AI-POWERED INSIGHTS INTO EC2 CPU UTILIZATION

Detecting Irregularities in Cloud Computing Performance

EC2 CPU Utilization Anomaly Detection

Author: Sumukha C R

Project Type: Anomaly Detection using Machine Learning and Deep Learning

---

1. Problem Understanding and Approach

Cloud infrastructure systems, such as Amazon EC2, continuously generate time-series metrics like CPU utilization, memory usage, and network throughput. Sudden spikes or drops in CPU usage can signal system failures, overload, or configuration issues. Early detection of these anomalies is crucial for reducing downtime and enhancing system reliability.

The dataset provided, "ec2_cpu_utilization_24ae8d.csv" from the Numenta Anomaly Benchmark (NAB), includes CPU utilization percentages over time. The project's aim is to develop an automated anomaly detection pipeline capable of identifying unusual CPU behavior without relying on labeled data.

Objectives

1. Analyze the data and extract meaningful time-series features.
2. Train unsupervised and deep learning models for anomaly detection.
3. Compare model behavior and derive key insights.
4. Summarize findings and propose future improvements.

---

2. Feature Engineering Rationale

Raw CPU utilization values alone are insufficient to capture complex patterns such as seasonality or gradual drifts. To enable models to learn the underlying structure of the data, the following engineered features were created:

Feature Type	Description	Rationale
Rolling Mean & Std	Moving window average and deviation	Captures short-term trends and volatility
Z-Score	Normalized deviation from rolling mean	Identifies spikes beyond normal variation
STL Decomposition	Trend, seasonal, and residual components	Separates long-term drift, periodic cycles, and irregular noise
Lag Features	Previous time steps (1, 2, 3, 6, 12)	Allows sequence models to observe past context
Standard Scaling	Normalization of all numerical inputs	Ensures equal weighting across features

These features transform the time series into a multi-dimensional input that encodes both short-term dynamics and long-term patterns, improving model robustness.

---

3. Model Selection and Comparison

Two complementary models were used to detect anomalies:

(a) Isolation Forest (Unsupervised Tree-based)

- **Concept:** Randomly partitions data and isolates rare points that require fewer splits.
- **Advantages:** Fast, interpretable, and effective for high-dimensional data.
- **Configuration:** 300 estimators, contamination = 0.01.
- **Output:** Anomaly score ("iso_score") for each observation.

(b) LSTM Autoencoder (Sequence Reconstruction)

- **Concept:** Learns to reconstruct “normal” sequences. High reconstruction error indicates an anomaly.
- **Architecture:** 2 LSTM encoder layers → bottleneck → 2 LSTM decoder layers → TimeDistributed Dense output.
- **Sequence Length:** 24 time steps.
- **Output:** Mean squared reconstruction error ("ae_score") per sequence.

Comparison Summary

Aspect	Isolation Forest	LSTM Autoencoder
Learning Type	Unsupervised (static)	Deep (temporal)
Speed	Fast	Slower
Captures Seasonality	Partial	Strong
Detects Gradual Drift	Limited	Yes
Interpretability	High	Moderate
Best For	Sudden spikes	Long-term deviations

The combination of both models provides a balanced perspective — Isolation Forest detects local outliers, while LSTM Autoencoder captures evolving trends.

---

4. Key Findings and Business Insights

1. **Periodic CPU Usage Pattern:** The system exhibits daily cycles, consistent with scheduled workloads or batch processes.
2. **Detected Anomalies:**
   - Isolation Forest flagged sudden peaks (approximately 2–3× higher than baseline).
   - LSTM Autoencoder identified gradual drifts during workload transitions.
3. **System Behavior Insight:** Anomalies align with transitions between low and high utilization phases, potentially indicating auto-scaling or deployment events.
4. **Operational Value:**
   - Early anomaly alerts enable proactive resource scaling.
   - Reduces Mean Time to Recovery (MTTR) through targeted diagnostics.
   - Supports SLA compliance and infrastructure reliability monitoring.

Overall, the models demonstrate that data-driven anomaly detection can replace manual threshold tuning and scale across multiple EC2 instances.

---

5. Limitations and Future Improvements

Limitation	Suggested Improvement
Univariate input (CPU only)	Extend to multivariate metrics (memory, network, disk I/O).
Static thresholding	Use adaptive thresholds or percentile tuning via validation.
Lack of labeled anomalies	Incorporate human feedback or semi-supervised learning.
Batch inference only	Convert pipeline to real-time streaming (AWS Lambda + S3 + CloudWatch).
No deployment interface	Add REST API / Flask endpoint for production scoring.

---

6. Conclusion

This project demonstrates an end-to-end Anomaly Detection Pipeline for cloud metric monitoring using both classical and deep learning methods. The modular structure ("src/", "notebooks/", "reports/") :

- Data preprocessing and feature design
- Model implementation and comparison
- Visualization and insight communication
- Awareness of deployment and scalability considerations

By extending this work to real-time and multi-metric scenarios, it can serve as a robust foundation for intelligent cloud monitoring systems.