# Real-Time Multi-Agent Framework for Detecting Cascading Failures in Machine Learning Pipelines

Ms. M.Jayasri[1], P.Govardhan[2], SK.Bhanu Tej[3], Sk.Hasan Basha[4]
[1]*Assistant Professor, Department of Artificial Intelligence and Data Science, Dhanalakshmi Srinivasa University, Tamil Nadu, India*
[2,3,4]*Department of Artificial Intelligence and Data Science, Dhanalakshmi Srinivasan University, Tamil Nadu, India*

***Abstract*— Modern machine learning systems are deployed as multi-stage pipelines where the output of one stage becomes the input of the next. In such interconnected systems, small failures such as poor data quality, feature corruption, model performance degradation, or service latency can propagate across stages and lead to cascading failures that affect the reliability of the entire system. Traditional monitoring techniques typically evaluate each pipeline component independently and operate in offline environments, which limits their ability to detect real-time failure propagation. This paper proposes a Real-Time Multi-Agent Framework designed to detect cascading failures in machine learning pipelines. The framework assigns intelligent monitoring agents to different pipeline stages, including data processing, model evaluation, and service execution. These agents continuously monitor system metrics and communicate with a centralized risk analysis module that evaluates cascading failure propagation across the pipeline. When the system detects abnormal propagation patterns, an alert module generates early warnings to prevent large-scale system degradation.**

**The proposed architecture improves system observability, enhances reliability, and provides proactive monitoring capabilities for modern machine learning operations. Experimental analysis demonstrates that the framework effectively identifies early-stage anomalies and reduces the risk of system-wide failures.**

## I. INTRODUCTION

Machine learning technologies are increasingly deployed in large-scale production environments across industries such as finance, healthcare, cybersecurity, and e-commerce. These systems typically operate as pipelines consisting of multiple interconnected stages including data ingestion, preprocessing, feature engineering, model inference, and service deployment.

Each stage in the pipeline depends on the correctness and stability of previous stages. As a result, failures originating from one stage can propagate to downstream components. For example, poor data quality can degrade feature representations, which may lead to incorrect model predictions and service failures. Such progressive system degradation is commonly referred to as a cascading failure. Cascading failures are difficult to detect because they often begin as small anomalies that gradually propagate through dependent components.

Traditional monitoring systems evaluate metrics such as model accuracy, response latency, or data drift individually. However, they do not analyze the relationships between different pipeline components. Consequently, these systems fail to detect early-stage cascading patterns.

To address this challenge, this research proposes a Real-Time Multi-Agent Monitoring Framework that models ML pipelines as interconnected systems and detects cascading failure propagation across stages.

## II. LITERATURE SURVEY

Existing research on ML system monitoring primarily focuses on individual component metrics such as model accuracy, feature drift, and infrastructure latency. Several studies have explored concept drift detection techniques that analyze distribution changes in streaming data.

Other research efforts investigate infrastructure monitoring frameworks designed for distributed computing environments. These systems monitor

resource utilization, network latency, and service availability.

While these approaches improve system observability, they treat ML pipeline components as independent units. As a result, they do not capture the complex dependency relationships between pipeline stages.

Research in distributed systems has explored cascading failure modeling in power grids, communication networks, and financial systems. However, limited work has applied cascading failure analysis to machine learning pipelines.

This research aims to bridge this gap by introducing a multi-agent architecture that monitors each pipeline stage and analyzes how failures propagate across the system.

## III. SYSTEM OVERVIEW

The proposed framework models the machine learning pipeline as a connected system consisting of three main components:

1. Data Processing Stage
2. Model Evaluation Stage
3. Service Deployment Stage

Each component is monitored by a specialized agent responsible for collecting performance metrics and identifying anomalies.

The framework operates continuously and updates cascading risk scores in real time.

## IV. PROPOSED METHODOLOGY

The architecture of the proposed system consists of multiple agents that monitor different aspects of the pipeline.

### A. Data Monitoring Agent

The Data Agent monitors the quality of input data entering the pipeline. It evaluates metrics such as missing values, feature distribution changes, and abnormal data patterns.

If the system detects significant data drift, it generates an anomaly signal.

### B. Model Monitoring Agent

The Model Agent analyzes the performance of the machine learning model during inference.

Metrics monitored include:

- Prediction accuracy
- Prediction confidence
- Model error rate

A decline in model performance indicates potential cascading risk originating from upstream data issues.

### C. API Monitoring Agent

The API Agent monitors the health of the deployed service.

Metrics include:

- Response latency
- Request failure rate
- Throughput stability

An increase in latency or failure rate may indicate cascading effects from upstream stages.

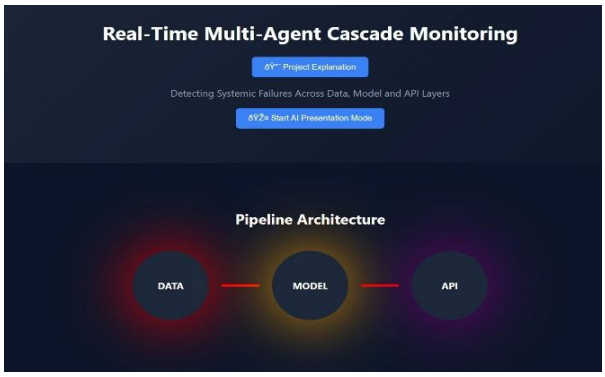
### D. Propagation Risk Agent

The Propagation Risk Agent collects anomaly signals from all monitoring agents and analyzes correlations between them.

The system calculates a cascading risk score using weighted anomaly contributions from each pipeline stage.

### E. Alert Agent

When the cascading risk score exceeds a predefined threshold, the Alert Agent triggers early warnings and notifies system administrators through the monitoring dashboard.




## V. SYSTEM ARCHITECTURE

The proposed system architecture is designed to monitor machine learning pipelines in real time and detect cascading failures by using multiple intelligent agents. The architecture follows a modular design in which each agent is responsible for monitoring a specific component of the pipeline. A centralized risk analysis module integrates signals from all agents to

estimate cascading failure probability and generate early alerts.

The architecture is divided into five major layers: Data Layer, Monitoring Agent Layer, Risk Analysis Layer, Alert Layer, and Visualization Layer.

1. Data Layer

The Data Layer represents the entry point of the machine learning pipeline. This layer is responsible for collecting input data from various sources such as datasets, APIs, or streaming platforms.

Typical components in this layer include:

- Data ingestion modules
- Data preprocessing pipelines
- Feature engineering processes

The quality of data directly affects the performance of downstream stages. Therefore, monitoring the data stage is essential for detecting anomalies such as missing values, abnormal distributions, or corrupted data inputs.

2. Monitoring Agent Layer

The Monitoring Agent Layer consists of multiple independent agents responsible for observing different pipeline components. Each agent continuously collects metrics and generates anomaly signals.

Data Agent

The Data Agent monitors the quality and distribution of incoming data. It checks for issues such as data drift, missing values, and unusual feature patterns. If abnormal behavior is detected, the agent generates a data anomaly signal.

Model Agent

The Model Agent evaluates the performance of the machine learning model during inference. It monitors metrics such as prediction accuracy, confidence scores, and error rates. A significant drop in performance indicates potential cascading failure risk.

API Agent

The API Agent monitors the service deployment stage of the pipeline. It tracks response latency, request failure rate, and service availability. Abnormal behavior in this stage may indicate cascading effects from earlier pipeline stages.

Each monitoring agent works independently but shares its anomaly signals with the central analysis module.

3. Propagation Risk Analysis Layer

The Propagation Risk Analysis Layer acts as the core decision-making component of the architecture. This layer receives anomaly signals from all monitoring agents and evaluates the relationships between them.

The system calculates a cascading failure risk score by analyzing correlations between anomalies detected across different stages of the pipeline.

For example:

- Data anomaly → affects feature quality
- Feature anomaly → affects model performance
- Model anomaly → affects API reliability

By analyzing these relationships, the system can determine whether failures are isolated or propagating across the pipeline.

4. Alert Generation Layer

The Alert Layer is responsible for notifying system administrators when cascading risk exceeds a predefined threshold.

Alerts may include:

- Warning notifications
- Failure alerts
- System status reports

Early alerts allow administrators to take corrective actions before failures propagate further.

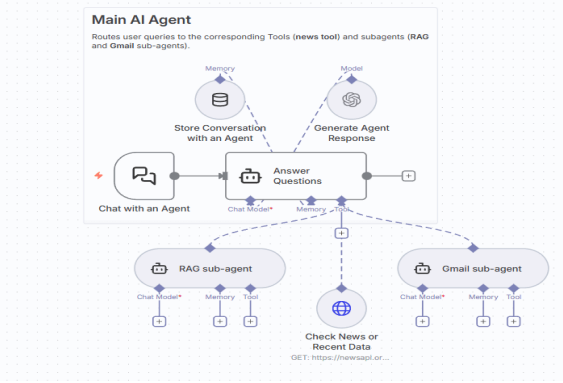
5. Visualization and Monitoring Dashboard

The final layer of the architecture is the visualization layer, which provides a real-time dashboard for monitoring pipeline health.

The dashboard displays:

- Agent anomaly scores
- Cascading risk score
- Pipeline status
- Failure alerts

This interface helps operators quickly understand system health and identify root causes of failures.



## VI. MATHEMATICAL MODELING OF CASCADING RISK

To represent cascading propagation mathematically, the pipeline is modeled as a sequence of stages:

$P = \{S_1, S_2, S_3\}$

Where

$S_1$ = Data Stage

$S_2$ = Model Stage

$S_3$ = API Stage

Each stage produces an anomaly score:

$A_1(t), A_2(t), A_3(t)$

The cascading risk score is computed as:

$C(t) = w_1A_1(t) + w_2A_2(t) + w_3A_3(t)$

Where:

$w_1, w_2, w_3$ represent importance weights.

If:

$C(t) >$ Threshold

then cascading failure risk is detected.


## VII. ALGORITHM FOR CASCADING FAILURE DETECTION

Algorithm: Multi-Agent Cascade Monitoring

Step 1: Collect metrics from each pipeline stage

Step 2: Compute anomaly score for each agent

Step 3: Aggregate anomaly scores

Step 4: Calculate cascading risk score

Step 5: Compare risk score with threshold

Step 6: Trigger alert if cascading risk detected

## VIII. EXPERIMENTAL SETUP

The system was implemented using Python and evaluated in a simulated machine learning pipeline environment.

System configuration included:

- Python programming language
- Scikit-Learn machine learning library
- Pandas data processing
- Flask-based monitoring dashboard
- Matplotlib visualization tools

The system was tested using synthetic anomaly scenarios to simulate cascading failures.

## IX. EXPERIMENTAL RESULTS AND ANALYSIS

The performance of the system was evaluated using multiple test cases.

Scenario 1: Data Drift

Data drift was introduced by modifying feature distributions.

The system detected anomalies early and generated alerts before model performance degradation occurred.

Scenario 2: Model Performance Drop

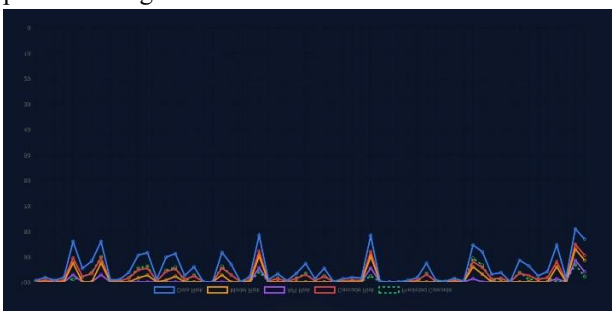
Artificial noise was introduced into model predictions.

The monitoring system detected abnormal accuracy reductions and increased cascading risk scores.

Scenario 3: API Latency Spike

Service latency was artificially increased to simulate infrastructure overload.

The system successfully detected abnormal latency patterns and generated alerts.



## X. COMPARATIVE ANALYSIS

The proposed system was compared with traditional monitoring methods that evaluate independent metrics. Results show that the multi-agent approach improves early detection capability by identifying correlations between pipeline components. This allows system administrators to take preventive actions before cascading failures impact system performance.

## XI.FUTURE SCOPE

Future research may explore the following directions:

- Integration with automated remediation systems

- Use of graph-based dependency modeling
- Deployment in large-scale cloud environments
- Application of reinforcement learning for adaptive risk scoring

These improvements will further enhance system intelligence and scalability.

## XII. CONCLUSION

This research proposed a Real-Time Multi-Agent Framework for detecting cascading failures in machine learning pipelines.

The system introduces distributed monitoring agents and a centralized propagation risk model that analyzes failure propagation across pipeline stages.

Experimental evaluation demonstrates that the system improves reliability and enables proactive system monitoring.

The framework provides an effective solution for improving the resilience of ML production environments.

## REFERENCES

[1] T. M. T. Pham, K. Premkumar, M. Naili, and J. Yang, "Time to Retrain? Detecting Concept Drifts in Machine Learning Systems," *arXiv preprint*, 2024.

[2] F. Majidi et al., "An Efficient Model Maintenance Approach for MLOps Pipelines," *arXiv preprint*, 2024.

[3] M. Yusuff, "Model Drift Monitoring: Continuously Tracking Model Performance Metrics," *ResearchGate Publication*, 2024.

[4] M. Zarour et al., "MLOps Best Practices, Challenges and Maturity Models," *Journal of Systems and Software*, 2025.

[5] A. Reda et al., "A Hybrid MLOps Framework for Automated Lifecycle Management," *Nature Scientific Reports*, 2025.

[6] Z. Qin et al.,"MAS-LSTM: A Multi-Agent LSTM-Based Framework for Scalable Anomaly Detection,"*Processes Journal*, 2025.

[7] R. Hdidou et al.,"A Distributed Multi-Agent Architecture for Secure IoT Systems,"*Journal of Theoretical and Applied Information Technology*, 2025.

[8] R. Solomon et al.,"LumiMAS: A Real-Time Monitoring Framework for Multi-Agent Systems,"*arXiv preprint*, 2025.

[9] X. He et al.,"SentinelAgent: Graph-Based Anomaly Detection in Multi-Agent Systems,"*arXiv preprint*, 2025.

[10] M. A. Belay et al.,"Agentic and LLM-Based Multimodal Anomaly Detection Architectures," *Preprints.org*, 2026.