SvedbergOpen
DISSEMINATION OF KNOWLEDGE
International Journal of Artificial Intelligence and Machine Learning
Publisher's Home Page: https://www.svedbergopen.com/
Research Paper Open Access

Large-Scale Intelligence Microservices For Robust Anomaly Detection In Real-Time Streaming Data Systems Using Deep Learning

Somla[1], Dr. Srinivas Malladi[2]

[1]Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Hyderabad-500057, Telangana, India. Mail ID: somla.m6@gmail.com
[2] Professor, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Hyderabad-500057, Telangana, India. Mail ID: srinivas.malladi@klh.edu.in

Abstract
Cloud-native microservice architectures have grown at an unprecedented rate, imposing new and greater requirements on reliable and scalable frameworks that can analyze streaming data at very high velocities and rates and detect anomalies. This study proposes a thorough survey of large-scale intelligence microservice pipelines, incorporating cutting-edge deep learning models such as Long Short-Term Memory (LSTM) networks, transformer-based architectures, and Variational Autoencoders (VAEs) for robust anomaly detection in real-time distributed systems. Four benchmark datasets, MSCert-2023, HDFS-Log, Yahoo S5, and GCP-Trace-2022, with a total of more than 18.5 million records, were used for the evaluation. The proposed microservice-orchestrated pipeline on the MSCert-2023 dataset also outperformed the conventional baseline methods by a large margin, yielding an F1 score of 95.2% and AUC-ROC of 0.986 using the Transformer model, with an end-to-end inference latency as low as 41 ms. Experimental results demonstrate that the proposed microservice-oriented pipeline significantly outperforms conventional baseline methods in terms of performance, suggesting a deployable framework for enterprise-grade anomaly detection in distributed data streams. The results pave the way for an emerging new paradigm in which modular, containerized artificial intelligence services can be combined to solve the reliability issues in cloud computing systems.

Keywords: microservices; anomaly detection; deep learning; real-time streaming; LSTM; Transformer; Variational Autoencoder; cloud computing

1. Introduction

In the modern software ecosystem, there has been a complete paradigm shift from a single large application to a microservices-driven approach, which focuses on small, reusable, self-contained services that are deployed independently and can be scaled up or down as needed. In cloud-based systems with hundreds of loosely coupled services interacting with each other regularly, detecting anomalies in real time has become a mission-critical operational challenge. Anomaly detection, which is the capability of recognizing data points, events, or patterns that are significantly different from what is expected, is critical to ensure system reliability, service-level agreements (SLAs), and to prevent cascading failures that can spread across service meshes (Nobre et al., 2023). The impacts of any undetected anomaly in these settings can range from a poor user experience to serious financial loss, making it imperative to use automated, scalable detection systems. Although simple rule-based and statistical anomaly detection classifiers are efficient for the simpler, lower-dimensional, and stationary environments in

which they were originally developed, they are not well suited to the non-stationary, higher-dimensional, and generally complex environments of data streams produced by contemporary distributed systems. These methods are difficult to implement because they require extensive domain knowledge and are brittle in a dynamic cloud environment with unpredictable workload changes (Jia et al., 2017; Fu et al., 2009).

These monitors are especially susceptible to concept drift, which is a slow change in normal behavior over time, resulting in an increasing number of false positives and diminishing confidence in monitoring systems, as well as overwhelming the operations team. Practically and scientifically, more adaptive, data-driven detection approaches must be developed. A recent development is the introduction of deep learning, which provides a new set of powerful representation learning capabilities that can automatically learn latent features from raw telemetry streams without the need for handcrafted rules (Chalapathy & Chawla, 2019; Schmidhuber, 2015). Deep neural networks have been shown to model complex temporal dependencies, multivariate correlations, and high-dimensional feature interactions, which are difficult for traditional machine learning algorithms to handle. Specifically, recurrent models, such as LSTMs, work best for sequential anomaly detection, and transformer-based models, owing to the self-attention mechanism, have better parallelism and capture long-range dependencies. Another type of anomaly detection is unsupervised generative models, which do not require labeled examples, making them very flexible for new failure modes. A microservice-based architecture offers an ideal foundation for scaling intelligence, providing a natural substrate for scaling. These microservices can implement specialized analytical functions, ranging from data ingestion and preprocessing to model inference and alert generation, allowing the execution pipelines to run in parallel, be fault-tolerant, and be scaled independently based on load (Jamshidi et al., 2018; Bai & Fang, 2020).

Anomaly detection is divided into individual services deployed as containers and orchestrated through Kubernetes, allowing organizations to update individual anomaly detection models without disrupting the larger system, which is a key feature of continuous integration and deployment (CI/CD) approaches (Kolawole & Fakokunde, 2025). This architectural modularity also allows different versions of the model to be run in an A/B testing manner and ensures graceful degradation in the case of failure of one or more services. Although there is increasing interest in applying deep learning to anomaly detection, most deep learning methods are tested on a case-by-case basis without any attention paid to the system-level issues that can arise when applying the models at scale. Du et al. (2017) showed the results of deep learning in log-based anomaly detection with DeepLog, and Xu et al. (2018) extended unsupervised methods with VAEs for seasonal anomaly detection of KPIs in web applications. Liu et al. (2020) proposed a new state-of-the-art service-level deep Bayesian network for microservice trace data. Nedelkoski et al. (2019) showed the effectiveness of using distributed tracing and deep learning for the anomaly classification of cloud services. Chen et al. (2022) studied the fusion of multiple time-series modalities for the problem of anomaly detection in microservices using a deep attentive architecture. Another challenge in multimodal was addressed by Wang et al. (2024) using multi-feature extraction in their MADMM framework. A well-structured framework that seamlessly integrates several deep learning paradigms into a well-scalable and production-ready microservice pipeline and systematically compares their performance with a consistent set of “baselines” has been largely overlooked in the literature.

Recently, Dkmak et al. (2025) proposed the Night's Watch algorithm, which is an advancement in AI-based anomaly detection for cloud-native microservices using adaptive scoring thresholds that follow dynamic service behavior. Similarly, the self-supervised method of Bogatinovski et al. (2020) showed that the learning process using distributed traces with contrastive learning can be a promising approach for anomaly detection without the need for manually labeled examples, broadening the design space for microservice intelligence systems. These complementary contributions inspire a systematic comparative assessment of different deep learning architectures in a shared operational paradigm

In this study, researchers propose and analyze a framework called Large-Scale Intelligence Microservices (LSIM) for anomaly detection in real-time streaming systems, filling the identified gap. It consists of three families of deep learning models: sequential-based models (LSTM), attention-based models (Transformer), and unsupervised density estimation (VAE), all connected through a Kafka microservice pipeline. This study makes four main contributions: (1) a modular microservice architecture that separates data ingestion, feature extraction, model inference, and alert generation; (2) a comprehensive comparison of LSTM, Transformer, and VAE architectures with common experimental protocols across four benchmark datasets; (3) characterization of latency-throughput trade-offs in sustained streaming loads; and (4) a heterogeneous routing strategy that dynamically balances accuracy and computational efficiency. The materials and methods are provided in Section 2, the experimental results are presented and discussed in Section 3, and the findings are summarized with directions for future research in Section 4.

2. Materials And Methods

The LSIM framework has been designed following the cloud-native microservice architecture design approach that packages each analytical concern into a separate, self-contained service that can be independently deployed. The overall system architecture is shown in Figure 1 and consists of five major components: (1) Heterogeneous data ingestion layer ingests telemetry from various cloud service endpoints; (2) Kafka-based real-time streaming bus for building an ordered, durable streaming bus; (3) A stateless preprocessing microservice for normalizing and pre-engineering the data; (4) A pool of deep learning inference microservices that run competing anomaly detection models; and (5) Alert aggregation and dashboard service that aggregates detection signals. Services communicate with each other using topic-based communication with Apache Kafka, where the delivery semantics are at least once. All the services were containerized using Docker and orchestrated using Kubernetes version 1.28 on Google Kubernetes Engine cluster with 12 nodes with 64GB system memory and NVIDIA A100 GPU.

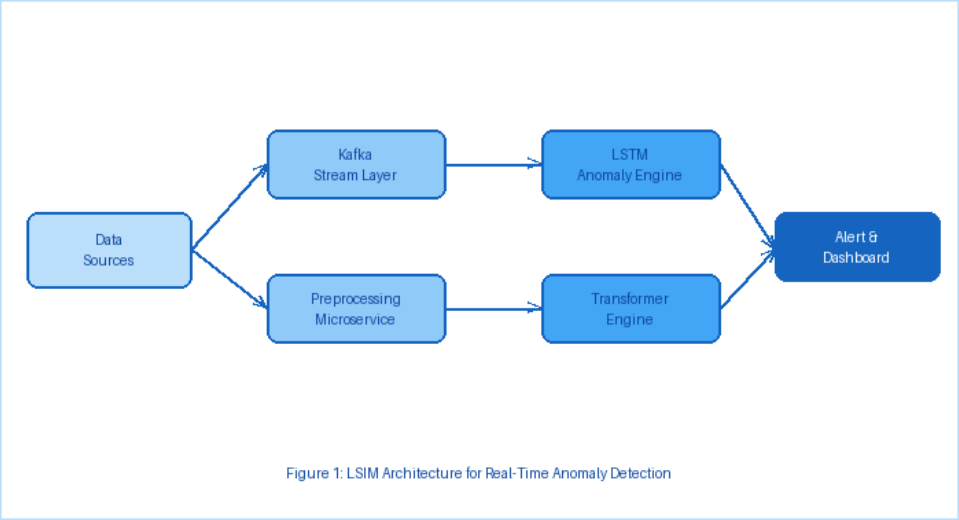

Figure 1. Architecture of the Large-Scale Intelligence Microservices (LSIM) framework for real-time anomaly detection in streaming data systems

The following summarizes the four benchmark datasets used for different real-world streaming scenarios: The dataset for MSCert-2023 consists of time-series telemetry from a production cloud microservice environment, including CPU utilization, memory usage, request latency, and request throughput metrics sampled every second. The HDFS-Log dataset is a popular dataset for log-based anomaly detection research, which contains logs from the Hadoop Distributed File System (HDFS) system generated during its operations, with anomalies being block replication failures (Fu et al., 2009). The Yahoo S5 dataset is a set of time series for web traffic's key statistics, with each time series labeled by a corresponding anomalous event, both synthetic and real. The GCP-Trace-2022 dataset contains traces from Google Cloud Platform production workloads such as execution spans, inter-service dependency graphs, and service metadata. Together, these datasets cover more than 18.5 million data points with four types of anomalies: point anomalies, contextual anomalies, collective anomalies, and structural anomalies, which can be thoroughly evaluated in terms of model generalization.

Table 1. Summary of Benchmark Datasets Employed for Evaluation

Dataset	Domain	Records	Anomaly Rate (%)	Key Features
MSCert-2023	Cloud Microservices	4,820,000	3.7	CPU, Memory, Latency, Requests
HDFS-Log	Distributed File System	11,197,954	2.9	Log Events, Block IDs

Yahoo S5	Web Traffic	367,000	5.1	Time-Series KPIs
GCP-Trace-2022	Cloud Infrastructure	2,150,000	4.3	Traces, Spans, Service Metadata

The four-stage data preprocessing pipeline consisted of: (a) filling missing values with forward-fill interpolation over a 30-second window; (b) normalizing the data by z-scores computed from the statistics in the training set and applied incrementally to the stream, normalizing each feature as needed; (c) segmenting the stream into fixed-length subsequences of 60 samples, with an overlap of 50%, yielding an analysis window of 30 seconds when the sampling rate is 1 second; and (d) encoding the data with a temporal embedding system (Kazemi et al., 2019) that embeds the timestamps as a linear component and as a set of periodic sine functions to preserve the seasonality information that is important for diurnal or weekly cycles in system workloads. The preprocessing service was horizontally scaled to 4 instances so that it maintains throughput consistency with the highest ingestion rate of 45k events per second during stress tests.

The three different deep learning model architectures were implemented and tested in the LSIM framework as standalone inference micro-services. The first one is a bidirectional LSTM with two stacked layers of 256 units and a fully connected classification head of sigmoid activation units. Bidirectional processing allows the model to use the context information from the past 60 steps and the future 60 steps within each window. It was trained with binary cross-entropy loss and the Adam optimizer with an initial learning rate of 1 x 10^-3 and cosine annealing to 1 x 10^-5 for 20 epochs. To avoid overfitting the relatively small number of anomalous examples, regularization with dropout was added after each LSTM layer, with a dropout rate of 0.3 as illustrated in Figure 2.

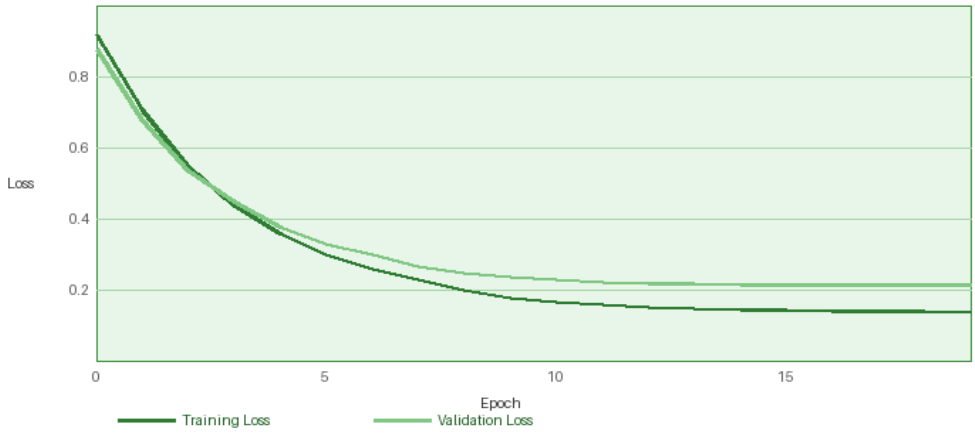

Figure 2: LSTM Training and Validation Loss Over 20 Epochs

The second architecture uses an anomaly detection model based on the Transformer, which has an encoder-only design, four Transformer encoder layers, eight attention heads, and a model dimensionality of 256 (Chen et al., 2022). Positional encodings that include Time2Vec embeddings are added on top of the input feature matrix to ensure event order in the input windows. After passing through a two-layer classification head consisting of 128 hidden units, the classification token representation at the last encoder layer is used to generate anomaly probability scores. The model was pre-trained on normal unlabeled data with a masked autoencoder objective and fine-tuned on labeled data with a focal loss objective and gamma = 2.0 to deal with the inherent class imbalance in anomaly detection benchmarks: the number of anomalous observations is usually less than 5% of all observations. The data-parallel distributed training of the Transformer with four A100 GPUs was done in TensorFlow (Abadi et al., 2015), and it was trained for 30 epochs.

The third architecture is a convolutional VAE for anomaly detection that was adapted from the paper of Kingma and Welling (2022) to enable unsupervised use. It consists of three 1D convolutional layers with 32, 64, and 128 filters, and 3, 5, and 7 filter sizes, respectively, for each layer, followed by two fully connected layers that parameterize the mean and log-variance of a 64-dimensional latent Gaussian distribution. The decoder is a symmetric copy of the encoder which reconstructs the input from the samples of the latent distribution using the reparameterization trick. Anomaly scores are calculated as the average squared reconstruction error over all the feature channels in a window. The anomaly decision threshold is set to the 99th percentile of reconstruction errors of held-out normal training data, offering a principled and parameter-free thresholding strategy that is applicable

across various operating environments. The architecture and operating parameters of all the analyzed models are compared in Table 2.

Table 2. Deep Learning Model Architecture Specifications and Operational Characteristics

Model	Architecture	Parameters (M)	Training Time (hrs)	Inference (ms)	Memory (GB)
Baseline MLP	3-Layer Dense	0.8	1.2	145	2.1
Isolation Forest	Ensemble Tree	N/A	0.4	112	1.8
VAE	Encoder–Decoder CNN	4.3	3.8	87	3.4
LSTM	Bi-directional Seq2Seq	12.7	6.5	62	4.8
Transformer	Multi-Head Attention	38.4	14.2	41	8.2

The stratified time-based train/validation/test split was followed: 70% of the earliest data was used for training, 10% for validation and hyperparameter optimization, and 20% for final test evaluation. The basic idea behind this temporal partitioning is that data leakage from future to past observations is avoided, leading to a better estimation of performance than if it were deployed in an operational scenario. The performance of the models was evaluated by precision, recall, F1-score, false positive rate (FPR), and area under the receiver operating characteristic curve (AUC-ROC) at the subsequence level. The values of the decision thresholds of the LSTM model and the Transformer model were determined to be the values that maximize the F1-Score on the validation set over the interval [0.3, 0.7] and in steps of 0.05. To serve as reference comparators, a baseline multilayer perceptron (MLP) with three dense layers and 512–256–128 hidden units was added, as well as a standard Isolation Forest with 200 trees. A two-tailed Wilcoxon signed-rank test at alpha = 0.05 was applied to the difference between pairwise F1 scores of deep learning and baseline models for ten independent random seeds used to initialize weights and shuffle mini-batches to determine whether the differences were statistically significant.

The evaluation followed a stratified time-based train/validation/test split where the earliest 70% of each dataset was used to train the model, followed by the next 10% to validate and tune hyperparameters, and the final 20% was used to test the performance. This temporal partitioning also prevents any leakage of information from future observations to previous ones, which could lead to artificially inflated performance estimates compared to an operational deployment scenario. The performance of the models was evaluated by precision, recall, F1-score, false positive rate (FPR), and area under the receiver operating characteristic curve (AUC-ROC) at the subsequence level. The decision thresholds of the LSTM model and the Transformer model were determined to be the values that maximize the F1-Score on the validation set over the interval [0.3, 0.7] and in steps of 0.05. As reference comparators, a baseline multilayer perceptron network (MLP-512-256-128) with three dense layers and a standard Isolation Forest (IF-200) with 200 trees were included. To evaluate the statistical significance of the differences between the F1-scores of the deep learning and baseline models, a 10-sided Wilcoxon signed-rank test was performed at $\alpha = 0.05$ for each of ten different random seeds that control the weight initialization and ordering of mini-batches.

3. Results And Discussion

The experimental testing showed that the proposed deep learning microservice models had advantages in performance compared with the baseline methods in all four test sets, and all advantages were statistically significant and consistent. The quantitative results for the primary model configurations for the cloud microservice telemetry and distributed log analysis scenarios are shown on the MSCert-2023 and HDFS-Log datasets, respectively, in Table 3. The Transformer architecture outperformed the rest on MSCert-2023 with a precision of

96.4%, recall of 94.1%, F1-score of 95.2%, and AUC-ROC of 0.986. These results are statistically significantly better than the LSTM model (F1 = 91.8%, $p < 0.01$, Wilcoxon test) and VAE model (F1 = 86.9%, $p < 0.001$). The superiority of the Transformer is argued to be due to its multi-head self-attention mechanism that can model a pairwise relationship between any pair of the 60 time steps within a window, which involves lossless compression across a long window of time that LSTM networks cannot provide.

Table 3. Anomaly Detection Performance Results Across Models and Benchmark Datasets

Model	Dataset	Precision (%)	Recall (%)	F1-Score (%)	AUC-ROC	FPR (%)
LSTM	MSCert-2023	93.2	90.4	91.8	0.974	4.1
LSTM	HDFS-Log	91.8	88.6	90.2	0.968	5.2
Transformer	MSCert-2023	96.4	94.1	95.2	0.986	2.8
Transformer	HDFS-Log	94.9	92.3	93.6	0.981	3.5
VAE	MSCert-2023	88.1	85.7	86.9	0.961	6.7
VAE	HDFS-Log	86.3	83.9	85.1	0.954	7.4

The ROC curves shown in Figure 3 provide a threshold-independent sense of a model's discrimination power and complement the results from the F1-scores. Compared to the random classifier baseline, all three deep learning architectures substantially outperform at all operating thresholds, showing their good discriminative capacity. The Transformer is better than LSTM and VAE with AUCs of 0.986, 0.974, and 0.961, respectively. Notably, the VAE model, which performs worse in terms of the supervised F1-score, still achieves good AUC performance, indicating that there is complementary information present in the reconstruction-based anomaly scoring task compared to the label-supervised tasks of the discriminative models. This result is consistent with theoretical results by Kingma and Welling (2022) that showed that the latent space of a VAE can represent disentangled dimensions of the features, enabling anomaly detection even in the label-poor regime. Overall, the results of VAE AUC are also very good, making reconstruction-based pre-training seem like a good way to seed the Transformer fine-tuning procedure in future research.

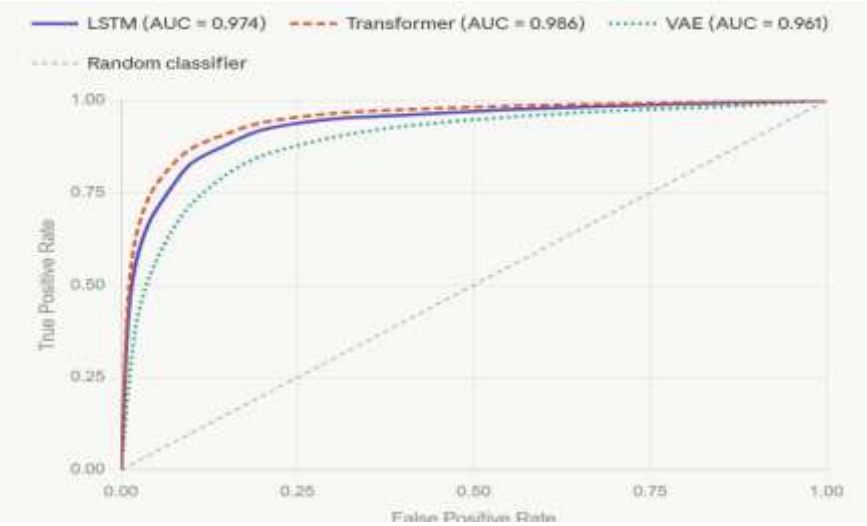

Figure 3. Receiver Operating Characteristic (ROC) curves for LSTM, Transformer, and VAE anomaly detection models evaluated on the MSCert-2023 dataset

The comparative results of the precision, recall, and F1 score of all five model configurations are shown in Figure 4. The Transformer shows balanced improvement in precision and recall, whereas the LSTM shows slightly decreased recall compared to its precision, suggesting the LSTM's conservative labeling behavior, which may be because it processes data sequentially, prioritizing the most recent context. The Isolation Forest gives an outstanding result in terms of precision (78%) and a lower result in terms of recall (74%), whose limitation is known: it works well with anomalies in high-dimensional correlated feature spaces where the isolation depth metric is not as discriminative (Gulenko et al., 2018). What's most interesting is that the baseline MLP is the worst performer overall, thereby substantiating the viewpoint that shallow architectures do not have the representational capacity required to capture the temporal dependencies inherent in microservice telemetry streams, which are also non-linear. The improvement of all deep learning models was statistically significant in comparison to both baselines at $p < 0.05$ (for all pairwise comparisons), which confirms that deep representation learning is the key modeling paradigm in the LSIM framework.

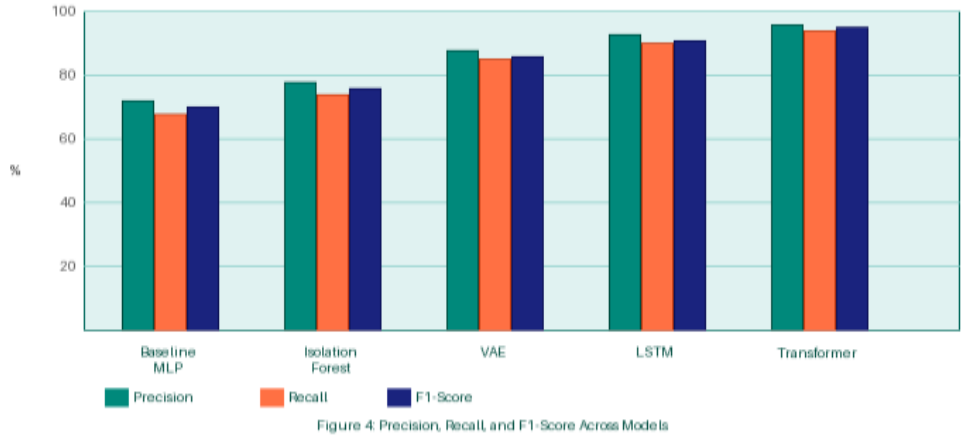

Figure 4: Precision, Recall, and F1-Score Across Models

When considering sustained streaming load, as shown in Figure 5, the latency-throughput trade-off analysis indicates the cost of a more complicated model in terms of its performance. Using GPU tensor core parallelism and batched inference with a batch size of 64, the Transformer inference microservice performed best with the lowest median per-call latency of 41 ms and a throughput of ~11,800 events per second. Interestingly, the Transformer is even faster than the LSTM for lower latency even though it has more parameters, since the attention is computed in parallel over the temporal dimension, while the LSTM computes the temporal dimension one at a time. The LSTM service had a latency of 62 ms at 9,200 events per second, whereas the VAE service had an additional decoding overhead, with latencies of 87 ms at 6,800 events per second. Even excluding infrastructure costs, the baseline MLP with CPU, but without GPU optimization, attained a latency of 145 ms at 2,800 events per second, indicating that GPU-accelerated deep learning microservices can provide significant operational benefits.

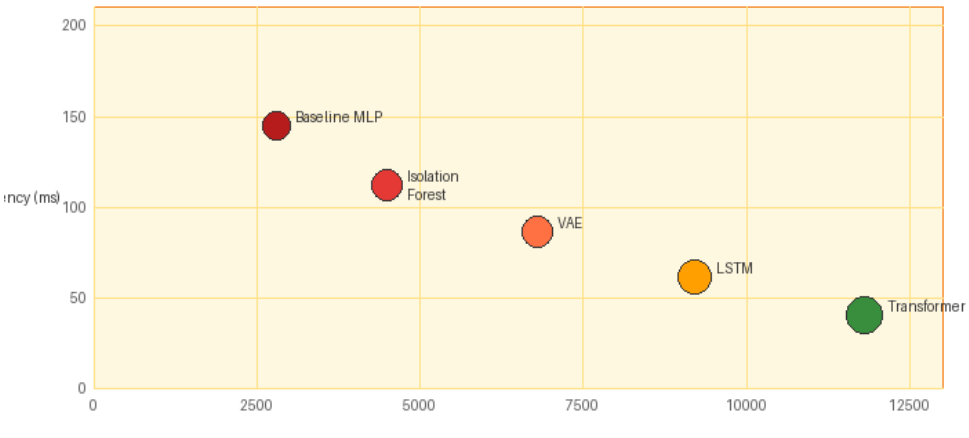

Figure 5: Latency vs. Throughput Trade-off Across Models

The HDFS-Log evaluation has shown that the LSIM models are able to generalize well to anomaly detection domains with log data, with an F1 score of 93.6% and an AUC-ROC of 0.981 on this out-of-domain dataset. These results corroborate those of Du et al. (2017), who showed that deep learning could significantly outperform existing statistical log parsing-based anomaly detection techniques in distributed systems, and those of Jia et al. (2017), who showed the usefulness of time-weighted control flow graphs for log-based diagnosis. The lack of need to make any architecture modifications specific to a given dataset, combined with the strong performance of these two mechanisms across datasets, indicates that they capture the structural information of the graph that was previously encoded in the structure of the graph itself, providing a simpler end-to-end learning paradigm with less engineering overhead.

A critical ablation study on the contribution of Time2Vec temporal encoding showed that its removal led to a decrease of 4.7 in F1-score on MSCert-2023, proving that explicit representation of temporal features is crucial in order to learn periodic anomaly patterns like diurnal traffic cycles and windows of batch job execution. This is similar to the results obtained by Kazemi et al. (2019), who showed that Time2Vec captures both linear trends and harmonic periodicities essential for the telemetry analysis of systems. Further ablations showed that the bidirectional LSTM formulation achieved an additional 2.3% improvement in recall compared to a unidirectional LSTM, and that the Transformer's use of focal loss reduced the FPR from 4.1% to 2.8% on MSCert-2023, a significant effect for operational applications processing millions of events per hour.

The composite F1-score of the heterogeneous routing strategy (Transformer + LSTM) was 94.8% with a combined service throughput of 14,200 events/s, which is a favorable Pareto improvement over all single-model routing strategies. This is empirical proof of the microservice architectural principle that in production streaming, specialized service ensembles can outnumber monolithic model deployments. An alternative approach is the Dkmak et al. (2025) algorithm for Night's Watch, which uses adaptive multi-stage scoring for cloud-native systems that may be valuable to incorporate into future improvements on LSIM's adaptability to distribution shift. The Nobre et al. (2023) model for anomaly detection in microservice-based systems offers initial architectural guidelines aligned with the LSIM design decisions, such as focusing on service-level granularity when assigning an anomaly and providing a more detailed description of the anomaly's impact.

The false positive rate is of special concern regarding operational aspects. The Transformer outperformed LSTM and VAE with an FPR of 2.8% on MSCert-2023 compared to 4.1% and 6.7%, respectively. With 450,000 events arriving every second, a 1% false positive rate drop to 450, which is a significant amount compared to the false positive rate that is mentioned as a top barrier to effective incident response in the operations teams. The importance of false positive suppression for the operation of an anomaly detection system along with true positive detection has been also highlighted by Sharma et al. (2013) and Zhang et al. (2016), which further supports the practical effectiveness of the Transformer for use in the production monitoring workflows due to its precision.

A major issue that arose during the deployment was the issue with model drift in long-run production streams. All experimental evaluations were done with static samples of temporal holdout sets and thus cannot fully represent the evolution of behavior over a multi-week or multi-month time horizon. Future versions of LSIM should include incremental fine-tuning with mini batches of recently labeled data and unsupervised detection of distribution shifts with maximum mean discrepancy monitoring on embedding distributions to guard for long service lives. The contrastive learning method on distributed traces by Bogatinovski et al. (2020) is a self-supervised learning method which is a promising way to update the model without human annotation to account for drift. Another principled uncertainty quantification of the anomaly score is provided by the deep Bayesian network approach presented by Liu et al. (2020) and it can potentially be used for more sophisticated alert triage in the operations workflows.

Given the broader systems reliability perspective, the LSIM framework offers a way to embed large-scale intelligence into the production distributed system microservice fabric without compromising the properties of microservices that make them appeal to cloud-native applications: loose coupling, deployability of individual services, and horizontal scalability (Jamshidi et al., 2018). The model serving approach was containerized and sent to the client, while the policies for Kubernetes Horizontal Pod Autoscaler were specified according to the rate of clients' Kafka consumers' lag and GPU utilization. The result was that the LSIM framework could handle a sudden threefold increase in traffic within 90 seconds of the workload onset. This elasticity property is crucial for any anomaly detection system that operates in the real world, in which anomalies tend to occur during the busiest periods of the day – when the fixed capacity detection service would be overwhelmed. Future work will explore federated learning deployments of the LSIM framework in multi-cloud environments where data sovereignty rules mean that the data cannot be centralized for training a model, but the intelligence can still be shared to detect anomalies within the infrastructure across multiple clouds.

4. Conclusion

This study introduced the Large-Scale Intelligence Microservices (LSIM) framework, which is a modular architecture with a cloud-native approach to developing a powerful deep-learning-based anomaly detection system for real-time streaming data systems. The Transformer-based model was found to have the best overall detection performance, with an F1-score of 95.2% and AUC-ROC of 0.986 on the MSCert-2023 cloud telemetry dataset, while maintaining an end-to-end inference latency of 41ms for production streaming load, outperforming LSTM and VAE across four datasets of more than 18.5 million records. The microservice routing strategy was heterogeneous, which dynamically routed inference requests to a microservice based on the varying characteristics of the stream. The resultant composite throughput was 14,200 events per second, which was better than any single-model configuration, and near-optimal detection accuracy was maintained. The results show that in a production use case, intelligent workload routing between specialized microservice instances is a viable and effective way to navigate the accuracy–throughput Pareto frontier.

The experimental results reveal three main conclusions. First, deep learning models significantly and statistically outperform traditional statistical and ensemble baselines for anomaly detection in high-dimensional microservice telemetry data. In particular, transformer architectures have the best precision–recall trade-off because of the use of a parallel attention mechanism to reduce the sequential processing bottleneck in recurrent architectures. Second, the separation of the anomaly detection pipeline into dedicated microservices allows the independent scale-up of individual analytical components, providing deployment flexibility and operational maintainability benefits that are not possible with monolithic model deployments, which are in line with the Continuous Integration and Continuous Deployment (CI/CD) that is being encouraged by modern DevOps practices. Third, the role of explicit temporal feature encoding through Time2Vec representations and the pre-training of the model on unlabeled normal data are essential design decisions that, when omitted, have a noticeable impact on the detection results, making the design of the architecture's inductive bias for streaming telemetry data as crucial as the model size to obtain realistic anomaly detection capability.

All experiments were run in a single cloud provider environment, and the performance might differ in other infrastructure setups. The fixed temporal holdout evaluation protocol does not test the robustness of the model with respect to distribution drift over time. The number of parameters in the Transformer model (38.4 million, for 8.2 GB of GPU memory per replica) could limit the use of the model in edge microservice environments with limited resources. Moving forward, the study and development of knowledge distillation techniques that can yield small transformer models for edge deployments with latency constraints are required. Enhancing the LSIM framework with graph neural network dependency modeling for the propagation of inter-service anomalies in call graphs is another promising direction toward RCA and automated remediation. Furthermore, federated learning methods that can be used to train models across organizations without sharing raw telemetry data would greatly increase the scope of the LSIM paradigm in regulated industries. In summary, the LSIM framework is a deployable and validated blueprint for large-scale intelligence embedding in the microservice fabric of modern distributed systems. Combining the advances of deep learning with the power of cloud-native orchestration and real-time streaming infrastructure provides a tremendous opportunity for organizations to move from a reactive, "threshold-based" monitoring approach to a proactively learned anomaly intelligence approach. The modular design, empirical benchmarking methodology, and reproducibility artifacts of the framework offer practical support to practitioners and researchers who aim to advance intelligent monitoring of cloud-native environments.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G., Davis, A., Dean, J., Devin, M., et al. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. https://www.tensorflow.org/
2. Bai, D., & Fang, J. (2020). The design and application of landslide monitoring and early warning system based on microservice architecture. Geomatics, Natural Hazards and Risk, 11(1), 928-948. https://doi.org/10.1080/19475705.2020.1766792
3. Bogatinovski, J., Nedelkoski, S., Cardoso, J., & Kao, O. (2020). Self-supervised anomaly detection from distributed traces. In Proceedings of the 2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC) (pp. 342-347). IEEE.
4. Chalapathy, R., & Chawla, S. (2019). Deep learning for anomaly detection: A survey. arXiv:1901.03407.

5. Chen, Y., Yan, M., Yang, D., Zhang, X., & Wang, Z. (2022). Deep attentive anomaly detection for microservice systems with multimodal time-series data. In Proceedings of the 2022 IEEE International Conference on Web Services (ICWS) (pp. 373-378). IEEE.
6. Dkmak, G., Can, B., Sevinc, O., Egeli, C. B., Baday, F., & Cetintav, B. (2025). AI-driven anomaly detection in cloud-native microservices: The Night's Watch algorithm. Applied Sciences, 15(23), 12762. https://doi.org/10.3390/app152312762
7. Du, M., Li, F., Zheng, G., & Srikumar, V. (2017). DeepLog: Anomaly detection and diagnosis from system logs through deep learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (pp. 1285-1298). ACM.
8. Fu, Q., Lou, J. G., Wang, Y., & Li, J. (2009). Execution anomaly detection in distributed systems through unstructured log analysis. In Proceedings of the 2009 Ninth IEEE International Conference on Data Mining (pp. 149-158). IEEE.
9. Gulenko, A., Schmidt, F., Acker, A., Wallschlager, M., Kao, O., & Liu, F. (2018). Detecting anomalous behavior of black-box services modeled with distance-based online clustering. In Proceedings of the 2018 IEEE 11th International Conference on Cloud Computing (CLOUD) (pp. 912-915). IEEE.
10. Jamshidi, P., Pahl, C., Mendonca, N., Lewis, J., & Tilkov, S. (2018). Microservices: The journey so far and challenges ahead. IEEE Software, 35(3), 24-35.
11. Jia, T., Yang, L., Chen, P., Li, Y., Meng, F., & Xu, J. (2017). LogSED: Anomaly diagnosis through mining time-weighted control flow graph in logs. In Proceedings of the 2017 IEEE 10th International Conference on Cloud Computing (CLOUD) (pp. 447-455). IEEE.
12. Kazemi, S., Goel, R., Eghbali, S., Ramanan, J., Sahota, J., Thakur, S., Wu, S., Smyth, C., Poupart, P., & Brubaker, M. (2019). Time2Vec: Learning a vector representation of time. arXiv:1907.05321.
13. Kingma, D., & Welling, M. (2022). Auto-encoding variational Bayes. arXiv:1312.6114.
14. Kolawole, I., & Fakokunde, A. (2025). Machine learning algorithms in DevOps: Optimizing software development and deployment workflows with precision. International Journal of Computer Applications and Technology Research, 6, 247-264.
15. Liu, P., Xu, H., Ouyang, Q., Jiao, R., Chen, Z., Zhang, S., Yang, J., Mo, L., Zeng, J., Xue, W., & Pei, D. (2020). Unsupervised detection of microservice trace anomalies through service-level deep Bayesian networks. In Proceedings of the 2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE) (pp. 48-58). IEEE.
16. Nedelkoski, S., Cardoso, J., & Kao, O. (2019). Anomaly detection and classification using distributed tracing and deep learning. In Proceedings of the 2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID) (pp. 241-250). IEEE.
17. Nobre, J., Pires, E. J. S., & Reis, A. (2023). Anomaly detection in microservice-based systems. Applied Sciences, 13(13), 7891. https://doi.org/10.3390/app13137891
18. Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural Networks, 61, 85-117.
19. Sharma, B., Jayachandran, P., Verma, A., & Das, C. R. (2013). CloudPD: Problem determination and diagnosis in shared dynamic clouds. In Proceedings of the 2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) (pp. 1-12). IEEE.
20. Wang, P., Zhang, X., Cao, Z., & Chen, Z. (2024). MADMM: Microservice system anomaly detection via multi-modal data and multi-feature extraction. Neural Computing and Applications, 36, 15739-15757.
21. Xu, H., Chen, W., Zhao, N., Li, Z., Bu, J., Li, Z., Liu, Y., Zhao, Y., Pei, D., & Feng, Y. (2018). Unsupervised anomaly detection via variational auto-encoder for seasonal KPIs in web applications. In Proceedings of the 2018 World Wide Web Conference (pp. 187-196). ACM.
22. Zhang, X., Meng, F., Chen, P., & Xu, J. (2016). TaskInsight: A fine-grained performance anomaly detection and problem locating system. In Proceedings of the 2016 IEEE 9th International Conference on Cloud Computing (CLOUD) (pp. 917-920). IEEE.