**Topic 7: Real time multimodal anomaly detection for critical infrastructures (RETIMU-AD)**

# **Project Overview**

The Real-Time Multimodal Anomaly Detection for Critical Infrastructures (RETIMU-AD) project focuses on enhancing the security and resilience of critical infrastructure by detecting unusual or suspicious events in real time. By analyzing diverse data sources, including images, logs, and audio in an integrated dataset, RETIMU-AD offers a comprehensive view of potential threats. This multimodal approach addresses the limitations of single-modality systems, which often miss complex patterns that could indicate risks.

Current anomaly detection systems typically rely on single data types, such as network logs or sensor feeds, which limit their effectiveness in detecting complex threats. Recent advancements have explored combining modalities, using technologies like CNNs for image analysis and RNNs for sequential data, enhancing detection accuracy. However, most existing solutions lack real-time, multimodal integration, which is crucial for high-stakes environments like critical infrastructure.

# **Project Objective**

The primary objective of RETIMU-AD is to fill this gap by developing a real-time, multimodal anomaly detection system tailored to the unique needs of critical infrastructure. The project will utilize an integrated technology stack: Kafka for data streaming, Apache Spark for processing, Python-based machine learning models for anomaly detection, InfluxDB for time-series data storage, and Grafana for real-time visualization. This system will empower stakeholders to monitor data flows, identify anomalies, and respond swiftly, thereby enhancing security and operational resilience.

# **System Architecture**

A screenshot of a computer

Description automatically generated

**System** **Architecture**

## **Data Ingestion Layer:**

The Data Ingestion Layer is where raw data from the Industrial Control System (ICS)—such as sensor readings, system logs, and audio streams—is initially gathered and structured. Data flows into this layer through a Kafka Producer, which organizes it into specific Kafka Topics, with Zookeeper managing the Kafka cluster. This setup provides a structured, reliable pipeline to organize data by type, ensuring that each data stream is kept separate and manageable. By using Kafka Topics, we streamline data handling and create a consistent flow, preparing it for real-time processing in the next layer.

## **Pretrained Machine Learning Model:**

After ingestion, the dataset is also directed to a pretrained machine learning model, which performs early anomaly detection. This model acts as a preliminary filter, identifying any significant deviations or unusual patterns in the data. By analyzing data at this early stage, the model helps flag potential issues quickly, allowing for a proactive response. The output of the model—whether it’s flagged anomalies or specific patterns—feeds into the Stream Processing Layer, where it is combined with the data from Kafka Topics for further analysis and validation.

## **Stream Processing Layer**

In the Stream Processing Layer, data from Kafka Topics and the output from the pretrained model are processed in real time using Spark Streaming. This layer includes pipeline streaming setups that allow for complex data transformations, anomaly verification, and real-time alerting. Spark Streaming serves as a rapid-response framework, performing validation checks on the model’s predictions and applying additional analytical pipelines to identify trends, detect changes, or score the severity of detected anomalies. This ensures that any insights generated are robust and ready for further processing or immediate action.

### **Pipeline Components:**

* **Simple Moving Average (SMA) / Exponentially Weighted Moving Average (EMA)**: Computes moving averages to establish a baseline trend, helping identify abrupt changes in the data stream.
* **Error Rate-Based Change Detection**: Monitors predict error rates from the model, flagging potential anomalies or shifts when error rates rise, indicating possible issues in data or model accuracy.
* **Real-Time Alerts**: Triggers immediate alerts based on anomalies detected across various pipelines, enabling prompt response to critical issues.
* **Data Logging to InfluxDB**: Logs detected anomalies and metadata to InfluxDB, creating a historical record for later analysis and trend monitoring.
* **Count-Based Anomaly Detection**: Tracks event frequencies within specified time windows, flagging unusual spikes or dips, such as frequent error logs, which could indicate emerging problems.
* **Time-Based Threshold Monitoring**: Observes specific metrics (e.g., response time) and flags any data points that exceed set thresholds within a time window, quickly identifying deviations.
* **Rolling Window Standard Deviation**: Calculates variability over a rolling window to detect increases in data fluctuation, which may indicate instability or noise.
* **Simple Rule-Based Filters**: Applies predefined conditions to flag straightforward anomalies, like exceeding temperature or pressure limits, useful for detecting well-known issues.

## **Data Storage & Batch Processing Layer**

The **Batch Processing Layer** leverages **Spark Batch Processing** to perform in-depth analysis of historical data. This periodic analysis refines the detection model, identifies recurring patterns, and enables the system to adapt to long-term trends. By processing accumulated data over time, Spark Batch provides a broad perspective on system behavior, supporting model improvement and proactive adjustments.

**Batch Processing Components**:

* **Data Aggregation and Summary Statistics**: Aggregates historical data to calculate mean, median, and standard deviation for a snapshot of data distribution and long-term shifts.
* **Histogram Analysis**: Creates histograms to visualize data distribution, identifying clusters or unusual concentrations.
* **Rolling Window Average Analysis**: Calculates rolling averages to track long-term trends in data values over time.
* **Basic Clustering (e.g., K-Means)**: Groups data into clusters to distinguish typical vs. atypical patterns, flagging outliers.

These batch processing functions ensure that insights are comprehensive, with trends and recurring issues easily identifiable. This historical analysis supports model refinement, helping the system to become more accurate and responsive over time.

### **Data Storage for Consumer 1 (Logstash & Elasticsearch)**

For **Consumer 1**, processed data flows to **Logstash** and **Elasticsearch**. **Logstash** ingests and formats data for Elasticsearch, while **Elasticsearch** indexes this data, making it easily searchable and suitable for log and event data management. By using Elasticsearch, the system can store, search, and analyze logs efficiently, supporting both real-time queries and long-term log storage. This setup is optimized for handling log data that can be visualized in the next layer through **Kibana**.

### **Data Storage for Consumer 2 (InfluxDB)**

For **Consumer 2**, time-series data is stored in **InfluxDB**, which is designed to handle time-series data effectively. InfluxDB captures real-time logs and statistics over time, allowing for quick retrieval and historical tracking. This storage approach supports trend analysis, long-term monitoring, and enables visualization of time-based metrics in the next layer through **Grafana**.

**Visualization Layer**

* **Consumer 1 (Kibana)**: Uses Elasticsearch-indexed data to visualize logs and events in real time, providing dashboards for immediate anomaly detection and system monitoring.
* **Consumer 2 (Grafana)**: Displays time-series data from InfluxDB, showing historical trends and recurring patterns, supporting long-term analysis and proactive maintenance.