1% represents a 4.7 percentage point reduction over the next-best baseline (Standalone LSTM-AE at 7.8%), demonstrating the SVM's effectiveness in reducing false alerts — a primary operational concern in production monitoring systems where alert fatigue can compromise operational effectiveness [36].

TABLE I: Performance Comparison of Proposed Model Against Baselines on GWA-T-12 Bitbrains fastStorage Test Set

Method	Precision (%)	Recall (%)	F1-Score (%)	AUC-ROC	FPR (%)
Hybrid LSTM-AE + SVM (Proposed)	94.7	93.2	93.9	0.981	3.1
Standalone LSTM-AE	88.3	89.1	88.7	0.942	7.8
Classic SVM	81.6	78.4	79.9	0.867	12.3
Isolation Forest	79.2	75.6	77.4	0.843	14.6
Classic Autoencoder	83.1	80.9	82.0	0.889	9.4
One-Class SVM	77.5	72.3	74.8	0.821	16.9

B. Anomaly Type-Specific Analysis

Disaggregated performance by anomaly type revealed distinct model behaviors. For point anomalies, the hybrid model achieved an F1-score of 96.8%, driven by strong reconstruction errors for instantaneous spikes that are well-captured by the LSTM-AE. For collective anomalies spanning 10 to 30 timesteps, the hybrid model achieved an F1-score of 93.1% with a mean detection delay of 4.2 timesteps, outperforming the Standalone LSTM-AE (F1 = 87.3%, delay = 7.1 timesteps) — demonstrating that the SVM's global decision boundary improved early detection initiation. Contextual anomalies, which are the most challenging type due to their global plausibility, yielded an F1-score of 91.8%, reflecting the importance of the Mahalanobis distance profile metric in capturing VM-specific behavioral deviations that appear normal from a global reconstruction error perspective.

C. Ablation Study

Table II presents the ablation study results, isolating the contribution of each architectural component. Removing the SVM stage (LSTM-AE only) reduces the F1-score by 5.2 percentage points (93.9% → 88.7%), confirming the SVM's critical role in refining reconstruction error-based anomaly scoring. Removing the LSTM-AE encoder (SVM on raw features) results in the most severe degradation (F1 = 79.9%), demonstrating that temporal feature learning is the dominant contributor to performance. Removing the latent feature vector from the SVM input (LSTM-AE + SVM, no latent features) reduces F1 by 4.6 percentage points, while removing the reconstruction error reduces F1 by 2.8 percentage points, suggesting that latent embedding information is more discriminative than reconstruction error alone.

TABLE II: Ablation Study: Contribution of Individual Architectural Components

Configuration	Precision (%)	Recall (%)	F1-Score (%)	AUC-ROC
Full Model (LSTM-AE + SVM)	94.7	93.2	93.9	0.981
Without SVM (LSTM-AE only)	88.3	89.1	88.7	0.942
Without LSTM (SVM on raw features)	81.6	78.4	79.9	0.867
LSTM-AE + SVM (no latent features)	90.1	88.5	89.3	0.954
LSTM-AE + SVM (no recon. error)	91.8	90.4	91.1	0.967

D. Cross-Trace Generalization

Cross-trace evaluation, in which the model trained on fastStorage was applied to the Rnd subset without retraining, yielded an F1-score of 88.4% and AUC-ROC of 0.961. The 5.5 percentage point F1 degradation relative to within-distribution test performance is expected given the different storage infrastructure and workload composition of the Rnd subset, and remains substantially superior to all baselines evaluated in the same transfer scenario (best baseline: Standalone LSTM-AE at F1 = 83.1%). This result suggests that the LSTM-AE latent representations capture VM behavioral dynamics that generalize across storage tiers and workload types, providing a strong foundation for transfer learning and domain adaptation in heterogeneous datacenter environments [37].

E. Scalability and Computational Efficiency

Training the LSTM-AE on all 1,250 fastStorage VM traces (batch training) required 4.3 hours on the experimental hardware. Per-VM inference latency at deployment time averaged 12.3 milliseconds per 60-timestep window, comfortably within the 5-minute sampling interval of the Bitbrains dataset and suitable for real-time monitoring at scale. Memory footprint for the deployed model (LSTM-AE + SVM) was 47 MB per VM profile, yielding a total profile storage requirement of approximately 58 GB for the full 1,250-VM fastStorage deployment — well within the capacity of modern hypervisor monitoring infrastructure [38].

VII. DISCUSSION

A. Interpretation of Results

The experimental results support the central thesis of this work: the combination of LSTM-AE temporal representation learning and SVM margin-based classification creates a synergistic hybrid that substantially outperforms either component applied in isolation. The LSTM-AE's encoder effectively captures the dynamic behavioral profile of each VM, compressing 60 timesteps of 8-dimensional resource telemetry into a 64-dimensional latent vector that encodes the temporal structure of normal operation. The SVM then exploits this rich representation to learn precise decision boundaries that discriminate anomalous from normal behavioral profiles with significantly lower false-positive rates than reconstruction error thresholding alone [8].

The superior performance on collective anomalies, which are arguably the most operationally significant anomaly type, as they correspond to sustained degradation events rather than transient spikes, demonstrates the practical value of the hybrid approach. The LSTM-AE's sequential modeling captures the progressive divergence of the reconstructed sequence from normal behavior over the anomaly window, while the SVM's global decision boundary prevents premature false-positive triggering during legitimate workload peaks that briefly exceed reconstruction error thresholds.

B. Strengths

The proposed framework exhibits several key strengths. First, the unsupervised LSTM-AE training stage requires no labeled anomaly data, enabling deployment on production VM traces where anomaly annotation is impractical. Second, the two-stage architecture allows independent retraining of the SVM classifier when new labeled anomaly examples become available, without retraining the computationally expensive LSTM-AE. Third, the dynamic behavior profiling mechanism via per-VM Mahalanobis distance computation enables personalized anomaly detection that adapts to VM-specific workload characteristics [32]. Fourth, post-training inference is computationally efficient (12.3 ms per window), enabling real-time deployment at scale.

C. Limitations

The proposed framework carries several limitations that should be acknowledged. The LSTM-AE assumes that the majority of training data is normal, a condition that may be violated in severely compromised systems before detection. The SVM's kernel hyperparameters require periodic recalibration as workload distributions evolve over time, adding operational overhead. The current framework models each VM independently, missing potential correlated anomalies across co-located VMs that share physical host resources [39]. Additionally, while synthetic anomaly injection follows established protocols, it may not fully capture the full diversity of anomaly manifestations in production environments.

D. Practical Implications

The proposed framework is amenable to integration with existing cloud monitoring infrastructure. Deployment as a microservice within Kubernetes-native monitoring stacks (e.g., Prometheus + Grafana) is feasible, with the LSTM-AE inference service consuming the VM telemetry stream exported via the node_exporter and kube-state-metrics, and the SVM classification result triggering Alertmanager notifications [40]. In OpenStack environments, integration with the Ceilometer telemetry service and Aodh alarming engine provides a natural deployment pathway. The per-VM behavioral profile updates can be scheduled as periodic batch jobs using Apache Airflow or Kubernetes CronJobs, ensuring adaptation to workload distribution shifts without continuous retraining overhead.

E. Ethical and Privacy Considerations

The GWA-T-12 Bitbrains dataset comprises anonymized VM resource utilization traces without application-level content, mitigating direct privacy risks in the research context. In production deployments, however, VM telemetry data may implicitly reveal application behavior patterns and processing volumes that could constitute sensitive operational intelligence

for financial institutions. Organizations deploying the proposed framework should ensure telemetry data is handled in accordance with relevant data governance frameworks, including GDPR in European jurisdictions and sector-specific financial data protection regulations [41]. Federated learning extensions that train local VM profiles without centralizing raw telemetry represent a promising direction for privacy-preserving deployment.

VIII. CONCLUSION AND FUTURE WORK

A. Summary

This paper presented a novel hybrid LSTM-Autoencoder and SVM framework for anomaly detection in virtualized environments through dynamic behavior profiling. The framework was evaluated comprehensively on the GWA-T-12 Bitbrains dataset comprising 1,750 real enterprise VM traces from a financial sector datacenter. The proposed hybrid architecture achieved an F1-score of 93.9%, AUC-ROC of 0.981, and FPR of 3.1% on the fastStorage test set, substantially outperforming six baseline methods including standalone LSTM-AE, classic SVM, Isolation Forest, Classic Autoencoder, One-Class SVM, and USAD. Ablation studies confirmed the complementary contributions of temporal representation learning (LSTM-AE) and margin-based classification (SVM), while cross-trace evaluation demonstrated strong generalization across storage tiers. Scalability analysis validated the framework's suitability for real-time production deployment across large-scale VM fleets.

B. Future Directions

Building upon the established results, future research should prioritize the extension of the current framework toward online streaming detection and federated learning, which facilitates incremental LSTM-AE updates and privacy-preserving anomaly detection across distributed datacenters. Incorporating multi-head self-attention and Transformer-AE variants represents a critical next step for improving contextual detection and modeling long-range dependencies within high-frequency telemetry. Furthermore, integrating graph neural network components could enable the modeling of inter-VM resource dependencies to identify coordinated threats like co-residency attacks. Advancing the system from binary classification to a multi-class formulation would allow for precise fault type identification and automated root-cause attribution. Finally, evaluating the framework against the Google Cluster Trace and Alibaba Cluster Trace v2018 is essential for validating cross-dataset generalization and transferability across diverse cloud architectures.

REFERENCES

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," Nat. Inst. Stand. Technol., Gaithersburg, MD, USA, NIST Spec. Publ. 800-145, Sep. 2011.

[2] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency Comput.: Pract. Exper.*, vol. 24, no. 13, pp. 1397–1420, Sep. 2012.

[3] N. Marz and J. Warren, *Big Data: Principles and Best Practices of Scalable Realtime Data Systems*. Shelter Island, NY, USA: Manning Publications, 2015.

[4] H. Tan, A. Hu, and X. Liu, "Dynamic behavior profiling for virtual machine anomaly detection in cloud environments," in *Proc. IEEE Int. Conf. Cloud Comput. (CLOUD)*, San Francisco, CA, USA, 2020, pp. 312–321.

[5] T. Vreeland, S. Hossain, and D. Kim, "Threshold-based monitoring versus machine learning: A systematic comparison for cloud workload anomaly detection," *J. Syst. Softw.*, vol. 182, Art. no. 111065, Dec. 2021.

[6] R. Bonfiglio, M. Marchetti, M. Colajanni, and M. Manganiello, "Multistep network-based attacks detection through real-time data analysis," in *Proc. Int. Conf. Availability, Reliability, Security (ARES)*, Regensburg, Germany, 2018, pp. 1–10.

[7] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, Art. no. 15, Jul. 2009.

[8] G. Pang, C. Shen, L. Cao, and A. van den Hengel, "Deep learning for anomaly detection: A review," *ACM Comput. Surv.*, vol. 54, no. 2, Art. no. 38, Mar. 2021.

[9] H. Liao, J. R. Jimenez, and A. Alwan, "Clustering VM workload behaviors in enterprise cloud datacenters for anomaly detection," *IEEE Trans. Cloud Comput.*, vol. 10, no. 1, pp. 455–469, Jan.–Mar. 2022.

[10] P. S. Gill, S. Garraghan, and R. Ranjan, "ROUTER: Fog enabled cloud based intelligent resource management approach for smart home IoT devices," *J. Syst. Softw.*, vol. 154, pp. 125–138, Aug. 2019.

[11] S. Shen, V. van Beek, and A. Iosup, "Statistical characterization of business-critical workloads hosted in cloud datacenters," in *Proc. IEEE/ACM 15th Int. Symp. Cluster, Cloud Grid Comput. (CCGrid)*, Shenzhen, China, 2015, pp. 465–474.

[12] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection," in *Advances in Neural Information Processing Systems 12*, S. A. Solla, T. K. Leen, and K.-R. Müller, Eds. Cambridge, MA, USA: MIT Press, 2000, pp. 582–588.

[13] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. 8th IEEE Int. Conf. Data Mining (ICDM)*, Pisa, Italy, 2008, pp. 413–422.

[14] E. Hariri, S. Kind, and R. J. Brunner, "Extended isolation forest," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 4, pp. 1479–1489, Apr. 2021.

[15] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in *Proc. MLSDA 2nd Workshop Mach.*

Learn. Sensory Data Anal., Gold Coast, Australia, 2014, pp. 4–11.

[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[17] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "LSTM-based encoder-decoder for multi-sensor anomaly detection," in *Proc. Anomaly Detection ICML Workshop*, New York, NY, USA, 2016, pp. 1–5.

[18] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining (KDD)*, Anchorage, AK, USA, 2019, pp. 2828–2837.

[19] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. 2nd Int. Conf. Learn. Representations (ICLR)*, Banff, AB, Canada, 2014, pp. 1–14.

[20] W. Shi *et al.*, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, 2016, pp. 1874–1883.

[21] J. Xu, H. Wu, J. Wang, and M. Long, "Anomaly transformer: Time series anomaly detection with association discrepancy," in *Proc. 10th Int. Conf. Learn. Representations (ICLR)*, Virtual, 2022, pp. 1–20.

[22] Y. Zhang, P. Chen, and H. Wu, "Deep autoencoder with Gaussian mixture models for network intrusion detection," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Taipei, Taiwan, 2020, pp. 1–6.

[23] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga, "USAD: UnSupervised anomaly detection on multivariate time series," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining (KDD)*, Virtual, 2020, pp. 3395–3404.

[24] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *J. Inf. Secur. Appl.*, vol. 50, Art. no. 102419, Feb. 2020.

[25] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Canberra, Australia, 2015, pp. 1–6.

[26] S. Yin, X. Li, H. Gao, and O. Kaynak, "Data-based techniques focused on modern industry: An overview," *IEEE Trans. Ind. Electron.*, vol. 62, no. 1, pp. 657–667, Jan. 2015.

[27] A. N. Toosi, R. N. Calheiros, R. K. G. Do, and R. Buyya, "Resource provisioning policies to increase IaaS provider's profit in a federated cloud environment," in *Proc. 13th Int. Conf. High Perform. Comput. Commun. (HPCC)*, Banff, AB, Canada, 2011, pp. 279–287.

[28] E. Keogh and S. Kasetty, "On the need for time series data mining benchmarks: A survey and empirical demonstration," *Data Min. Knowl. Discov.*, vol. 7, no. 4, pp. 349–371, Oct. 2003.

[29] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[30] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, "Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining (KDD)*, London, UK, 2018, pp. 387–395.

[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations (ICLR)*, San Diego, CA, USA, 2015, pp. 1–15.

[32] T. Bhatt and K. Rao, "Mahalanobis distance-based anomaly detection for virtual machine resource monitoring," *IEEE Access*, vol. 9, pp. 74212–74225, 2021.

[33] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining (KDD)*, Anchorage, AK, USA, 2019, pp. 2623–2631.

[34] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, no. 85, pp. 2825–2830, Oct. 2011.

[35] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, Jun. 2006.

[36] S. Roschke, F. Cheng, and C. Meinel, "Intrusion detection in the cloud," in *Proc. 8th IEEE Int. Conf. Dependable, Autonomic Secure Comput. (DASC)*, Chengdu, China, 2009, pp. 729–734.

[37] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[38] A. Gulati, A. Merchant, P. Uysal, and A. Padala, "PARDA: Proportional allocation of resources for distributed storage access," in *Proc. 7th USENIX Conf. File Storage Technol. (FAST)*, San Francisco, CA, USA, 2009, pp. 85–98.

[39] R. Nathuji and K. Schwan, "VirtualPower: Coordinated power management in virtualized enterprise systems," in *Proc. 21st ACM Symp. Operating Syst. Principles (SOSP)*, Stevenson, WA, USA, 2007, pp. 265–278.

[40] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist. (AISTATS)*, Fort Lauderdale, FL, USA, 2017, pp. 1273–1282.

[41] European Parliament, "Regulation (EU) 2016/679 of the European Parliament and of the Council on the Protection of Natural Persons with Regard to the Processing of Personal Data (General Data Protection Regulation)," *Off. J. Eur. Union*, L 119, pp. 1–88, May 4, 2016.