Enhancing the Evaluation of Fault Detection Models in Smart Agriculture Using LLM Agents for Rule-Based Anomaly Generation

Paolo Lindia[1,*], Riccardo Cantini[1], Francesco Bettucci[2], Luigi Sartori[2] and Paolo Trunfio[1,3]

[1]Università della Calabria, Rende (CS), Italy
[2]Università degli studi di Padova, Legnaro (PD), Italy
[3]Relatech SpA, Rende (CS), Italy

Abstract

In the context of Agriculture 4.0, advanced technologies such as the Internet of Things (IoT), artificial intelligence (AI), and big data analytics play a critical role in enhancing the efficiency and sustainability of farming operations. These innovations enable real-time monitoring and decision-making, improving the efficiency, sustainability, and productivity of agricultural systems. Central to Agriculture 4.0 is the deployment of sensors embedded in agricultural machinery, such as tractors, which continuously collect data on key operational metrics, including engine performance, fuel consumption, soil conditions, and equipment health. The effective analysis of such data is essential for predictive maintenance, as early detection of potential anomalies can prevent costly breakdowns and reduce downtime. However, finding real-world datasets containing examples of anomalies in agricultural machinery is highly challenging, making it difficult to develop and assess the effectiveness of anomaly detection models. Additionally, classical methods for anomaly generation, such as stochastic and adversarial approaches, may be difficult to apply given the intricate patterns and time dependency of these data. To address this gap, our work leverages Large Language Models (LLMs) and agentic workflows to generate realistic anomaly scenarios from agricultural data. Using a rule-based approach that combines prompt engineering techniques with a multi-agent system, we create synthetic anomalies that can later be used to evaluate anomaly detection models. These models would then enable the timely identification of potential machinery failures, reducing maintenance costs, minimizing downtime, and significantly lowering the environmental impact by preventing inefficiencies such as increased fuel consumption from faulty equipment, reducing the need for replacement parts, and conserving energy and resources used in repairs.

Keywords

Smart Agriculture, Large Language Models, Agentic Workflows, Predictive maintenance, Green AI, Environmental Sustainability, Internet of Things, Anomaly Detection, Anomaly Generation

1. Introduction

IoT sensor networks are increasingly leveraged in Industry 4.0 and Smart Agriculture to enhance productivity and sustainability through advanced sensing, data fusion, and machine learning. In this context, anomaly detection techniques can be effectively applied for real-time monitoring of machinery and systems, preventing failures and optimizing operational efficiency [1].

In Smart Agriculture, anomaly detection techniques primarily rely on multivariate streams of sensor data, consisting of measurements taken from multiple sensors at regular intervals. Due to the unique challenges inherent in IoT sensor data, such as temporal and spatial correlations, high dimensionality, and inherent noise, recent techniques increasingly rely on deep learning methods. Specifically, autoencoders and recurrent or convolutional neural networks have been employed for their ability to handle complex

1st Workshop on Green-Aware Artificial Intelligence, 23rd International Conference of the Italian Association for Artificial Intelligence (AIxIA 2024), November 25–28, 2024, Bolzano, Italy

*Corresponding author.

Authors contribution: P.L., R.C., P.T.: *Conceptualization*, *Investigation*, *Methodology*, *Software*, *Validation*; F.B., L.S.: *Data curation*, *Validation*.

paolo.lindia@dimes.unical.it (P. Lindia); rcantini@dimes.unical.it (R. Cantini); francesco.bettucci@phd.unipd.it (F. Bettucci); luigi.sartori@unipd.it (L. Sartori); trunfio@dimes.unical.it (P. Trunfio)
0000-0002-7550-1331 (P. Lindia); 0000-0003-3053-6132 (R. Cantini); 0009-0001-3758-1158 (F. Bettucci); 0000-0001-6437-3402 (L. Sartori); 0000-0002-5076-6544 (P. Trunfio)

and noisy datasets [2, 3, 4]. Despite the demonstrated effectiveness of such methods, identifying representative anomalous data for testing purposes remains a significant challenge, particularly in IoT settings where data is spatiotemporal and real-world anomalies are often rare or difficult to observe. Anomaly generation becomes therefore crucial in overcoming this challenge by enabling the creation of synthetic anomalies that closely resemble real-world data distributions. Classical methods for anomaly generation, such as rule-based or stochastic approaches, often fail to capture the complex dependencies between spatial and temporal features, resulting in unrealistic anomalies. In addition, while more sophisticated techniques like adversarial methods and latent models can generate realistic data, they are computationally expensive and require extensive tuning, which may hinder their application in this domain.

To address these limitations, we propose a novel rule-based anomaly generation approach that leverages the context-aware capabilities of Large Language Models (LLMs). Our methodology extends beyond a single LLM by employing LLM agents in a collaborative workflow, where each agent contributes specialized knowledge to produce the final synthetic anomalies. By incorporating LLM agents into the rule generation process, we enable a more informed, context-driven creation of anomalies that better reflect the spatiotemporal complexities of IoT sensor data. This hybrid approach combines the interpretability and simplicity of rule-based methods with the nuanced understanding and adaptability of LLM agents, resulting in a more efficient and realistic anomaly generation process suitable for testing detection algorithms in dynamic, real-world environments.

The main contributions of the paper can be summarized as follows:

- We advance the application of LLM agents in Smart Agriculture, showing how such systems can cooperate within an agentic workflow to generate realistic synthetic anomalies.
- The proposed method integrates a rule-based approach with the capabilities of LLMs, addressing the limitations of traditional methods in handling high-dimensional spatiotemporal IoT data.
- Our approach enhances the testing of anomaly detection systems, leading to more reliable real-time monitoring and improved operational efficiency.

The remainder of the paper is organized as follows. In Section 2, we discuss related work in the field of anomaly generation, highlighting the main applications of LLMs to Smart Agriculture. Section 3 provides an in-depth description of the proposed approach showing its application to a real-world case study. Finally, Section 4 concludes the paper.

2. Related Work

Large Language Models (LLMs) have recently gained significant traction due to their remarkable natural language understanding and generation capabilities [5, 6, 7]. These systems are increasingly being integrated into Smart Agriculture, providing powerful tools for data-driven decision-making and precision farming. Conversational assistants powered by LLM agents provide farmers and agricultural professionals with insights drawn from vast datasets to support resource management, enhance crop health, and optimize environmental conditions, thereby improving productivity and sustainability [8, 9].

In this work, we explore how LLM-based agents can be synergistically leveraged in the field of smart agriculture to generate synthetic real-world anomalies. This task is critical for improving and evaluating the performance of anomaly detection systems. Several methodologies have been developed to generate synthetic anomalies that closely resemble real-world scenarios, enabling a robust assessment of detection algorithms. Major approaches in the literature leverage conditional generation approaches and *Generative Adversarial Networks* (GANs), in which two neural networks—a generator and a discriminator—compete with each other during the training process. Specifically, the discriminator tries to create realistic synthetic data, i.e., anomalous instances, while the discriminator tries to differentiate between normal and anomalous data. This process leads to the generation of highly realistic anomalies that closely resemble actual outliers, making GANs particularly useful in testing the robustness of anomaly detection systems. As an example, Uzolas et al. leverage conditional GANs for

the generation of realistic single-chromosome images following user-defined banding patterns [10], while Salem et al. [11] uses a Cycle-GAN to generate synthetic anomalous data from normal data for improving anomaly detection in imbalanced datasets. Zhang et al. [12] introduce DefectGAN, which generates anomaly samples by superimposing learned defect foregrounds onto a normal background, while Niu et al. propose SDGAN [13], which modifies defect-free images to introduce surface defects using a generator trained with cycle consistency loss on both normal and anomalous images. Duan et al. [14] introduce a few-shot defect image generation technique, producing structural anomalies from a limited set of defect samples. It enhances a pre-trained StyleGAN2 backbone by adding defect-aware residual blocks to manipulate features within learned defect masks.

Besides GANs, *Diffusion Models* (DMs) have been also leveraged for generating synthetic anomalies by perturbing normal patterns. DMs are a family of probabilistic generative models that progressively add noise to data and then learn to reverse this process to generate new samples. In the field of anomaly generation, Dai et al. present GRAD [15], an unsupervised anomaly detection framework using a diffusion model called PatchDiff to generate contrastive patterns by disrupting global structures while preserving local ones. GRAD also includes a self-supervised reweighting mechanism and a lightweight detector to efficiently identify anomalies. Hu et al. [16] propose a diffusion-based few-shot anomaly generation model, leveraging the strong prior knowledge of a latent diffusion model trained on large datasets to improve the realism of generated anomalies. Zhang et al. introduce RealNet [17], another diffusion-based approach that relies on Strength-controllable Diffusion Anomaly Synthesis (SDAS) to generate synthetic anomalies of varying strengths, mimicking real-world anomalies. RealNet also incorporates feature selection and residual detection methods to improve anomaly detection while managing computational cost, showing significant improvements on several benchmark datasets.

While these anomaly synthesis methods are effective, they depend on real defect images and cannot generate unseen types of anomalies. Furthermore, these methods are usually computationally intensive and often require extensive tuning to produce meaningful results.

3. Proposed Approach: Leveraging LLM Agents for Anomaly Generation in Agricultural Machinery

In this section, we provide a detailed description of the proposed approach aimed at generating real-world anomalies in multivariate sensor data from agricultural machinery, specifically tractors. We leverage an agentic workflow in which different LLM agents interact with each other to produce high-quality anomalous test data. The proposed methodology is articulated in two main phases:

1. *Best LLM selection via zero-shot operational range generation* — First, the best LLM must be selected from all those available, including GPT-4o and LLama3.1. For this purpose, CAN bus sensor data from tractors are analyzed to extract the operational ranges of the different variables considered. By comparing these real ranges with those generated by various Large Language Models (LLMs) through zero-shot prompting, we identify the LLM that exhibits the highest level of expertise in the domain of agriculture and tractor operations.
2. *Anomaly generation through an agentic workflow* — The methodology employs an agentic workflow to generate anomalies, which involves collaboration between two LLM-based agents: (*i*) the first agent generates anomaly rules based on insights from the selected LLM; (*ii*) the second agent transforms the generated rules into executable Python code. This code applies the anomalies to the original non-anomalous data, effectively simulating real-world deviations and faults.

Finally, as the test anomalies are generated, they are used to assess the performance of deep learning-based anomaly detection models. Specifically, an LSTM-based autoencoder is trained on a dataset representing a work session of the tractor and then tested against the synthetic anomalies generated as described above. This approach mimics real-world processes of anomaly detection in agricultural machinery, allowing for an assessment of the effectiveness of the generated anomalies.

3.1. Best LLM selection via zero-shot operational range generation

Figure 1 depicts the flowchart used in the first phase of the methodology, dedicated to selecting the LLM that exhibits the highest expertise in the agricultural domain, specifically regarding tractors and their sensor data. The selection process involves several LLMs, specifically *GPT-4o*, *Llama 3.1 70B*, *Gemini Pro*, and *Mistral Large 2*. Their effectiveness is measured by their ability to generate operational ranges for key tractor variables, which are then compared to the actual ranges extracted from tractor sensor data.

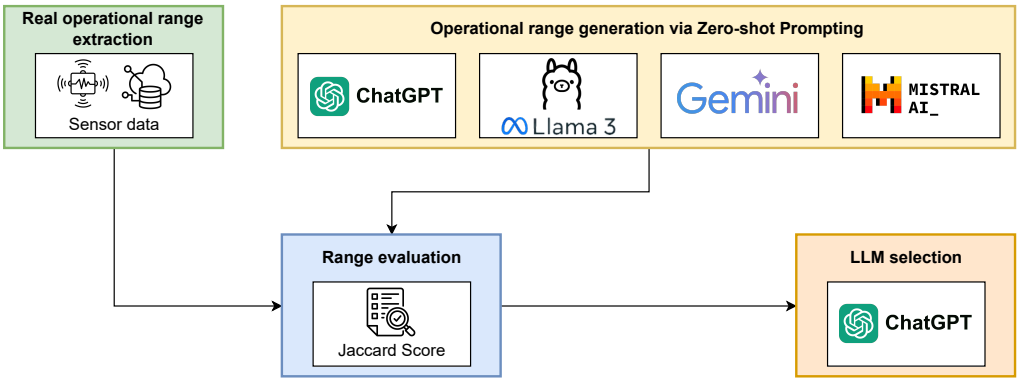


Figure 1: Best LLM selection via zero-shot operational range generation.

In the following yellow box, we report the prompt used for querying the different LLMs to generate operational ranges of variables. Each model is provided with a prompt containing the variable name, its unit of measurement, and a description. Generation is performed through *zero-shot prompting*, which means that the prompt used to interact with the model does not include any example or demonstration.

> As a seasoned expert in New Holland T7 165 S tractors, we seek your expertise in diagnosing various operational variables retrieved from the CAN bus of the tractor. You are provided with a list of variables, each with its name, unit of measurement, and description. These variables are listed according to the following format: `- <var_name> (<unit>): <description>`. Your task is to generate the operational range of each variable, which jointly takes into account the different activities performed by the tractor, i.e. *idling*, *moving*, *plowing*, and *turning*.
>
> Format your output as follows:
>
> - `<var_name>: <operational_range> (<unit>)`
> - ...
>
> Input variables:
>
> - CAN1.LFE1.EngineFuelRate (l/h): Amount of fuel consumed by the engine per unit of time.
> - CAN1.EFLP1.EngineOilPressure1 (kPa): Gage pressure of oil in the engine lubrication system as provided by the oil pump.
> - ...

Table 1 presents the operational ranges generated by each LLM for the various key variables associated with tractor sensor data. Each row of the table details the ranges produced by the evaluated models for a given variable, while the final column provides the actual ranges extracted from the sensor data. This comparative analysis highlights the discrepancies and alignments between the internal knowledge of LLMs and real-world data, which are crucial for determining the most effective LLM for the subsequent phases of the methodology.

To quantitatively assess the accuracy of LLM-generated ranges, we compared them with ground truth values derived from tractor sensor data by introducing a continuous version of the Jaccard index that quantifies the similarity between two ranges. Given two intervals $[l_1, u_1]$ and $[l_2, u_2]$, where l_1 and u_1 represent the lower and upper bounds of the first interval, and l_2 and u_2 represent the bounds of the

Id	Features	GPT-4o	Llama3.1 70B	Gemini-Pro	Mistral Large 2	Real ranges
F1	CAN1.LFE1.EngineInstantaneousFuelEconomy (km/l):	1.5 - 12 km/l	2 - 10 km/l	0.2 - 0.8 km/l	0.5 - 3.0 km/l	0 - 125.5 km/l
F2	CAN1.EEC3.AftrtrtmntExhstGsMssFlwRt (kg/h):	30 - 800 kg/h	10 - 100 kg/h	10 - 200 kg/h	0 - 500 kg/h	0 - 693.97 kg/h
F3	CAN1.EEC2.AtlMxmmAvllEngnPrntTrq (%):	50 - 100 %	20 - 80 %	10 - 100 %	0 - 100 %	0 - 99.15 %
F4	CAN1.FD1.FanSpeed (rpm):	500 - 2500 rpm	500 - 1500 rpm	500 - 1800 rpm	0 - 5000 rpm	0 - 2109.0 rpm
F5	CAN1.EEC3.EnginesDesiredOperatingSpeed (rpm):	600 - 2200 rpm	1500 - 2500 rpm	700 - 2200 rpm	500 - 2500 rpm	1000 - 1067.5 rpm
F6	CAN1.LFE1.EngineFuelRate (l/h):	3 - 40 l/h	10 - 50 l/h	5 - 70 l/h	0 - 200 l/h	0 - 29.35 l/h
F7	CAN1.IC1.EngineIntakeAirPressure (kPa):	90 - 200 kPa	80 - 120 kPa	100 - 150 kPa	80 - 120 kPa	98 - 202 kPa
F8	CAN1.CCVS1.WheelBasedVehicleSpeed (km/h):	0 - 40 km/h	0 - 40 km/h	0 - 50 km/h	0 - 60 km/h	0 - 18.53 km/h
F9	CAN2.TSC1.EngnRqstdTrqTrqLmt_0 (%):	0 - 100 %	20 - 50%	10 - 100 %	0 - 100 %	0 - 100 %
F10	CAN1.EEC1.EngineSpeed (rpm):	600 - 2200 rpm	1500 - 2500 rpm	700 - 2200 rpm	500 - 2500 rpm	0 - 2217 rpm
F11	CAN2.gnss_speed.Speed (m/s):	0 - 15 m/s	0 - 10 m/s	0 - 14 m/s	0 - 60 m/s	0 - 4.94 m/s
F12	CAN1.VEP1.BatteryPotentialPowerInput1 (V):	11 - 14.5 V	12 - 14 V	12 - 15 V	12 - 24 V	11.37 - 14.19 V
F13	CAN1.A1SCRDSI1.Atttt1DsExstFdAtDsQtt (g/h):	0 - 20 g/h	10 - 50 g/h	0 - 60 g/h	0 - 20 g/h	0 - 7051.2 g/h
F14	CAN1.EEC1.EngineDemandPercentTorque (%):	0 - 100 %	20 - 80 %	0 - 100 %	0 - 100 %	0 - 99 %
F15	CAN1.EEC1.ActualEnginePercentTorque (%):	0 - 100 %	20 - 80 %	10 - 100 %	0 - 100 %	0 - 98 %
F16	CAN1.EEC3.NominalFrictionPercentTorque (%):	5 - 15 %	10 - 30 %	5 - 40 %	5 - 40 %	6 - 11 %
F17	CAN1.FD1.EngineFan1EstimatedPercentSpeed (%):	0 - 100 %	20 - 80 %	0 - 100 %	0 - 100 %	0 - 73.06 %
F18	CAN1.IC1.EngnIntkMnfld1Tmprtr (degC):	-10 - 80 degC	40 - 80 degC	20 - 90 degC	-20 - 120 degC	14 - 67 degC
F19	CAN1.EEC1.AtlEngnPrntTrqFrtnl (%):	0 - 0.875 %	0 - 1 %	0 - 0.875 %	0 - 0.875 %	0 - 0.875 %
F20	CAN2.gnss_attitude.Heading (deg):	0 - 360 deg	0 - 360 deg	0 - 359 deg	0 - 360 deg	0.23 - 359.8 deg
F21	CAN2.TSC1.EngnRqstdTrqTrqLmt_3 (%):	0 - 100 %	50 - 80 %	10 - 100 %	0 - 100 %	0 - 100 %
F22	CAN1.IC1.EngineIntakeManifold1Pressure (kPa):	90 - 200 kPa	80 - 120 kPa	50 - 150 kPa	80 - 120 kPa	98 - 202 kPa
F23	CAN1.EEC2.EnginePercentLoadAtCurrentSpeed (%):	0 - 100 %	20 - 80 %	0 - 100 %	0 - 100 %	0 - 100 %
F24	CAN1.TSC1.EngnRqstdTrqTrqLmt (%):	0 - 100 %	20 - 80 %	0 - 100 %	0 - 100 %	0 - 99 %
F25	CAN1.EFLP1.EngineOilPressure1 (kPa):	100 - 500 kPa	300 - 500 kPa	200 - 800 kPa	0 - 1000 kPa	96 - 536 kPa
F26	CAN1.VEP1.KeySwitchBatteryPotential (V):	11 - 14.5 V	12 - 14 V	12 - 15 V	12 - 24 V	11.37 - 14.19 V

Table 1
Zero-shot generation of the operational range for the selected features using different LLMs. Real ranges, extracted from sensor data, are shown for reference.

second interval, the *Jaccard similarity* (J) for intervals is defined as follows. Let:

- $U = \max(u_1, u_2) - \min(l_1, l_2)$ be the *union* of the two intervals (i.e., total covered range length).
- $I = \max(0, \min(u_1, u_2) - \max(l_1, l_2))$ be the *intersection* of the intervals, which is calculated based on the overlap between the intervals. $I = 0$ if the intervals do not overlap. Otherwise, I represents the length of the overlapping interval.

Then, the *Jaccard similarity* for intervals can be expressed as $J([l_1, u_1], [l_2, u_2]) = \frac{I}{U}$, with $J \in [0, 1]$ where 0 means no overlap, and 1 means the intervals are identical.

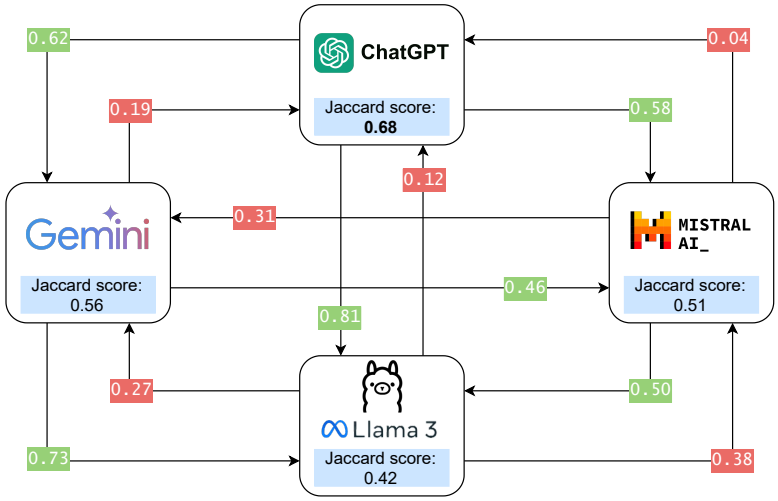


Figure 2: LLM win rates graph with average Jaccard interval score achieved by the tested LLMs. Each edge $\mathscr{M}_i \rightarrow \mathscr{M}_j$ represents the percentage of features where $\mathscr{M}_i$ achieved a higher Jaccard score compared to $\mathscr{M}_j$.

For each variable, the average Jaccard similarity score was calculated across all comparisons between the real and generated ranges. The LLM with the highest average Jaccard score was selected as the most appropriate model for generating anomaly rules in the subsequent steps of the proposed methodology. Figure 2 illustrates the win rates of the evaluated LLMs alongside the average Jaccard interval scores achieved by each model. The plot shows that GPT-4o consistently outperforms all other models and demonstrates good accuracy in generating intervals that closely resemble the actual operational ranges extracted from tractor sensor data, confirming its suitability as the chosen model.

3.2. Anomaly generation through an agentic workflow

Once the most appropriate LLM is selected, the anomaly generation process is performed through an agentic workflow, as illustrated in Figure 3.

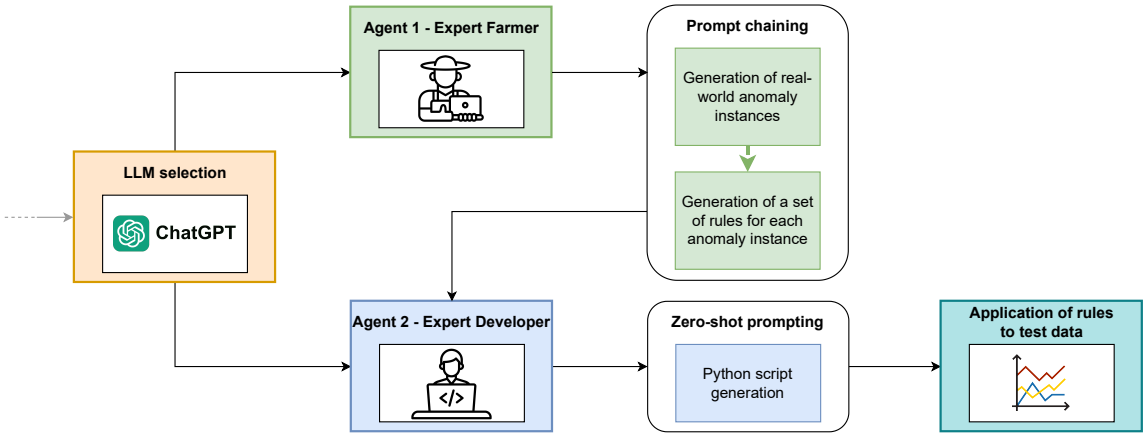


Figure 3: Agentic workflow for anomalies generation

This workflow involves two LLM-based agents:

- *Expert farmer*: Its role is to generate realistic cases of anomalies in the form of rules that can be applied to test data, resulting in anomalous test instances.
- *Expert developer*: Its role is to convert the set of rules generated by the expert farmer into a runnable Python script, which can be executed, via tool use, on the test dataset to produce a structured set of anomalous test instances for benchmarking anomaly detection methods.

In the following sections, the prompts used to query the LLM-based agents are shown, along with the generated output.

3.2.1. Expert farmer Agent — Anomaly generation via prompt chaining

In this step, the *prompt chaining* technique is employed to generate meaningful anomaly instances, as indicated by the green-colored boxes. Using *prompt chaining*, a sequence of prompts generates complex outputs by linking multiple tasks together. Initially, the first agent (i.e., the *expert farmer*) generates a set of significant anomaly cases across various activities, such as *plowing*, *moving*, *turning*, and *idle* operations. These anomalies are then used to create rules that modify the operational ranges of the variables, thereby generating anomalies. For each anomaly, a corresponding rule is created that specifies its duration and how the operational ranges are altered to simulate the anomaly within the data. These rules are then passed to the second agent (i.e., the *expert developer*) for further processing.

As a seasoned expert in New Holland T7 165 S tractors, we seek your expertise in diagnosing various operational variables retrieved from the CAN bus of the tractor. You are provided with a list of variables, each with its name, operational range, unit of measurement and description. These variables are listed according to the following format: `- <var_name> (<operational_range> <unit>): <description>`. Your task is to generate instances of significant anomalies based on the activity performed by the tractor, i.e., "plowing," "moving," "turning," "starting," and "idling". Each anomaly instance must include:

- a description of the anomaly instance
- the list of variables involved in the anomaly instance
- the activity performed by the tractor when the anomaly shows up.

Format your output as follows:

```
- <instance_name>: <description>
    - variables involved:
        - <var_name>
        - ...
    - <activity_performed>
- ...
```

Input variables:

- CAN1.LFE1.EngineFuelRate (0 - 29.35 l/h): Amount of fuel consumed by the engine per unit of time.
- CAN1.EFLP1.EngineOilPressure1 (96 - 536 kPa): Gage pressure of oil in the engine lubrication system as provided by the oil pump.
- ...

Based on: (*i*) the generated anomaly instances, (*ii*) the descriptions, (*iii*) the activities performed, and (*iv*) the operational range of the involved variables, generate a set of rules for each anomaly instance describing how each variable involved varies numerically. Also, specify the overall duration of the anomaly for each instance. Consider that the session in which the anomalies will be applied lasts approximately 2 hours, with observations recorded at a frequency of 1 Hz.

Format your output as follows:

```
- <instance_name> (<activity_performed>):
    - <duration>
    - rules:
        - <var_name>: <rule_description>
        - ...
- ...
```

Table 2 presents the output generated by the first agent. Specifically, the following information is reported:

- The anomaly name, which concisely describes the issue.
- The performed activity during which the anomaly occurs.
- An issue description that provides useful details on how the anomaly affects the normal operation of the tractor.
- The duration of the anomaly.
- The variables affected.
- The associated rules specifying how each variable deviates from its expected range over time.

ID	Anom. Name	Activity	Issue description	Dur.	Involved features	Rule descriptions
1	Fuel Consumption Spike	Plowing	The tractor shows unusually high fuel consumption during operation, despite consistent speed and load. Instantaneous fuel economy drops sharply, and the fuel rate is well above normal.	10 min	CAN1.LFE1.EngineInstantaneousFuelEconomy	Drops below 20 km/l from a normal range of 50-70 km/l.
					CAN1.LFE1.EngineFuelRate	Increases to above 20 l/h from a normal range of 5-10 l/h.
					CAN1.EEC2.EnginePercentLoadAtCurrentSpeed	Increases to above 80% from a normal range of 30-50%.
2	Overheating Engine	Moving	The tractor's engine temperature suddenly rises above normal limits, increasing fan speed to compensate. Since intake air temperature and pressure remain normal, this indicates a potential issue with the cooling system.	20 min	CAN1.IC1.EngnIntkMnfld1Tmprtr	Rises to above 67°C from a normal range of 20-40°C.
					CAN1.FD1.FanSpeed	Increases to above 2000 rpm from a normal range of 1200-1600 rpm.
					CAN1.EFLP1.EngineOilPressure1	Drops below 100 kPa from a normal range of 200-400 kPa.
					CAN1.EEC1.EngineSpeed	Fluctuates between 1800-2200 rpm from a normal steady range of 1500-1700 rpm.
3	Torque Instability	Turning	The tractor's engine torque output fluctuates, causing jerky movements and inefficient performance. A misalignment between requested and actual torque values suggests an issue with the engine control system.	15 min	CAN2.TSC1.EngnRqstdTrqTrqLmt_0	Varies erratically between -50% and 100% from a normal steady range of 20-40%.
					CAN1.EEC1.EngineDemandPercentTorque	Fluctuates between 0% and 99% from a normal steady range of 30-60%.
					CAN1.EEC1.ActualEnginePercentTorque	Deviates between 0% and 98% from a normal steady range of 30-60%.
4	Battery Voltage Drop	Idle	The tractor's battery voltage drops below the normal range, potentially causing electrical issues like erratic behavior of control units.	30 min	CAN1.VEP1.BatteryPotentialPowerInput1	Drops below 11.37 V from a normal range of 12.5-14 V.
					CAN1.VEP1.KeySwitchBatteryPotential	Drops below 11.37 V from a normal range of 12.5-14 V.

Table 2
Anomaly instances generated by GPT-4o. Each instance includes a description and a set of associated features.

3.2.2. Expert developer Agent — Python script generation and application of rules to test data

The second agent, acting as a Python programming expert, is prompted to transform the anomaly rules, generated by the expert farmer LLM agent, into an executable Python script.

> As an expert Python developer, we seek your assistance in code scripting. You are provided with a set of rules for different anomaly instances that describe how each variable involved varies numerically, along with the overall duration of the anomaly. Anomaly instances are listed according to the following format:
>
> - `<instance_name> (<activity_performed>):`
> - `<duration>`
> - rules:
> - `<var_name>: <rule_description>`
> - ...
>
> Based on this information, generate a Python function that applies a given anomaly instance to a time series of sensor data. The code must adhere to the following requirements:
>
> - all anomaly instances are handled;
> - random values are used instead of fixed anomalous values;
> - the input dataframe is read from a csv given as input; the start time and the anomaly to be applied are given as input;
> - output the required function without any example usage.
>
> Input anomaly instances:
>
> - Fuel Consumption Spike (Plowing):
> - 10 minutes
> - rules:
> - CAN1.LFE1.EngineInstantaneousFuelEconomy: Drops below 20 km/l from a normal range of 50-70 km/l.
> - CAN1.LFE1.EngineFuelRate: Increases to above 20 l/h from a normal range of 5-10 l/h.
> - CAN1.EEC2.EnginePercentLoadAtCurrentSpeed: Increases to above 80% from a normal range of 30-50%.
> - ...

In this case, as shown in the blue-colored box, *zero-shot prompting* is employed, wherein the agent generates a Python script based on the provided anomaly rules without any prior examples or specific training data. The script is designed to take the clean, non-anomalous test dataset as input and apply the anomalies according to the rules generated by the first agent. The generated script is executed to create four distinct datasets by applying the anomalies to the test dataset for each possible activity.

Through this agentic workflow, the entire process of anomaly rule generation and application can be automated, providing a robust method for simulating consistent anomalous behaviors. This, in turn, supports the evaluation of anomaly detection models, by providing realistic and domain-specific anomalies that accurately reflects potential issues that could arise in real-world operations.

3.3. Auto-encoder evaluation on synthetic test anomalies

Here we show how the previously generated anomalous test datasets can be effectively leveraged to assess the effectiveness of a deep learning-based anomaly detection model. In particular, for each possible activity, including plowing, moving, turning, or idle, an LSTM autoencoder is trained on a normal working session, encompassing non-anomalous data from CAN bus sensors (see Figure 4).

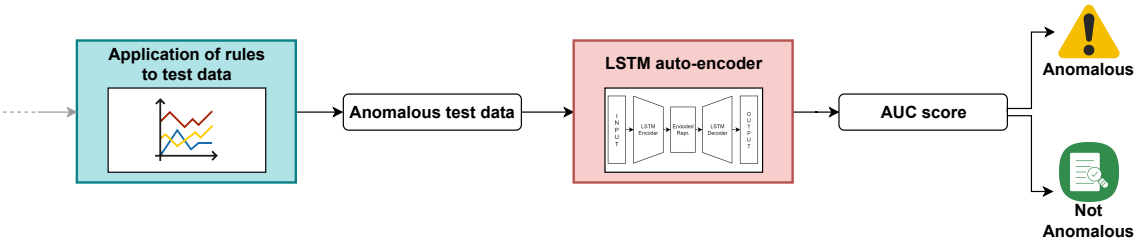
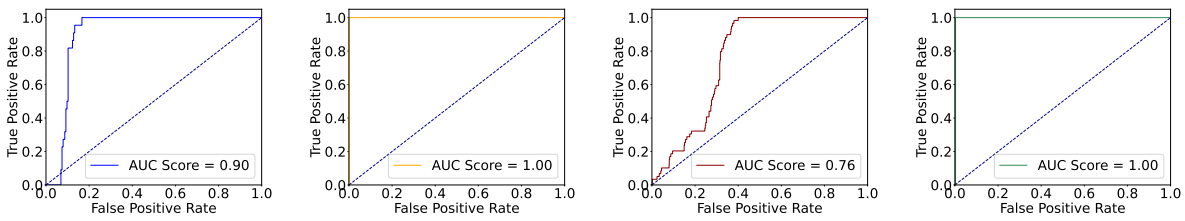


Figure 4: LSTM auto-encoder testing on anomalous generated data for a given activity.

The LSTM autoencoder works by reconstructing the input time series. A large reconstruction error suggests that the input data may deviate from normal patterns, indicating an anomaly. The detection performance of each autoencoder is measured using the Area Under the Receiver Operating Characteristic Curve (AUC) score. It ranges from 0 to 1, where a score of 1 indicates perfect separation between anomalies and normal data, while 0.5 suggests that the model is equivalent to random guessing.

Figure 5 presents the ROC curves for the four anomalous instances considered during the anomaly generation process. Each curve illustrates the model's ability to distinguish between anomalous and non-anomalous data across a diverse set of potential scenarios. Specifically, two cases (figure 5b and 5d) achieve perfect classification (AUC = 1.00), while the other two cases (figure 5a and 5c) show strong (AUC = 0.90) and moderate (AUC = 0.76) performance, respectively. These results suggest that the model is highly effective in detecting anomalies, with some variability depending on the specific type of anomaly and the amount of training data from sensors. Furthermore, the ability to generate activity-specific test data facilitates a more granular analysis of model performance, providing insights into how different types of anomalies might be detected in real-world deployments.



(a) Fuel Consumption Spike (b) Overheating Engine (c) Torque Instability (d) Battery Voltage Drop

Figure 5: LSTM auto-encoder evaluation for each anomaly instance.

4. Conclusion

In this work, we advance the application of LLM agents in Smart Agriculture by proposing a rule-based approach for the automatic generation of synthetic anomalies in agricultural machinery. By generating realistic, domain-specific anomalies, the system creates a rich dataset that accurately reflects potential issues that could arise in real-world operations. This enables effective evaluation of anomaly detection models and allows researchers and developers to test their algorithms against a variety of plausible scenarios. The generated datasets support thorough benchmarking, helping to identify the strengths and weaknesses of different anomaly detection methods. Moreover, the ability to generate diverse datasets tailored to specific activities—such as plowing, moving, turning, and idling—facilitates more granular analysis of model performance. This can lead to insights into how different types of anomalies might affect operational efficiency, safety, and tractor maintenance. Ultimately, the proposed methodology fosters an iterative feedback loop, where the performance of anomaly detection models can be continuously improved based on simulated data. This enhances their robustness and reliability in real-world applications, ensuring efficient utilization of agricultural resources and paving the way for more sustainable agricultural practices. Future work will focus on integrating domain-specific knowledge through agentic RAG (Retrieval-Augmented Generation), further improving context awareness of the system and enabling LLMs to better comprehend complex scenarios.

Acknowledgments

This work has been funded by the project "AGRITECH: National Research Centre for Agricultural Technologies" - CUP CN00000022, of the National Recovery and Resilience Plan (PNRR) financed by the European Union "Next Generation EU", and by the "FAIR – Future Artificial Intelligence Research" project - CUP H23C22000860006.

References

[1] A. A. Cook, G. Mısırlı, Z. Fan, Anomaly detection for iot time-series data: A survey, IEEE Internet of Things Journal 7 (2019) 6481–6494.

[2] D. Kim, H. Yang, M. Chung, S. Cho, H. Kim, M. Kim, K. Kim, E. Kim, Squeezed convolutional variational autoencoder for unsupervised anomaly detection in edge device industrial internet of things, in: 2018 international conference on information and computer technologies (icict), IEEE, 2018, pp. 67–71.

[3] Y. Guo, W. Liao, Q. Wang, L. Yu, T. Ji, P. Li, Multidimensional time series anomaly detection: A gru-based gaussian mixture variational autoencoder approach, in: Asian Conference on Machine Learning, PMLR, 2018, pp. 97–112.

[4] T. Kieu, B. Yang, C. S. Jensen, Outlier detection for multidimensional time series using deep neural networks, in: 2018 19th IEEE international conference on mobile data management (MDM), IEEE, 2018, pp. 125–134.

[5] T. B. Brown, Language models are few-shot learners, arXiv preprint arXiv:2005.14165 (2020).

[6] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, Y. Iwasawa, Large language models are zero-shot reasoners, Advances in neural information processing systems 35 (2022) 22199–22213.

[7] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, et al., A survey of large language models, arXiv preprint arXiv:2303.18223 (2023).

[8] I. N. Glukhikh, T. Y. Chernysheva, Y. A. Shentsov, Decision support in a smart greenhouse using large language model with retrieval augmented generation, in: Third International Conference on Digital Technologies, Optics, and Materials Science (DTIEE 2024), volume 13217, SPIE, 2024, pp. 166–173.

[9] R. K. Raman, A. Kumar, S. Sarkar, A. K. Yadav, A. Mukherjee, R. S. Meena, U. Kumar, D. Singh, S. Das, R. Kumar, et al., Reconnoitering precision agriculture and resource management: A

comprehensive review from an extension standpoint on artificial intelligence and machine learning, Indian Research Journal of Extension Education 24 (2024) 108–123.

[10] L. Uzolas, J. Rico, P. Coupé, J. C. SanMiguel, G. Cserey, Deep anomaly generation: An image translation approach of synthesizing abnormal banded chromosome images, IEEE Access 10 (2022) 59090–59098.

[11] M. Salem, S. Taheri, J. S. Yuan, Anomaly generation using generative adversarial networks in host-based intrusion detection, in: 2018 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), IEEE, 2018, pp. 683–687.

[12] G. Zhang, K. Cui, T.-Y. Hung, S. Lu, Defect-gan: High-fidelity defect synthesis for automated defect inspection, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2021, pp. 2524–2534.

[13] S. Niu, B. Li, X. Wang, H. Lin, Defect image sample generation with gan for improving defect recognition, IEEE Transactions on Automation Science and Engineering 17 (2020) 1611–1622.

[14] Y. Duan, Y. Hong, L. Niu, L. Zhang, Few-shot defect image generation via defect-aware feature manipulation, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 37, 2023, pp. 571–578.

[15] S. Dai, Y. Wu, X. Li, X. Xue, Generating and reweighting dense contrastive patterns for unsupervised anomaly detection, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, 2024, pp. 1454–1462.

[16] T. Hu, J. Zhang, R. Yi, Y. Du, X. Chen, L. Liu, Y. Wang, C. Wang, Anomalydiffusion: Few-shot anomaly image generation with diffusion model, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, 2024, pp. 8526–8534.

[17] X. Zhang, M. Xu, X. Zhou, Realnet: A feature selection network with realistic synthetic anomaly for anomaly detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 16699–16708.