GROUP 14

A GENERAL ANOMALY DETECTION FRAMEWORK FOR FLEET-BASED CONDITION MONITORING OF MACHINES

Kilian Hendrickx, Wannes Meert, Yves Mollet, Johan Gyselinck, Bram Cornelis, Konstantinos Gryllias, Jesse Davis

PROJECT FOCUS:

Early fault detection in machine fleets before failure occurs

KEY INNOVATION:

- Fleet-based approach leveraging similar machines
- Unsupervised anomaly detection without historical labeled data
- Real-time online monitoring with visualization

PROBLEM UNDERSTANDING

Critical Importance: In industrial environments, machines are expected to work with maximum reliability — there’s no room for failure. When breakdowns occur, the consequences go far beyond technical faults.

- Production losses and unexpected downtime
- High repair and maintenance costs
- Serious injuries or loss of life
- Environmental damage

REAL-TIME EXAMPLE

On June 30, 2025, a major explosion occurred at a chemical factory owned by Sigachi Industries Limited in Telangana, India. The blast triggered a fire, killing 46 people and injuring 33 others.

According to the Telangana Fire Department, the plant lacked adequate safety measures, including fire alarms and heat sensors.

It’s a harsh reminder. When machines fail and safety is ignored, the consequences can be deadly.

DRAWBACKS OF EXISTING SOLUTIONS:

HANDCRAFTING INDICATORS

- **Time consuming** - Setting manual vibration thresholds for each motor type requires weeks of expert tuning.
- **Not scalable** - Each new machine or setup needs its own set of custom rules, in addition to its working conditions (location, humidity).

SUPERVISED LEARNING

- **Requires large historical labeled datasets** - A model needs thousands of examples labeled *faulty* vs *normal* to train properly.
- **Data diversity** - Data needed for every fault type
- **Still needs human involvement** - Experts must annotate historical logs, validate model predictions, and adjust settings regularly.

SEMI-SUPERVISED LEARNING

- **Ignores environmental factors** - A model might misclassify elevated temperatures due to seasonal heat as a machine fault.
- **Limited real-time adaptability** - Adapting to a sudden change in workload or operational pattern requires retraining, not just reconfiguration.

NEED FOR EDGE?

ELIMINATES LATENCY – ENABLES REAL-TIME DETECTION

Problem: Many existing methods are slow because they depend on batch uploads to the cloud or offline processing.
How edge helps: Data is processed on-site as soon as it’s generated. This allows for immediate anomaly detection and response, especially critical in high-risk industrial settings.
Example: A spike in machine temperature triggers a local alert instantly - not minutes later when the data reaches the cloud.

WORKS IN LOW OR NO CONNECTIVITY ENVIRONMENTS

Problem: Many plants, especially remote or legacy ones, lack consistent internet.
How edge helps: Edge devices operate independently of connectivity. They continue to retrieve and buffer data, learning and monitoring locally, even when offline.
Example: An edge node in a rural wind farm continues detecting gearbox faults without needing cloud access.

NEED FOR EDGE?

HANDLES ENVIRONMENTAL CONTEXT LOCALLY

Problem: Existing methods take only machinery data and do not account for environmental factors, leading to false results based on the conditions.
How edge helps: Unsupervised learning across machines helps identify the current working conditions of the machine tailored to the environmental conditions.
Example: If a high ambient temperature is detected, higher motor heat is considered normal - preventing false alerts.

PROTECTS DATA PRIVACY AND REDUCES BANDWIDTH USE

Problem: Sending all machine data to the cloud is expensive and sometimes restricted.
How edge helps: Only relevant summaries, anomalies, or compressed reports are sent to the cloud, if needed at all.
Example: Instead of uploading 24/7 vibration logs, the edge device only pushes a clip that contains data about the deviations encountered.

OUR SOLUTION

4-STEP APPROACH

Raw Sensor Data → [1] Machine Comparison → [2] Fleet Clustering → [3] Anomaly Detection → [4] Visualization

Step 1: *Machine comparision*

- Pairwise similarity measurement
- Domain-specific preprocessing
- Normalization techniques

Step 2: *Fleet clustering*

- Hierarchical clustering
- Automatic cluster number determination
- Graph visualization for interpretability

Step 3: *Anomaly detection*

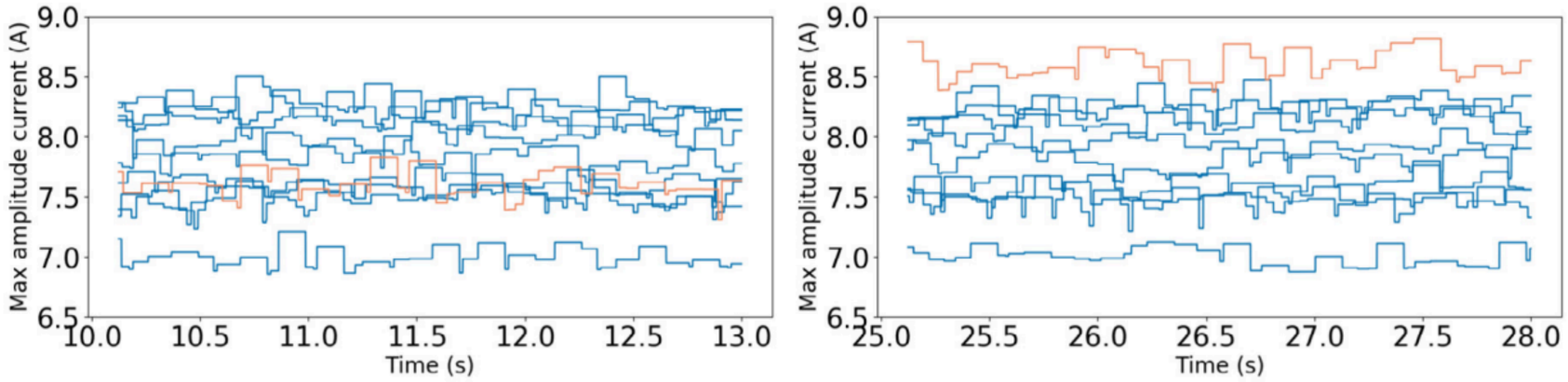
- Anomaly score = fraction of machines outside each cluster
- Assumption: majority of machines are healthy
- Threshold-based classification

OUR SOLUTION

4-STEP APPROACH

Step 4: Visualization

- Interactive dashboards for domain experts
- Signal analysis, similarity matrices
- Interpretable results for validation

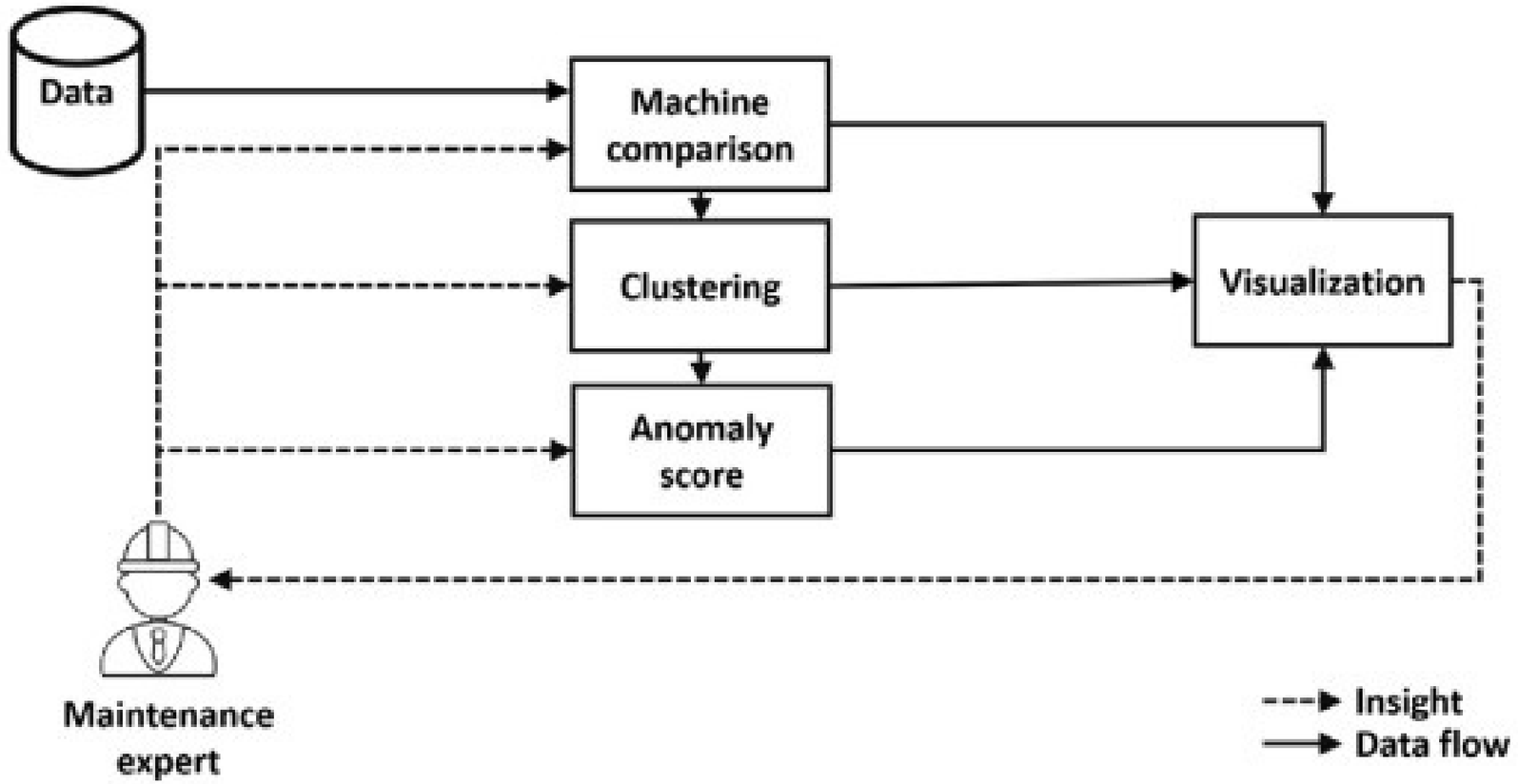


(a) All machines healthy

(b) D2_10 (orange) faulty

The above image shows the machine identfied as D2_10 flagged based on electrical signal behavior - specifically, the **Maximum amplitude current** over time.
Unsupervised Anomaly Detection: The system didn't require labeled fault data. It learned normal patterns and flagged the outlier (D2_10) when it deviated.

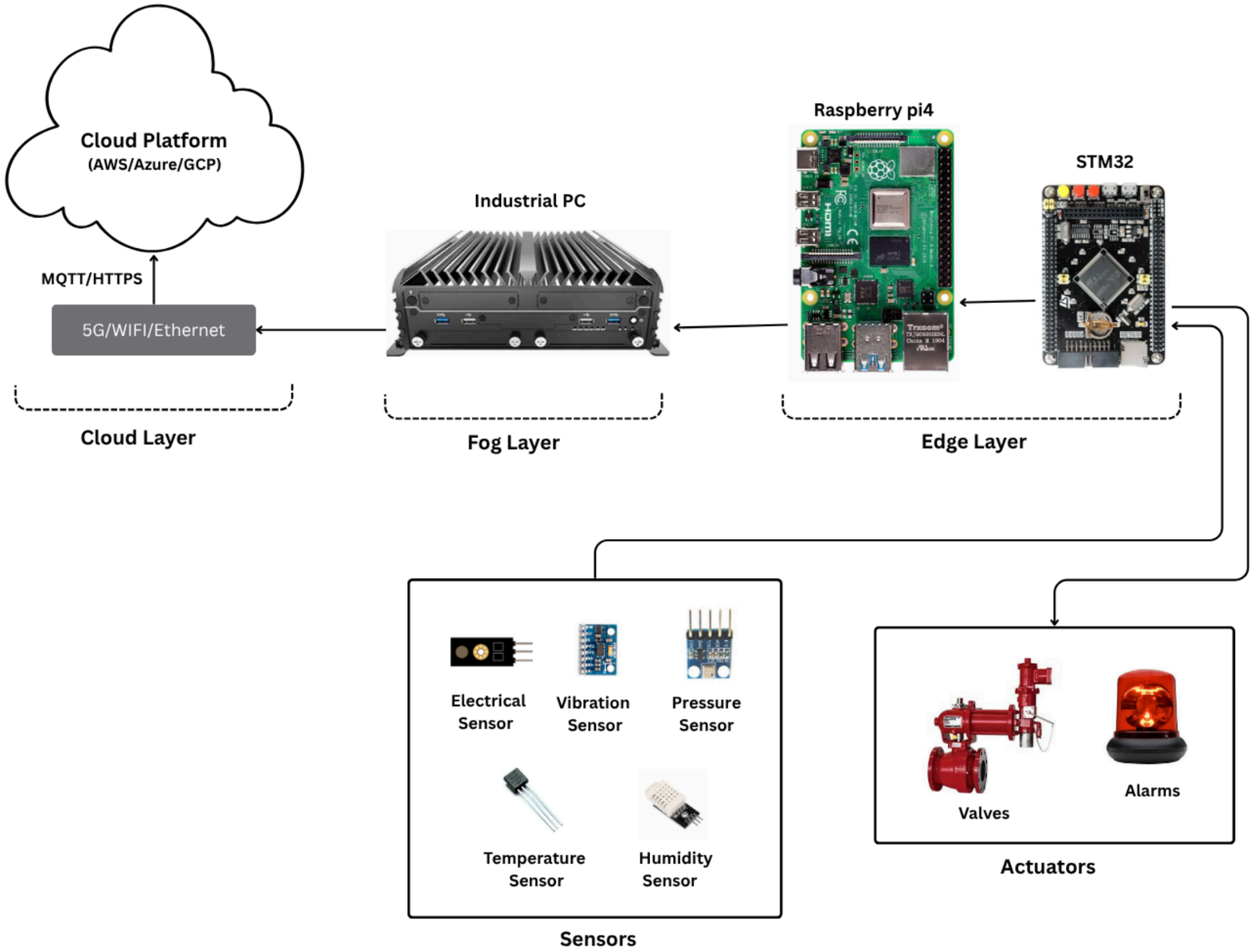
OUR SOLUTION



Assumptions:

- Majority of the machines in a cluster are healthy
- All machines have similar operating conditions
- There is a similar environment for all machines within a cluster

ARCHITECTURE DESIGN



STATE OF THE ART & LITERATURE

1. D'Amato et al. It validates different unsupervised techniques against expert knowledge and identifies effective methods like the Local Outlier Factor (LOF) for monitoring a fleet of assets.

2. Kholod et al. proposed using Federated Learning (FL) for anomaly detection in an industrial setting. In their approach, individual machines in the fleet collaboratively train a shared model without sending raw data to a central server, which is a key concept for modern edge applications.

3. Vercruyssen et al. provides a direct, empirical comparison of performing data-driven condition monitoring on edge devices versus in the cloud. Their research offers crucial insights into the trade-offs between these two architectures.

4. Chalapathy and Chawla offer a comprehensive review of how deep learning is used for anomaly detection. A significant portion of current research uses these methods. Their paper explains techniques where models learn a "representation of normality," which is an advanced version of the "majority is healthy" assumption in our paper.

5. Gou et al.(2024) They confirm that unsupervised methods are increasingly used for anomaly detection when failure data isn't available, validating the approach of our paper. Their work also highlights the importance of fleet history in sectors like the automotive industry.

TASK ALLOCATION FOR THE NEXT MONTH

Sarigasini M:

- Create dataset for different clusters.
- Visualise and verify it belongs to the specified cluster.
- Task scheduling and priority queue.

Dwarakesh Venkatesh:

- Implement algorithm for preprocessing data at STM-level.
- Find or create ML model for comparing two machines and clustering.

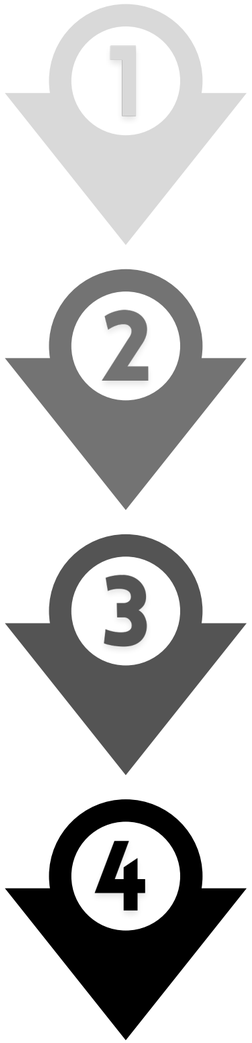
Shyam Sundar Raju:

- Setup simulation environment in Simpy.
- Integrates the AI models and algorithms.
- Cloud implementation.

Pooja Shree S:

- Implement heartbeat and sensor failure mechanism.
- Load balancing.
- Fog implementation.

NEXT STEPS



Implement basic entities in SimPy

Add preprocessing at STM-level

Simulate Edge-node analysis

Add Fog-layer along with failsafe methods and simulate Cloud actions

PROJECT GOALS

ACCURACY: >90% FAULT DETECTION RATE

LATENCY: DETECT FAULTS IN <5SEC ONCE IT OCCURS

SCALABILITY: SIMULATING MULTIPLE EDGE NODES

VISUALIZATION: PROVIDE VISUAL OUTPUT

ADDITIONAL INFORMATION

Throughput - Distributing workloads, Local data processing.

Load balancing - Each cluster is given its own edge processor, if one is overloaed, incoming data can easily be redirected.

Task scheduling - Critical operations (like gas leak or high temp) are given high priority, and processed immediatly while others are buffered

Reliability - Heartbeat mechanism is used to keep track of all processors, in case of network outage, processed data is stored in local buffer and transmitted later. Also,The edge divices act autonomously, cloud acts only as a storage medium and for report creations.

Clustering methodology:

- Data normalization
- Similarity measurement
- Heirarchial clustering