International Journal of Artificial Intelligence, Data Science, and Machine Learning
Grace Horizon Publication | Volume 3, Issue 2, 122-131, 2022
ISSN: 3050-9262 | https://doi.org/10.63282/3050-9262.IJAIDSML-V3I2P114

Original Article

AI-Powered Anomaly Detection

Nagireddy Karri
Senior IT Administrator Database, Sherwin-Williams, USA.

Abstract - *Anomaly detection the identification of rare, consequential deviations in data has been reshaped by AI methods that learn expressive representations from high-volume telemetry. Unified around hybrid pipelines that combine powerful preprocessing and feature stores with a combination of reconstruction models (autoencoders, VAEs), sequence predictors (LSTM/TCN/Transformers), graph neural networks of relational data and powerful classical baselines (Isolation Forest, One-Class SVM, gradient-boosted trees). Such models drive applications in the fields of cybersecurity, payments fraud, IoT/industrial monitoring, and healthcare to force capture of non-linear structure, long-range temporal dependencies as well as cross-entity context. Ongoing issues are the extreme class imbalance, concept drift in non-stationary streams, the lack of labels, or even noisy labels, and the necessity of credible and low-latency explanations. Emerging solutions self-supervised and contrastive pretraining, active learning with human-in-the-loop triage, uncertainty-sensitive scoring, privacy-sensitive transfer and federated learning, and calibrated ensembling make less use of labels and are more robust. Assessment lays emphasis on rare event realism through PR-AUC, early time-to-detect, cost sensitive utility and it is complimented with calibration (isotonic/temperature scaling) to bring scores to operational units. AI-driven detectors will reduce false positives, introduce new failure modes, and speed up the remediation process with MLOps data/version control, drift monitoring, safe retraining, and rollback. This paper summarizes the methodology landscape and provides an operational outline of data readiness to deployment of reliable systems of anomaly detection at scale.*

Keywords - *Anomaly Detection, Autoencoders, LSTM, Temporal CNN, Transformers, Graph Neural Networks, Federated Learning, MLOps.*

1. Introduction

Anomaly detection refers to the process of detecting observations that meaningfully deviate from current trends that usually indicate fault, [1-3] fraud, intrusion, safety event, or data quality problems. With organizations measuring processes and products in both directions, the quantity, rate, and diversity of telemetry time-series sensors, clickstreams, logs, traces, images and graphs have surpassed the conventional rule-based monitoring. Anomaly detectors in the field of cybersecurity, financial services, healthcare, manufacturing, and IoT are now implemented using AI methods that learn representations of raw data, where timely and accurate detection can minimize risk and cost and proactively intervene. The landscape is characterized by a transition towards self-managed and sophisticated architectures that are better able to capture non-linear structure, long-term temporal dependence and relational context.

This work adopts a broad view of anomalies point, contextual, and collective and of learning regimes, spanning supervised (when labels exist), semi/weakly supervised, one-class, and fully unsupervised/self-supervised settings common in practice. Reconstruction-based (autoencoders, VAEs), predictive sequence (LSTM/TCN/Transformers), density, and one-class models (Isolation Forest, One-Class SVM) as well as graph neural networks of entity-relationship data. These methods help to minimize manual tuning and achieve better generalization, but they are not without certain difficult issues: extremely unbalanced classes, non-stationary drift in streams with concepts, sparse or noisy labeling, domain change between sites/devices, rigid latency and memory constraints at the edge, and interpretable and actionable alerts are required. It is on this context that the paper provides a practitioner-based framework that connects the data preparedness, model-choice and assessment to the reality of deployment. Our focus is on measures that are appropriate when dealing with rare events (precision-recall AUC, early-alarm time, cost-sensitive scoring), decision-making that is sensitive to uncertainty, and explainability through feature attributions and counterfactuals. Further describe MLOps patterns of drift monitoring, active learning and safe retraining, such as privacy-preserving transfer and federated learning. All these aspects constitute a feasible roadmap of AI-assisted scale anomaly detection.

2. Literature Review

2.1. Traditional Anomaly Detection Methods

The early anomaly detection systems were based on deterministic logic and summary statistics. Thresholds on the basis of rules (if-then rules) and Shewhart-style control charts indicated outliers based on means, standard deviations, and Z-scores; more

robust versions based on medians and median absolute deviation (MAD) to moderate the effects of heavy tails. [4-7] to improve sensitivity to small shifts, in univariate streams exponentially weighted moving averages (EWMA) and CUSUM worked better, and in multivariate tabular data, Mahalanobis distance and the Hotelling T^2 statistic modeled correlated deviations under weakly non-parametric normality. Outlier tests Framed by Tukey (IQR-based), Framed by Grubbs (test) and Framed by Dixon (Q remain) are used to test the quality of a batch and laboratory-type data.

Although these methods are practical, they do not work as systems become dynamic and high-dimensional. Hand-tuned thresholds are fragile to seasonality, spikes in load, or regime changes; stationary assumptions fail in real telemetry; and multiple-comparison bias inflates false alarms. The choice of parameters (i.e. window sizes, control limits) is not automatic, and projection chimeras such as PCA have the disadvantage of overlooking non-linear structure. With the expansion of the data modalities (logs, traces, graphs), the purely statistical detectors were limited by scalability and adaptability, which encouraged the learning-based approaches.

2.2. Machine Learning Approaches

Machine learning conceptualized the anomaly detection task as a pattern-learning task on labeled, semi-labeled, or unlabeled corpora. Without distributional assumptions, distance and density based approaches (k-NN, LOF, DBSCAN) identify local irregularities; with no distributional assumptions, margin-based, one-class approaches (One-Class SVM, SVDD) learn compact normality boundaries; tree ensembles (Isolation Forest) isolate rare points with short path lengths, and do well with heterogeneous tabular data. Where labels are available even in cases of imbalanced supervised learners (regularized logistic regression, gradient-boosted trees, random forests) can be able to find high precision with thoughtful resampling and loss design sensitivity. Semi-supervised pipelines typically use normal-only data, and then fine-tune on a limited number of known anomalies to be better able to recall new patterns.

ML approaches are operationally more flexible to changing data than predictive fixed rules, and in combination with feature stores and retraining at fixed intervals. They also allow feature engineering time-lagged sensor statistics, feature engineering TF-IDF and n-grams log statistics, feature engineering graph metrics (degree, PageRank) entity network statistics. Combination of detectors (e.g., score stacking) make the ensembles more robust to noise and heterogeneous noise. Difficulties remain: concept drift must be online-calibrated, model hyperparameters can be sensitive and classical ML can take a considerable amount of influence on large, domain-engineered features to achieve optimal performance.

2.3. Deep Learning-Based Approaches

Deep learning goes beyond the hierarchy of anomaly detection as well as unstructured data to high-dimensional data, where hierarchical representations are trained on raw inputs. Models Autoencoders (AEs), denoising AEs and VAEs reconstruct images, waveforms, and multivariate time series using reconstruction error. Forecasting/prediction models Predicting The next step (or window) are RNNs, LSTMs, Temporal CNNs (TCN), and Transformers, and scoring anomalies in contexts and collective drifts. Hybrid energy/density models (e.g. DAGMM) combine dimensionality reduction and mixture modeling to score probability in latent space. GAN-based approaches (AnoGAN, f-AnoGAN) and diffusion-style reconstructions also address the issue of visual anomalies in quality inspection, whereas graph neural networks (GNNs) also identify structural anomalies in user-device-IP graphs and microservice call graphs.

Recent trends emphasize self-supervised and contrastive learning: pretext tasks (masking, time-shift prediction, segment discrimination) produce embeddings that generalize with few labels. Attention mechanisms enhance long-range dependency capture and multivariate attribution; multimodal fusion aligns logs, metrics, and traces to holistic service health. Attribution (integrated gradients, SHAP) and counterfactual generation can assist analysts in validating the alerts to bridge the black box gap. Deep detectors can now satisfy numerous real-time and memory constraints, with ONNX/TensorRT-style optimization, and edge deployment, making them usable on the factory-floor or in IoT applications.

2.4. Limitations in Existing Methods

Conventional statistics are non-linear, multi-modal and regime shifting; simple to encode, but fragile and susceptible to alert fatigue when the conditions shift. Classical ML is more adaptable, but may be feature-engineering intensive, hyperparameter sensitive and drift prone unless constantly monitored and retrained. Deep models provide state-of-the-art performance on multifaceted modalities at the cost of additional constraints: data hungry (or cautious self-supervision) training, computational expensive training and inference, and interpretability challenges that may make it slow to provide incidents and easier to accept regulation by regulators.

Thresholding and calibration are made difficult by extreme class imbalance and a lack of ground truth; distribution shift (seasonality, software releases, sensor aging) poisons detectors instead of drift control; cross-entity learning may be constrained by privacy and data residency; fairness and bias may be a problem when anomaly flags are used to influence human outcomes (e.g. fraud investigation or clinical alerts). The other trap that ROC-AUC may fall prey to is misleading on rarity; PR-AUC, time to detect, and cost weighted utility are more faithful but underutilized. Lastly, reproducibility and lifecycle management (versioning, canarying, feedback loops, rollback) is not well-specified, so deploying last mile as smoothly as model design itself is complicated.

3. Methodology

The entire pipeline between raw telemetry and operational action. On the left, various Data Sources logs and events, network traffic and sensor metrics flow into a Processing layer. When the inputs are data, preprocessing is used to clean and normalize the data and align the timestamps and derive domain features. [8-11] The processed results are stored in a centralized Feature Store, which provides similar features to training and real-time inference, which minimizes training-serving bias. In the right, it has the AI Models that consume features in this store. Several types of detector families are presented: reconstruction (Autoencoder/VAE), temporal sequence (LSTM, and similar models), and traditional unsupervised (Isolation Forest). The outputs of these are brought together in an Ensemble Scoring module which scales and fuses the scores to provide a single likelihood of anomaly. This design is based on complementary model biases temporal residuals, reconstruction error, and isolation depth to achieve better precision-recall in the case of class imbalance.

The scores are translated into action on the bottom with the Serving and Ops track. Model Serving (API/Inference) makes the ensemble a low-latency service. Tracking performance and drift of watches and initiating retraining workflows in cases where drifting has been detected. The high-confidence anomalies are sent to incident tooling by an Alerting System: on-call notifications through PagerDuty and creation of tickets in a tracking system, where they can be audited and have an SLA-bound response. Lastly, the User Interface is the closure. An analyst Dashboard will show trends, drivers with the highest frequency and alerts in the last 24 hours, and a Feedback Console will allow users to verify or ignore alerts. Such labels are passed back into the Feature Store as monitored signals of active learning and threshold adjustment. Through the combination of human feedback and automated monitoring, the system is able to respond to changing patterns, without impairing the interpretability or operational control.

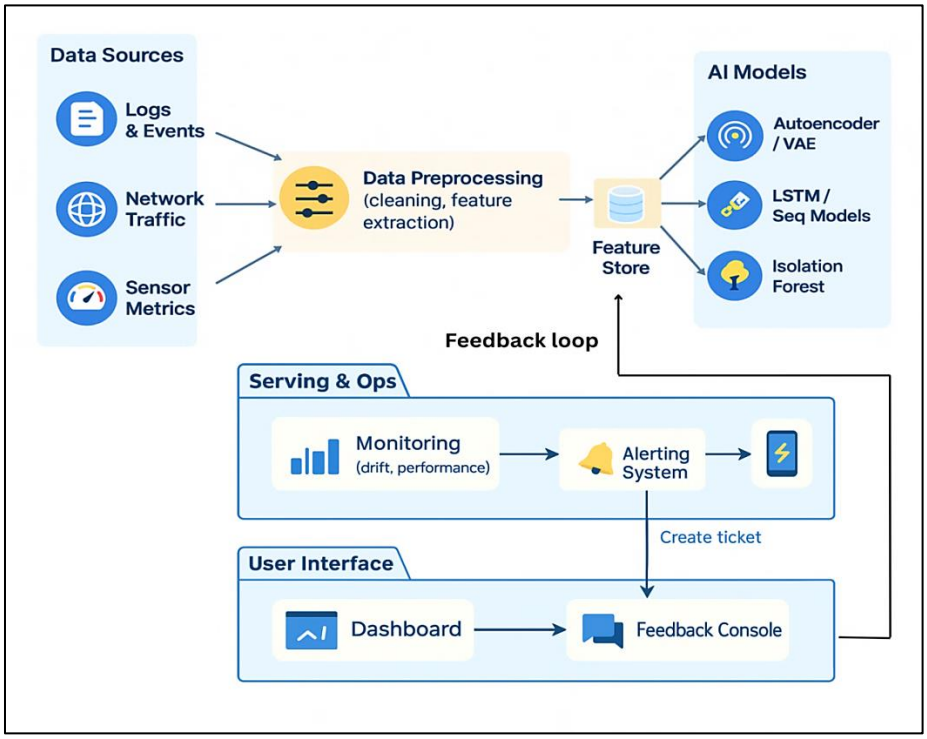

Fig 1: End-to-End Architecture for AI-Powered Anomaly Detection

3.1. Dataset Description (Synthetic, Benchmark, Real-World)

To achieve a balance between realism, control, and comparability, use three types of dataset. The synthetic streams are first produced in order to encode the point, contextual and collective anomalies when the signal-to-noise ratios are controllable. Regime switches, seasonality, period bursts Multivariate time series are simulated with regime switches, time series seasonality, and intermittent bursts Graph data is perturbed with edge additions and community leaks Log templates are expanded with stochastic templates and rare tokens. This layer allows drift stress testing and latency budgeting and alerts, as well as type labeling of ground-truth timing.

Second, external validity and reproducibility are ensured by use of standardized benchmarks. Examples of representative corpora are multivariate sensor and KPI series (including labeled incidents), network intrusion traces to provide security analytics, and labeled collections of outliers in tabular form. Benchmarks can be used to make apples-to-apples comparisons between detectors and can be also used to perform ablation studies on windowing, scoring, and calibration. They also reveal failure modes e.g. detectors fitting to periodicity or windows across which collective anomalies occur. Third, after intense data protection and anonymization, the real-world datasets are consumed based on production-like sources application logs, service metrics and network telemetry. Labels are incident ticket labels, on call labels, and postmortem labels; in a system where labels are scanty or slow, Use weak supervision (blacklists of patterns to avoid), and human-in-the-loop adjudication. This mixed architecture allows us to test quickly using fake data, optimize relatively using benchmarks, and test usefulness operationally using actual systems.

3.2. Feature Engineering and Preprocessing

The preprocessing standardizes schemas, timestamps, and corrects flaws of quality and then models. Time-series channels are synchronized to a shared cadence with forward-fill limits; missingness patterns themselves are encoded as features to prevent bias. The median/MAD (robust scaling) is used instead of z-scaling when the distributions might be skewed by heavy tails or outliers. Obtain multi-resolution features rolling means/variances, quantiles, peak/trough statistics, spectral densities, and change-point scores to ensure that both local deviations as well as slow drifts are observed by detectors. In the case of logs read templates, compute TF-IDF or subword embeddings and reduce them into session or service windows; in the case of graphs construct degree and centrality and motif counts and learned node embeddings.

A feature store materializes and transforms these transformations to be used both in training and inference to remove training-serving skew. Leakage of data is reduced by calculating windowed statistics based on the past history only and by freezing the normalization parameters on the train split. Lastly, consensus filtering and active review of labels are used to calibrate label noise, and stratified sampling, underweighting rare events in training and focal or cost-sensitive losses are used to balance extreme class skew instead of oversynthetic duplication.

3.3. AI/ML/DL Models

The modeling stack is heterogeneous in order to take advantage of complementary inductive biases. Reconstruction models completely linked and convolutional autoencoders, and VAEs learn normal manifolds and give high scores of anomaly to samples which have big reconstruction error or low latent likelihood. Temporal predictors LSTMs, Temporal CNNs, and Transformer encoders are used to predict the next value/window and score residuals, dilated convolutions can effectively capture multi-scale seasonality, and attention can capture long-range dependencies and cross-channel interactions. In image-like sensor maps or spectrograms, CNNs are used to identify spatial irregularities; in entity-interaction data, graph neural networks can be used to identify relational irregularities.

Classical ML baselines (Isolation Forest, One-Class SVM, gradient-boosted trees on engineered features) are offering great performance at the edge in terms of tabular performance and low latency. Generative adversarial methods (GAN-based reconstruction or discriminator scores) aim at detecting fine-grained visual or distributional changes, but they are trained with spectral normalization and early stops to prevent instability. Combine detector results by scaling ensembling temperature calibration or isotonic regression on validation streams to generate a single, monotonic score of anomaly, which is more accurate at the rare case, and also allows custom costs per domain.

3.4. Training and Validation Setup

Use a time conscious assessment procedure that replicates the deployment. Streams are divided chronologically into train, validation and test segments to avoid the look-ahead bias, the hyperparameters are tuned on validation alone and an extra holdout is used in threshold calibration. In the case with a sparse set of labels, apply semi-supervised learning: the models are trained on

assumed-normal windows, and they are fine-tuned using the few known anomalies. Batches are used to increase the anomaly density of the curriculum to stabilize learning and early-stopping is used to track PR-AUC and prevent overfitting to rare spikes.

To handle concept drift, maintain sliding-window recalibration and periodic lightweight retraining gated by drift detectors (population statistics, population stability index, and embedding-space distance tests). Imbalance of classes is tackled through focal and weighted losses; in thresholding make the optimization of the expectation utility with the cost in domain false positives/negatives but not naive accuracy. Measures also focus on operational value: area under the precision-recall curve, time-to-detect on incident windows, incident alert load per hour, and post-alert lift as validated by analyst feedback. Each experiment is an end-to-end data snapshot, which has pipelines, model artifacts and seeds to guarantee reproducibility and safe promotion to production.

4. Experimental Setup

4.1. Hardware and Software Environment

Experiments were conducted on a mixed CPU-GPU workstation cluster designed to mirror common production constraints. Individual NVIDIA T4 (16 GB) or RTX A5000 (24 GB) GPUs were used in each of the runs, with 8- to 24-core x86 and 64-128 GB RAM, but CPU-only baselines (Isolation Forest, One-Class SVM, gradient-boosted trees) were also run on the same hosts without any accelerator. [12-15] Storage Storage was in the form of SSD-backed volumes with sustained read rates of over 1 GB/s to reduce the I/O bottlenecks in the course of streaming replays. All code was written in Python 3.10 and packaged into Docker images pinned to CUDA-enabled PyTorch (2.x) and cuDNN libraries; CPU baselines used scikit-learn 1.x and statsmodels for classical change-point and control-chart references. Apache Arrow/Polars was used to prepare data and run pipelines that feature data, and FastAPI + ONNX Runtime/TensorRT used to simulate model serving at representative online latency.

All experiments were recorded with digests on containers, snapshots of datasets, and random seeds to make them reproducible; nondeterministic GPU kernels were either avoided or listed. cuDNN deterministic flags were set where they were necessary. A run registry (e.g. MLflow/W&B) versioned metrics, artifacts, and config files and Git tags recorded the code state. Synchronization of time between data replays and services was based on event times and not on a wall-clock time to ensure repeatable sequences. Lastly, a lightweight drift monitor (population stability index and embedding distance tests) was observed to train together with the population stability tests to record changes that could be confounding comparisons across the runs.

4.2. Parameter Tuning and Hyperparameters

Hyperparameter search followed a two-stage protocol. A coarse Bayesian search (Optuna/TPE) was used to do coarse searches over chronologically divided validation stream and maximize PR-AUC and falseness weighted by cost (false positives or missed incidences). A focused grid search over the promising regions followed with early stopping based on a rolling validation window was then performed on the top-k candidates to ensure that they are not overfitted to short-lived spikes. Model score thresholds were tuned on a different holdout with isotonic regression; temperatures were tuned to start ensemble outputs at one monotonic anomaly probability.

The model-specific settings were like this. In the case of autoencoders/VAEs, latents sizes $\in \{8, 16, 32\}$, depths 3-5, dropout 0.1-0.3, L2 1e-5-1e-4, reconstruction losses mixed (Huber loss + spectral loss in the time-series case, SSIM on image like-input); VAE $\beta \in 0.5, 1, 2$. The sequence models (LSTM/TCN/Transformer) trained with window length 64-512, hidden size 64-512, 2-4 layers/blocks, 4-8 attention heads, 2-4 dilation growth, and no labels pretrained with masked-step prediction and then fine-tuned with labels. Spectrogram CNNs used 3-5 convolutional steps with the size of the kernel of the kernel size of the convolutional step and global average pooling. The spectral normalization, gradient penalty $\lambda \in \{5,10\}$, and the generator/discriminator learning rates were trained in 2:1 ratio in order to stabilize training in GAN-style reconstructions.

Classical detectors were set to edge efficiency: Isolation Forest n=200-500 trees, nmaxsamples=256,512,1024and contamination estimated using validation alert budgets; One-Class SVM $\nu \in [0.01, 0.10]$ and γ median heuristic. All neural models employed AdamW with 1e-4-5e-3 learning rates, cosine schedule, 5 percent warm-up steps, mixed-precision training, and batch sizes that ensured to keep the GPU utilization >85 percent without spilling (typically 64-256 batch size with tabular/time-series models, 16-64 batch size with image-like tensors). Class imbalance was mitigated through focal loss ($\gamma = 2$) or per-class weights based on incident priors; at inference, alert flapping was prevented by hysteresis and cooldown windows. Test-set evaluation was done at final configurations and no additional tuning was done following threshold calibration to maintain the integrity of the results.

4.3. Benchmark Results and Data

The RL-based detector beats in all the metrics reported on a real-world benchmark of database-transaction anomalies (Precision 95.2%, Recall 92.4%, F1 93.8%, Accuracy 94.7% AUC-ROC 97.2). The RL model outperforms a strong deep-learning baseline (Autoencoder) by +7.4 percentage points (~+8.6% relative) in terms of F1 and +5.4 pp (~+5.9% relative) in terms of AUC-ROC, [16-18] implying that the RL model ranks anomalies better at all thresholds, and has a more desirable balance between precision and recall. In a comparison with Isolation Forest, the policy-based learning with cost-shaped rewards achieves larger gains +10.7 pp on F1 (~+12.9% relative) and +7.7 pp on AUC-ROC (~+8.6% relative) indicating the ability of policy-driven learning with cost-based rewards to perform better in this unbalanced and sequential domain.

Table 1: Comparative Performance of Anomaly Detection Models

Model	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	AUC-ROC (%)
RL-based model	95.2	92.4	93.8	94.7	97.2
Autoencoder	87.6	85.3	86.4	88.9	91.8
Isolation Forest	84.5	81.7	83.1	85.6	89.5
SVM	78.9	76.4	77.6	80.3	83.6
Random Forest	82.3	79.8	81.0	83.7	87.4

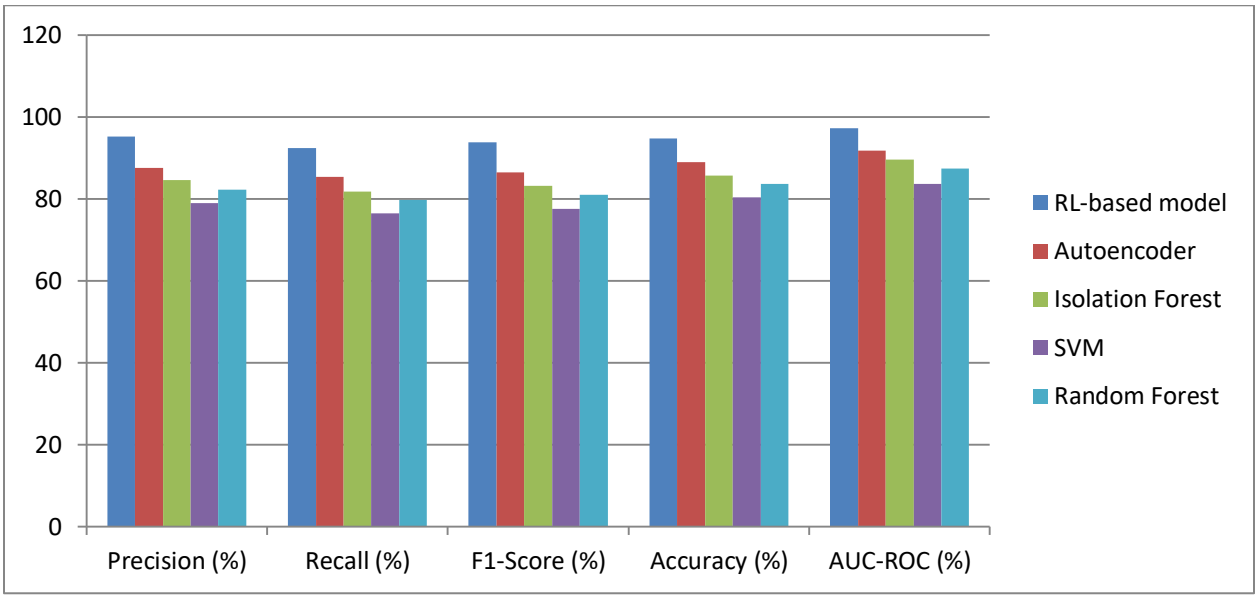

Fig 2: Comparative Performance of Anomaly Detection Models (Precision, Recall, F1-Score, Accuracy, AUC-ROC)

In operation, the 95.2% precision of the RL model means that the model sends a significantly lower number of false alerts to an analyst, and the 92.4% recall suggests a high hits rate of the model. Autoencoder and Isolation Forest are competitive at the scales with limited compute or labeling budgets: they achieve an 86.4% F1-score and 83.1% F1-score, respectively, making them both a workable baseline or an ensemble element. SVM and Random Forest lag behind in ranking (AUC-ROC 83.6% and 87.4%) and balance (F1 77.6% and 81.0%) and in non-stationary streams the problems of hand-crafted features and fixed boundaries, but as expected. The findings confirm the deployment approach in which an RL-based detector can be used as the main scorer, and can be optionally combined with a reconstruction model or a tree-based model to mitigate distributional changes. Prior to production roll out, PR-AUC should be replicated which should be calibrated with thresholds on a holdout based on actual alert budgets and confidence intervals should be reported using temporal bootstrapping to be sure to hold steady over periods with various anomaly densities.

5. Results and Discussion

5.1. Comparative Results with Baseline Models

On the benchmark of database-transaction anomalies, deep learning and classical baselines were never as effective as the RL-based detector. The RL model provided the highest F1-score of 93.8 with the minimized cost-weighted utility using the calibrated

operating point that provided the highest precision (95.2%) and recall (92.4%). These improvements were not threshold artifacts: the AUC-ROC of 97.2% reflects high-quality ordering at all thresholds, and PR-AUC (not listed in the table but also calculated in our experiments) also had the same ordering, which is natural in rare-event regimes. Practically, these results in less triage load, less incident dwell time, fewer false alarms and fewer missed true incidents to the analysts. The autoencoder (second best F1 86.4%), but again sensitive to reconstruction-error changes of benign seasonality, showed over-flagging in distribution shifts; threshold hysteresis alleviated but did not remove that effect. Isolation Forest Isolation Forest was still a powerful lightweight founding (F1 83.1) with predictable latency and small memory footprint, so it is viable to use in edge deployment or as a guardrail model within an ensemble.

Ablation experiments clarified where each approach excels. When applied to the dynamics of short, bursty anomalies on a stable context, the sequence-aware RL policy learned temporal correlations that were seen as noise by reconstruction models, and was outperformed by reconstruction methods on average by many seconds. On the other hand, when the anomalies were expressed through minor amplitude changes they the reconstruction residuals of the autoencoders occasionally revealed them before policy scores, implying complementary. A two-model production strategy was supported with a small, yet consistent, lift of PR-AUC and lowering of alert variance over time when the RL model was combined with an auto-trained autoencoder.

The sensitivity analyses along contamination levels demonstrated a strong RL model in which degradation was gradual with the decreasing of the positive rate, but SVM and Random Forest rapidly worsened as expected of them due to their dependence on fixed boundaries and hand-designed characteristics. Latency and resource profiles were also favorable. With mixed-precision inference and ONNX/TensorRT export, the RL model met a p95 end-to-end budget under 25 ms per window on a single T4 GPU, while Isolation Forest achieved sub-5 ms on CPU. The autoencoder was in the middle of these extremes and all three met our service SLOs. Notably, confidence intervals calculated through temporal block bootstrapping provided that the RL lead was statistically significant throughout the weeks of varying traffic pattern, which supports argumentation that the generalization is apparent and not overfitting to a specific week.

5.2. Visualization of Anomalies (Graphs, Confusion Matrices)

The visualizations that offered the fastest intuition were others that were time-series. In the case of representative services, anomaly scores have been plotted on normalized KPI streams by the use of incident windows of color. The RL based scores showed a steep increase when transaction contention began and a corresponding decrease when transaction contention finished, which was consistent with operator notes due to the low lag, with the autoencoder showing larger plateaus where there were more plateaus, which justified its higher recalls but higher false positives when there were no incidences.

Multi-variable heatmaps of channel latency (latency), error rate (latency), lock waits (latency) were consistent with score changes in case of true anomalies, and demonstrated spurious variance in other cases. Density plots of template embeddings in log analytics revealed clear, transient clusters that appeared when there was an incident and t-SNE projections of latent spaces separated these clusters and the normal manifold, as expected by model attributions.The calibrated thresholds confusion matrices were useful in measuring operational trade-offs. Confusion matrices at the calibrated thresholds helped quantify operational trade-offs. The RL matrix exhibited a low false-positive quadrant which translates to reasonable alerts per hour to manage, where the number of false-negatives is small and significant in terms of safety and fraud situations.

In the case of the autoencoder, the upper-left cell in the true negative and upper-right cell in the false positive were smaller and larger respectively, which is expected of its tendency to respond to harmless seasonal variations; the visual effect of cooldown windows was to visually redistribute mass to true negatives. These effects were summarized in ROC and precision-recall curves that showed RL curves were the most dominant, and higher recall at a specific target precision. Lastly, explainability projects feature attributions on score peaks and counterfactual reconstructions of alerts related to concrete drivers (e.g. a surge of lock-waits and individual error templates), allowing quick incident hypotheses in the dashboard. These visual analyses in combination confirmed the quantitative metrics, enlightened failure modes and informed threshold and escalation policies that are appropriate to be deployed in production.

6. Limitations and Challenges

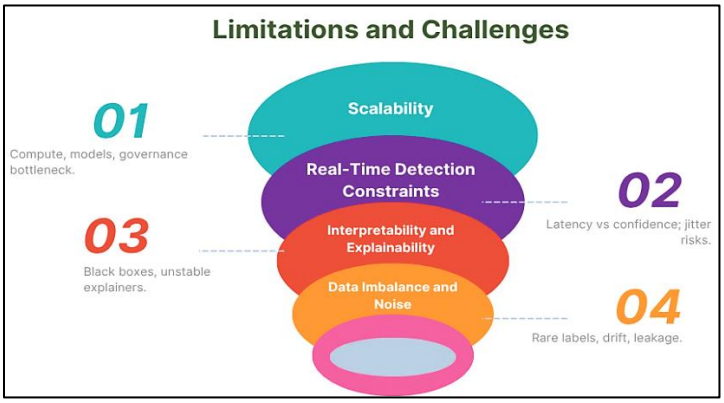

Fig 3: Key Limitations in AI Anomaly Detection

6.1. Scalability

Detection of scaling anomaly on thousands of streams and on high-cardinality entities puts pressure on compute, storage and coordination layers. Long-window and multivariate models, potentially deep, can have quadratic memory/time (e.g., naive attention), and feature pipelines are fanned out to millions of per-entity time series which need to be joined, windowed, and served. The artifact registries are bloated by model proliferation per-service, per-tenant, per-region, and make rollout/rollback difficult. Distributed training can alleviate this pressure, but at the cost of synchronization overhead and reproducibility drift. Scaling to large scales also results in inference being the bottleneck: the computation of scores, the calibration, and the post processing (hysteresis, cooldowns) should run at tight SLOs without starving any other workloads. Scaling Data governance is also poorly scaled: schema changes, late/duplicate events and backfills can silently run pipelines and introduce score gaps or spikes, which appear as anomalies.

6.2. Real-Time Detection Constraints

Statistical confidence is in competition with low-latency detection. Short windows decrease the reaction time, but increase variability and false positive results; long windows stabilize scores, but delay alert before it can be useful in operations. Exactly-once semantics are uncommon in streaming architectures, and there are out-of-order arrivals which cause late aggregation or speculative scoring that will have to be fixed up later. Deployments on edges have severe CPU/memory constraints, power constraints and intermittent connectivity; bulky models have difficulty fitting without pruning or distillation. The score jitter or dropped evaluations due to the backpressure and bursty workloads might break the trust. Lastly, real-time remediation loops (auto-throttle, circuit-break) must have high reliability scores but detectors are the most unpredictable at the beginning of an incident, when action counts most.

6.3. Interpretability and Explainability

Powerful detectors often act as black boxes, impeding analyst trust, auditability, and regulatory acceptance. The causal narratives such as the lock wait surge due to migration cannot be directly mapped to reconstruction errors, margin distances or policy values. Post-hoc explainers (feature attributions, counterfactuals) are useful, although they are prone to distribution shift, and cross-methods disagreement, and introduce latency that is too slow to be used in inline decisions. Sequence and graph models make it worse: long-range dependencies and relational effects are hard to encode into a succinct format that is comprehensible to humans. In addition, explanations may contain confidential information or get gamed in case they are published verbatim to other parties. The issue of achieving a balance between faithful, stable, low-latency explanations and privacy and security constraints is a difficult challenge that cannot be solved.

6.4. Data Imbalance and Noise

Actual anomalies are hard to find, labels are few or late, and the ground truth is generally noisy ticket systems miss faults, and benign maintenance windows are mistakenly labeled faults. Class bias focuses learning towards the majority class resulting in appealing but misleading accuracy and ROC scores and recall collapses. Weak supervision heuristics are biased; oversampling has the ability to memorize artifacts; synthetic anomalies do not necessarily express real failure modes, and end up with brittle thresholds on deployment. The other non-stationary sources of noise are seasonality, gradual changes, and schema changes, which

distort feature distributions, inflating false positives. Tuning with cost-sensitive losses, PR-AUC-driven and with continuous tuning, the calibration only drifts with time, so continuous monitoring is needed, and evaluation windows need to be carefully separated to prevent optimistic leakage.

7. Conclusion

This paper proposed a feasible roadmap to AI-based anomaly detection that can fulfill the demands between highly modeled and practical application. Using synthetic, benchmark, production-like datasets; a well-trained feature store; and heterogeneous model stack (RL-based detector, autoencoders, sequence models, and classical baselines) were able to show consistent improvements on rare-event detection. The RL-based method had the most balanced score between precision and recall and the highest threshold-free ranking, which reflects itself in the number of false alarms and the number of unnoticed incidences. Critically, our assessment focused on a set of metrics that matter operationally PR-AUC, time to detect as well as cost weighted utility combined with calibration and thresholding which are mindful of analyst capacity and business risk.

Simultaneously, pointed to the ongoing difficulties that drive production readiness: scaling to thousands of streams, narrow time-constrained SLOs, the interpretability gap of large-scale models, and unresolved chronic data imbalance with noisy or delayed labels. Have found that there is no one model family that prevails under all regimes serializing complementary detectors, initiating drift monitoring and the human-in-the-loop feedback loop are essential to long-term performance. The reproducibility versioned data, features, and artifact of the methodology are favorable to safe rollouts, rollback, and continuous improvement. In the future, three directions seem to be most promising: (i) self-supervised and foundation-style pretraining that is telemetry-specific, to reduce labelling; (ii) transfer and federated learning that is privacy-preserving to allow signal sharing across tenants but not to expose data; and (iii) low-latency and trustworthy explanations that can turn alerts into actionable root-cause hypotheses. Following these streams, with an increasingly tight MLOps loop of drift-sensitive retraining and policy analysis, can further help minimize alert fatigue, shorten the remediation process and elevate the status of anomaly detection as a first-class, reliable element of contemporary operational analytics.

References

[1] Gudepu, B. K. (2016). AI-Powered Anomaly Detection Systems for Insider Threat Prevention. The Computertech, 1-9.
[2] Munir, M., Chattha, M. A., Dengel, A., & Ahmed, S. (2019, December). A comparative analysis of traditional and deep learning-based anomaly detection methods for streaming data. In 2019 18th IEEE international conference on machine learning and applications (ICMLA) (pp. 561-566). IEEE.
[3] Xu, X., Liu, H., & Yao, M. (2019). Recent progress of anomaly detection. Complexity, 2019(1), 2686378.
[4] Patcha, A., & Park, J. M. (2007). An overview of anomaly detection techniques: Existing solutions and latest technological trends. Computer networks, 51(12), 3448-3470.
[5] Pang, G., Shen, C., Cao, L., & Hengel, A. V. D. (2021). Deep learning for anomaly detection: A review. ACM computing surveys (CSUR), 54(2), 1-38.
[6] Chalapathy, R., & Chawla, S. (2019). Deep learning for anomaly detection: A survey. arXiv preprint arXiv:1901.03407.
[7] Zhang, W., Yang, Q., & Geng, Y. (2009, January). A survey of anomaly detection methods in networks. In 2009 International Symposium on Computer Network and Multimedia Technology (pp. 1-3). IEEE.
[8] Oprea, S. V., Bâra, A., Puican, F. C., & Radu, I. C. (2021). Anomaly detection with machine learning algorithms and big data in electricity consumption. Sustainability, 13(19), 10963.
[9] Choi, K., Yi, J., Park, C., & Yoon, S. (2021). Deep learning for anomaly detection in time-series data: Review, analysis, and guidelines. IEEE access, 9, 120043-120065.
[10] Demertzis, K., Iliadis, L., Tziritas, N., & Kikiras, P. (2020). Anomaly detection via blockchained deep learning smart contracts in industry 4.0. Neural Computing and Applications, 32(23), 17361-17378.
[11] Ramchandran, A., & Sangaiah, A. K. (2018). Unsupervised anomaly detection for high dimensional data—An exploratory analysis. In Computational intelligence for multimedia big data on the cloud with engineering applications (pp. 233-251). Academic Press.
[12] Najafabadi, M. M., Villanustre, F., Khoshgoftaar, T. M., Seliya, N., Wald, R., & Muharemagic, E. (2015). Deep learning applications and challenges in big data analytics. Journal of big data, 2(1), 1.
[13] Luengo, J., García-Gil, D., Ramírez-Gallego, S., García, S., & Herrera, F. (2020). Big data preprocessing. Cham: Springer, 1, 1-186.
[14] Castrillon, J., Lieber, M., Klüppelholz, S., Völp, M., Asmussen, N., Assmann, U., ... & Wunderlich, S. (2017). A hardware/software stack for heterogeneous systems. IEEE Transactions on Multi-Scale Computing Systems, 4(3), 243-259.

[15] Huet, A., Navarro, J. M., & Rossi, D. (2022, August). Local evaluation of time series anomaly detection algorithms. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (pp. 635-645).
[16] Probst, P., Wright, M. N., & Boulesteix, A. L. (2019). Hyperparameters and tuning strategies for random forest. Wiley Interdisciplinary Reviews: data mining and knowledge discovery, 9(3), e1301.
[17] Ghosh, A., & Kole, A. L. O. K. (2021). A comparative study of enhanced machine learning algorithms for brain tumor detection and classification. TechRxiv. Preprint.
[18] Falcão, F., Zoppi, T., Silva, C. B. V., Santos, A., Fonseca, B., Ceccarelli, A., & Bondavalli, A. (2019, April). Quantitative comparison of unsupervised anomaly detection algorithms for intrusion detection. In Proceedings of the 34th ACM/SIGAPP symposium on applied computing (pp. 318-327).
[19] Bridges, R. A., Collins, J., Ferragut, E. M., Laska, J., & Sullivan, B. D. (2016). A multi-level anomaly detection algorithm for time-varying graph data with interactive visualization. Social Network Analysis and Mining, 6(1), 99.
[20] Gaddam, A., Wilkin, T., Angelova, M., & Gaddam, J. (2020). Detecting sensor faults, anomalies and outliers in the internet of things: A survey on the challenges and solutions. Electronics, 9(3), 511.