F1LLM: A Large Language Model Intelligent Agent for Formula 1 Telemetry Analysis and Support

Davide Benotto
s332150@studenti.polito.it
Politecnico di Torino
Turin, Italy

Paolo Riotino
s332530@studenti.polito.it
Politecnico di Torino
Turin, Italy

Manuele Mustari
S319694@studenti.polito.it
Politecnico di Torino
Turin, Italy

Abstract

This paper introduces a novel approach leveraging a large language model (LLM) to support telemetrists in post-race failure analysis. The proposed system integrates the following machine learning models: an autoencoder for anomaly detection and a classifier for failure classification, with a large language model (LLM) that interprets telemetry data and generates detailed explanations. Experiments were conducted on historical telemetry datasets from 2019 to 2024, leveraging real-world data to evaluate the system's performance. The results demonstrate the system's ability to detect anomalies, classify failures, and provide actionable insights, significantly streamlining the failure analysis process. These findings highlight the potential of the system to enhance telemetrist workflows and improve decision-making in dynamic and competitive motorsport scenarios.

ACM Reference Format:
Davide Benotto, Paolo Riotino, and Manuele Mustari. 2025. F1LLM: A Large Language Model Intelligent Agent for Formula 1 Telemetry Analysis and Support. In *Proceedings of Applied Data Science Project*. Politecnico Di Torino, Turin, Italy, 7 pages.

1 Introduction

Formula 1 racing is a highly data-intensive field, where teams rely on telemetry datasets to assess vehicle performance and identify potential anomalies or failures. Issues in critical components such as engines, gearboxes, or aerodynamic systems can result in significant performance losses or race retirements. Analyzing these failures after a race is a complex and time-consuming process that demands expertise and precision to pinpoint root causes and implement effective solutions.

To address these challenges, this work presents an intelligent system that combines machine learning models with a large language model (LLM) acting as an agent to support telemetrists in post-race failure analysis. The LLM dynamically invokes the analytical modules - anomaly detection and failure classification - based on the context provided in a prompt. These machine learning-based modules process telemetry data to detect irregularities and estimate the likelihood of specific failure types. The LLM then interprets these outputs to generate comprehensive explanations.

This paper is structured to provide a detailed overview of the proposed system, beginning with the integration of an LSTM autoencoder for anomaly detection, a CNN-LSTM-FC classifier for failure classification, and an LLM agent for interpreting telemetry data. The methodologies underlying these components are explained in depth, followed by an evaluation using real-world telemetry datasets. Through this structure, the paper demonstrates how the system supports telemetrists by automating failure analysis, identifying, explaining, and contextualizing failures, ultimately providing a comprehensive solution to the challenges of telemetry diagnostics in the dynamic context of racing.

2 Related Work

2.1 Autoencoder for Anomaly Detection

The field of anomaly detection has advanced significantly with machine learning, enabling precise identification of irregular patterns in complex datasets. Zamanzadeh Darban et al. [1] reviewed state-of-the-art deep learning methods for time series anomaly detection, while Zhang et al. [2] demonstrated the effectiveness of Variational Autoencoders (VAEs) for multivariate data. Moreover, von Schleinitz et al. [3] tailored autoencoders to motorsport, addressing multivariate anomaly detection under varying track and weather conditions, highlighting the adaptability of these models.

2.2 Classification Models for Failure Prediction

Failure classification is critical in motorsports. By detecting patterns in telemetry data, it supports targeted interventions and prevents recurring problems. Iqbal et al. [4] introduced deep ensemble models for anomaly detection, effectively capturing spatial and temporal dependencies in complex time series data. Similarly, Cai et al. [5] developed a model for driving style classification, whose methodologies show potential for failure classification and telemetry analysis in motorsports.

2.3 Large Language Models as Intelligent Agents

Large Language Models (LLMs) excel at processing prompts and contextual inputs, breaking down complex tasks, and dynamically selecting actions, as noted by Huang et al. [6]. This makes them well-suited for modular decision-making, automating workflows, and generating actionable insights.

3 Methodology

3.1 Problem Statement

In Formula 1, race telemetry data consists of multivariate time-series signals collected from sensors monitoring the car, track, and weather conditions. This project introduces a system in which a large language model (LLM) interprets data generated by machine learning models, responsible for anomaly detection and failure classification, delivering precise and actionable insights to support telemetrists in failure analysis and diagnostic tasks.

The problem can be divided into three stages:

3.1.1 Anomaly Detection. Let $X = \{x_1, x_2, ..., x_T\}$ represent the telemetry data, where $x_t \in \mathbb{R}^n$ is the feature vector at time t. The autoencoder is trained to reconstruct the input data X by learning a compress representation $\hat{X}$. The objective is to minimize the mean absolute reconstruction error, given by:

$$\mathcal{L}_{recon} = \frac{1}{T}\sum_{t=1}^{T} ||x_t - \hat{x}_t||$$

Anomalies are detected by evaluating the reconstruction error for each time step t. If $||x_t - \hat{x}_t||$ exceeds a predefined threshold, the corresponding x_t is classified as an anomaly.

3.1.2 Failure Classification. Given a sequence of detected anomalies $A = \{a_1, a_2, ..., a_m\}$, the classifier predicts a failure probability $P = \{p_1, p_2, ..., p_k\}$, where p_i corresponds to the likelihood of failure type i. The training objective is to minimize the cross-entropy loss:

$$\mathcal{L}_{class} = -\sum_{i=1}^{k} y_i log(p_i)$$

where y_i is the ground truth label for failure type i. This loss function measures the difference between the predicted probability distribution P and the true label y , guiding the model to assign higher probabilities to the correct failure types.

3.1.3 LLM Analysis. The LLM processes telemetry data and a domain-specific prompt, acting as an intelligent agent that dynamically decides whether to invoke the anomaly detection or failure classification module based on the prompt. The decision-making process maximizes:

$$P(action|data, prompt) = \prod_{t=1}^{N} P(a_t|a_1, \ldots, a_{t-1}, data, prompt),$$

where $P(a_t|a_1, \ldots, a_{t-1}, data, prompt)$ is the probability of selecting the next action given data and prompt while *action* represents the module choice. This enables the system to adaptively respond to analytical tasks described in the prompt.

3.2 Diagram of the Method

3.2.1 Data Preprocessing and Normalization. The process starts with the extraction of raw telemetry data using the FastF1 library, which provides access to a wide range of multivariate time-series signals such as engine RPM, speed, and throttle. This project focuses specifically on telemetry data from Formula 1 races held between 2019 and 2024.

The preprocessing step involves:

- **Filtering irrelevant features**: Only parameters critical to vehicle performance and failure prediction are retained, ensuring the focus remains on meaningful telemetry signals.
- **Normalization**: Feature values are scaled to a uniform range to improve training stability.

The preprocessed telemetry data is divided into two categories:

- **Training and validation data**: Includes telemetry from races held between 2019 and 2023, used to train and refine the system's models. All data points flagged as failures are removed, creating a "clean" dataset that represents typical race patterns. This ensures the autoencoder learns only normal behaviors.
- **Testing data**: Consists of telemetry from the 2024 season, including anomalies, and is exclusively used to evaluate the system's generalization and performance.

3.2.2 Anomaly Detection using LSTM Autoencoder. The preprocessed data is fed into an LSTM autoencoder, a neural network designed to identify patterns over time. This model consists of an encoder, which learns a compressed representation of the input data, and a decoder, which attempts to reconstruct the original input from this compressed representation. The structure of the LSTM autoencoder along with the its workflow is illustrated in Figure 1.

Anomalies are flagged when the reconstruction error $\mathcal{E}(t)$ exceeds a predefined threshold:

$$\mathcal{E}(t) = ||x_t - \hat{x}_t||$$

By training the autoencoder on clean data (i.e., no failures included), the model develops a clear understanding of "normal" telemetry patterns. When tested on new race data (2024), it highlights anomalies moments where the telemetry deviates significantly from learned patterns.

3.2.3 Failure Classification using CNN-LSTM-FC. Anomalies identified by the autoencoder serve as input for the failure classification model, which assigns a probability to each potential failure type. This stage leverages a hybrid CNN-LSTM architecture, followed by a fully connected (FC) layer:

- **CNN**: Identifies spatial correlations among features within a single time frame, enhancing the understanding of multi-dimensional telemetry data and extracting high-level feature representations.
- **LSTM**: Captures temporal dependencies in the anomaly sequence, ensuring the model considers the time-evolving nature of telemetry signals and learns patterns over time.
- **Fully Connected Layer**: Maps the features learned by the CNN-LSTM architecture to a probability distribution over the possible failure types, enabling the model to produce interpretable outputs for classification.

Testing is performed using anomalies from 2024, validating the classifier's ability to assign accurate failure probabilities to unseen scenarios.

3.2.4 Insight Generation using LLM. The final stage of the system employs a pre-trained Large Language Model (LLM) to act as an intelligent agent, assisting telemetrists by transforming the outputs from the machine learning modules into actionable insights. This process operates through the following inputs and outputs:

- **Input**: The LLM receives a prompt specifying the type of analysis to perform, along with data related to the race being analyzed. Acting as an agent, the LLM uses the prompt to determine whether to invoke the anomaly detection module (providing detected anomalies) or the failure classification

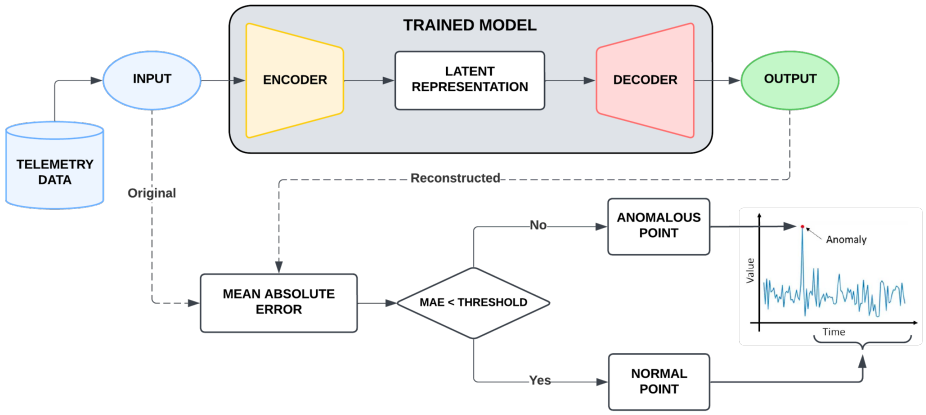

Figure 1: Workflow of the LSTM Autoencoder for Anomaly Detection

module (providing failure probabilities) and processes the results accordingly.

- **Output**: The LLM generates a detailed report tailored to the analysis performed. The report summarizes detected anomalies or interprets failure probabilities, providing clear insights.

By dynamically selecting and invoking the appropriate modules and contextualizing their outputs, the LLM functions as an AI agent, bridging the gap between raw data and decision-making.

3.3 Complete List of Technical Tools and Configurations

(1) **Models and Architectures**:
- *LSTM Autoencoder*: Trained to identify anomalies in telemetry data by learning "normal" patterns.
- *CNN-LSTM-FC Classifier*: Combines spatial feature extraction (CNN), temporal dependencies (LSTM), and a fully connected layer (FC) for failure classification through probability estimation.
- *LLM (Large Language Model)*: agent that invokes the appropriate module based on prompt and given data, generating contextualized explanations.

(2) **Datasets and Sources**:
- *Telemetry Data*: Extracted from the FastF1 library, focusing on races between 2019 and 2024.

(3) **Frameworks and Libraries**:
- *PyTorch*: For developing and training the models.
- *Hugging Face Transformers*: For leveraging pre-trained LLMs.
- *Gradio*: To provide an interactive interface for real-time input and output handling with the LLM.
- *NumPy, Pandas, Matplotlib, and other libraries*: For data preprocessing, analysis, and visualization.
- *Kaggle*: Used to run the LLM, as it requires computational resources beyond those available on a standard local machine.

(4) **Configurations**:
- *Development Environment*: Python 3.9 with PyCharm and VS Code.

(5) **Evaluation and Metrics**:
- *Cross-Entropy Loss*: Used for training the classifier.
- *Mean Absolute Error*: For anomaly detection using the autoencoder.
- *Accuracy and Loss*: To evaluate the performance of the models on classification tasks.

4 Experiment

4.1 FastF1 Dataset

The dataset used in this study was extracted from **FastF1** [7], an open-source library designed for Formula 1 data analysis. The telemetry data, provided in the form of **tabular data**, includes high-resolution, multivariate signals essential for anomaly detection and failure classification. While not official, the library compiles information from publicly available sources such as Formula 1 broadcasts and includes historical race data dating back to the 1950s. The project focused on the telemetry data available exclusively for the **2019** to **2024** seasons.

FastF1 organizes telemetry data as extended Pandas DataFrames, enriched with custom functions that simplify handling and analysis. It also integrates seamlessly with Matplotlib for visualization and implements caching for API requests, optimizing script execution times.

The initial dataset extracted from this library consisted of approximately **10 million telemetry records**, encompassing all drivers and races from 2019 to 2024. Due to the different granularities of the raw data, covering telemetry signals, weather information, and lap details, a **preprocessing pipeline** was implemented to merge and align these data sources effectively. Additional cleaning steps included the **removal of NaN values** and **duplicate records**, which significantly improved the dataset's quality and consistency. Moreover, laps involving **pit stops** were carefully managed during preprocessing to ensure the data's relevance for anomaly detection. For the training set, all laps in which a driver performed a pit stop were removed to allow the autoencoder to learn normal driving patterns without interruptions or irregular events. Additionally, the

column indicating pit stops was dropped from the training data to prevent the model from being biased by this feature. In contrast, the testing set retained laps with pit stops, to evaluate the system's performance under real-world conditions.

It is important to note that some inconsistencies in the dataset had to be corrected during preprocessing, especially for failures that caused a driver to retire from a race, since some of these were mislabeled or incomplete.

Building on the preprocessing steps described earlier, the dataset was reduced from an initial set of 61 features to **27 key features** through an iterative selection process. This reduction aimed to retain the most relevant variables while eliminating less informative or redundant features. Key features such as RPM, Speed, Throttle or Weather Conditions were retained for their direct relevance to the system's objectives. In contrast, less relevant or highly correlated variables, such as Sector Time, FreshTyre, Stint or specific track Speed measurements (SpeedFL, SpeedST, SpeedI1, SpeedI2) were excluded.

To ensure numerical stability and improve model performance, all numerical and temporal features were normalized using the **Min-Max Scaler**, while categorical variables were mapped to numerical values where necessary. The processed dataset was stored in **NPZ** format instead of CSV files to optimize disk space usage and facilitate faster data loading and management during experimentation.

4.2 Experiment Configuration

4.2.1 LSTM Autoencoder. The LSTM Autoencoder was developed to **detect anomalies** in telemetry data by learning to **reconstruct normal patterns**. The underlying assumption is that the model, trained solely on **clean data without anomalies**, will struggle to reconstruct abnormal patterns, resulting in higher reconstruction errors. To identify anomalies, a **threshold** was determined using the 99th percentile of reconstruction errors in the validation set during the final training epoch, with errors exceeding this threshold flagged as anomalies.

The final model architecture includes an **encoder** that compresses the input sequences into a representation with 64 hidden units, followed by a **latent layer** with 32 hidden units that further reduces dimensionality. A **decoder** with 64 hidden units reconstructs the sequences, and an **output layer** restores the original feature dimensions. The input comprises **sliding windows** of telemetry data, each consisting of sequences with a fixed length of 20 time steps. These windows overlap by advancing one record at a time, ensuring that all data points are included while effectively capturing temporal dependencies within short intervals.

While telemetry data is sequential, **k-fold cross-validation** was applied during training because the data's organization does not strictly adhere to the structure of typical time series, allowing effective validation of the model's performance. Telemetry data from the 2019 to 2023 seasons, excluding laps with pit stops, was used for training. The **AdamW optimizer** minimized the reconstruction loss, with hyperparameters such as **learning rate** and **batch size** tuned experimentally.

The **output** is the reconstructed telemetry data, with reconstruction errors calculated as the mean absolute error between input and output. The entire workflow of the Autoencoder, along with its structure, is shown in Figure 1.

4.2.2 CNN-LSTM-FC Failure Classifier. Failures encountered over the analyzed seasons were initially categorized into 45 anomaly types based on telemetry data and historical records. To streamline classification and address class imbalance, these anomalies were grouped into **8 categories**: '*Engine*', '*Power Unit*', '*Suspension and Drive*', '*Braking System*', '*Transmission and Gearbox*', '*Cooling System*', '*Aerodynamics and Tyres*', and '*Others*'. This grouping, guided by shared telemetry patterns and failure mechanisms, ensured a balanced dataset, as shown in Figure 2, and improved classifier performance while maintaining alignment with real-world failure contexts.

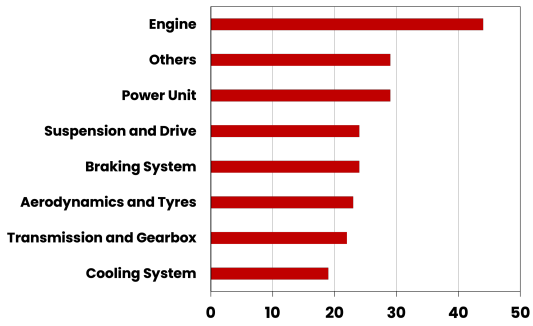

Figure 2: Failure category distribution across the analyzed dataset.

The failure classification model employs a CNN-LSTM-FC (Convolutional Neural Network + Long Short-Term Memory + Fully Connected) architecture designed to process sequences of telemetry data and output probabilities for each failure category. Figure 3 show the architecture. This architecture was chosen for its ability to capture both spatial correlations across features and temporal dependencies within sequences, making it particularly suited for the nature of telemetry data.

The model begins with a **convolutional layer**, which receives telemetry sequences as input. For each sequence, the convolutional layer processes overlapping windows of the input data (with a kernel size of 3) to extract spatial patterns between features, such as correlations between speed, throttle position, and tire temperatures. These telemetry sequences are extracted starting from the first moment the anomaly is detected until the driver's retirement, ensuring the model captures the full progression of the issue. This is followed by **max pooling**, which reduces the spatial dimensionality and highlights the most prominent spatial features. The second convolutional layer refines these features, further enhancing their representation before passing the output to the LSTM module.

The **LSTM layer** receives a sequence of temporally enriched feature vectors as input, structured as a 3D array of dimensions (`batch_size, sequence_length, features`). The LSTM processes these sequential features to capture evolving telemetry patterns over time. For instance, a sudden increase in speed combined with frequent DRS activation and high engine RPM might indicate potential stress on the engine or aerodynamic instability. The LSTM

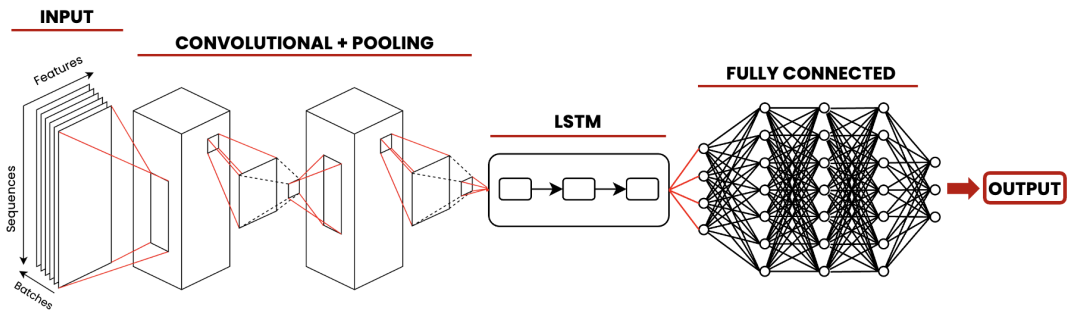

Figure 3: Failure Classifier Architecture

outputs a hidden state for each timestep, with the last hidden state summarizing the sequence's temporal dynamics.

Finally, the hidden state produced by the LSTM, which summarizes the temporal dynamics of the input sequence, is passed to the **fully connected (FC) layer**. The input to the FC layer is a 2D array of dimensions (`batch_size, hidden_size`), where hidden_size corresponds to the number of features in the last hidden state of the LSTM. The FC layer applies linear transformations and maps this input to the eight failure categories, producing an output of dimensions (`batch_size, 8`). This output represents the logits, which are then passed through a **softmax activation function** to convert them into a probability distribution. Each value in the resulting distribution corresponds to the likelihood of the input sequence being associated with a specific failure category.

The output from the classifier consists of **probability distributions** over the eight failure categories for each sequence, enabling detailed failure diagnostics for every segment of telemetry data. To determine the final anomaly classification for an entire detected anomalous event, a **majority voting** approach is applied across the sequence-level predictions. This ensures a robust and consistent categorization of the failure, allowing telemetrists to identify the most likely failure causes with high precision.

4.2.3 Large Language Model Agent. The **Large Language Model (LLM)** used in this system is **LLaMA 3.2-1B**, accessed via the Hugging Face Transformers library. With 1 billion parameters, LLaMA 3.2-1B provides an effective balance between performance and inference speed, making it well-suited for processing telemetry data and generating detailed textual insights. The use of LLaMA 3.2-1B was driven by its ability to handle complex reasoning tasks effectively within Kaggle's computational limits. To facilitate its implementation and interaction, the system leverages **Gradio**, which provides a user-friendly interface for real-time input and output handling.

The LLM functions as an **intelligent agent**, dynamically interacting with the machine learning modules and telemetry data. Its input consists of a structured prompt specifying the task (e.g., 'anomaly detection' or 'failure classification') and relevant telemetry data that contextualizes the request. Based on the keywords and instructions in the prompt, the LLM determines which analytical module to invoke. For example, a request such as "Let's operate anomaly detection on the given data" triggers the LLM to call the anomaly detection module, while a query like "Can you operate failure classification on the following data" invokes the failure classification module.

After invoking the selected module, the **LLM processes the module's output**. For the anomaly detection module, this output consists of reconstruction errors flagged as anomalies, while for the failure classification module, the output is a probability distribution over the eight predefined failure categories. The LLM then analyzes these results in the context of the initial prompt, leveraging its natural language processing capabilities to synthesize clear and actionable insights.

The **final output** generated by the LLM is a natural language report tailored to the telemetrist's requirements. This report includes a summary of detected anomalies or failure probabilities, an interpretation of the results, and actionable recommendations to address potential issues or optimize performance.

This combination of LLaMA 3.2-1B, Hugging Face, and Gradio ensures the system is both powerful and accessible for its intended applications.

4.3 Evaluation

The evaluation of the proposed system was conducted on telemetry data from the 2024 season, encompassing multiple races and diverse failure types. The process involved manual verification of detected anomalies, where anomalies flagged by the LSTM autoencoder were reviewed against actual failures recorded for each race.

The autoencoder achieved a reconstruction loss of 0.4, demonstrating its capability to effectively identify telemetry irregularities. These detected anomalies were then passed to the failure classification module for further analysis.

The CNN-LSTM-FC classification model accurately identified the correct failure category in 50% of the cases, where the top-predicted category matched the actual anomaly. When considering the top three predictions, the performance increased significantly, with 75% of cases including the correct failure category among the top-ranked predictions. This highlights the model's ability to capture relevant patterns and provide useful contextual insights, even if the highest probability category does not always align with the actual anomaly. However, in 25% of the cases, the predictions deviated significantly from the actual failure, often misclassifying anomalies into unrelated categories, such as "Others" or overlapping failure types, suggesting room for improvement in fine-tuning the model

Event	Driver	Actual	Predicted (%)
Hungarian GP	GAS	Suspension and Drive	Suspension and Drive (78.00%)
Las Vegas GP	GAS	Engine	Engine (81.05%)
Saudi Arabian GP	GAS	Transmission and Gearbox	Transmission and Gearbox (81.98%)
Mexico City GP	ALO	Braking System	Transmission and Gearbox (78.87%)
Canadian GP	LEC	Engine	Engine (91.58%)
Australian GP	VER	Braking System	Others (81.08%)
Italian GP	TSU	Cooling System	Transmission and Gearbox (28.75%)
Las Vegas GP	ALB	Cooling System	Transmission and Gearbox (51.73%)
Singapore GP	ALB	Power Unit	Cooling System (52.46%)
Japanese GP	ZHU	Transmission and Gearbox	Transmission and Gearbox (45.51%)
Australian GP	HAM	Engine	Others (65.86%)
British GP	RUS	Cooling System	Cooling System (90.09%)

Table 1: Failure Prediction Performance Across Events (2024)

and refining its decision-making process. The results are further shown in Table 1.

The hyperparameters for both the LSTM Autoencoder and the CNN-LSTM-FC Classifier are shown in Table 2.

Parameter	Autoencoder	Classifier
Epochs per split	10	25
K-Fold Splits	5	5
Learning Rate	0.0001	0.0001
Batch Size	128	64
Sequence Length	20	100

Table 2: Training parameters for the Autoencoder and Classifier.

LLaMA 3.2 1B was used to interpret model outputs, providing explanations and actionable insights. Although its smaller size limited its ability to handle complex scenarios, it was the most suitable option within the computational constraints. Future work could explore more advanced models on adequate resources.

Overall, the system demonstrated its capability to effectively identify anomalies and classify them into meaningful categories, showcasing the strength of integrating machine learning models with LLMs for failure analysis. The anomaly detection module performed well in isolating irregular telemetry patterns, and the classification model achieved reasonable accuracy in matching detected anomalies to their respective failure categories.

However, certain challenges remain, primarily stemming from the limitations of the available data and computational constraints. The dataset, while comprehensive in scope, included inaccuracies and inconsistencies that required extensive preprocessing, limiting the overall quality of the input data. Additionally, the failure classification model faced challenges due to the small number of labeled failures available for training, which restricted its ability to generalize effectively to unseen scenarios. The computational limitations of the LLM further impacted performance, as the selected LLaMA 3.2 1B, while efficient, lacked the capacity to fully contextualize complex telemetry scenarios. These factors suggest that future improvements could be achieved through access to higher-quality data, a larger and more diverse failure dataset, and more powerful computational resources to enable the use of advanced LLM architectures.

5 Conclusion

The objective of this work was to develop a comprehensive system integrating machine learning models and a Large Language Model (LLM) to analyze telemetry data and diagnose failures in Formula 1 vehicles. By combining an LSTM autoencoder for anomaly detection, a CNN-LSTM-FC model for failure classification, and the LLM for contextualizing outputs, the system aimed to support telemetrists with actionable insights and precise diagnostics.

The proposed methodology successfully addressed this objective by leveraging the strengths of machine learning and LLMs. The autoencoder efficiently detected anomalies in telemetry data, while the classifier demonstrated the ability to associate detected anomalies with specific failure categories. The integration of the LLM further enriched the analysis by providing detailed textual explanations, though its limited capacity highlighted the computational trade-offs in deploying such models.

Evaluation revealed that the system is effective in identifying and classifying anomalies, achieving accurate predictions in most cases. However, challenges remain, particularly with edge cases and failures that are underrepresented in the dataset. The limited granularity and quality of certain telemetry features, as well as the computational constraints of the LLM, contributed to these limitations.

From this study, valuable lessons have emerged. The importance of high-quality, well-labeled datasets is evident, as is the necessity for robust feature engineering to handle imbalanced classes. Additionally, the computational resources required for LLMs underscore the trade-offs between model complexity and practicality in resource-constrained environments.

Future work will focus on implementing additional modules, such as lap prediction and telemetry comparison, to extend the system's functionality. Enhancing the dataset with more granular and effective telemetry features, particularly for components like brakes, will be a priority. Lastly, integrating a more powerful LLM will be explored to improve contextual understanding and the depth of insights provided by the system. These advancements aim to further refine and enhance the system's capabilities in real-world scenarios.

The repository is availble here: **2024-P3-F1LLM**

References

[1] Zahra Zamanzadeh Darban, Geoffrey I. Webb, Shirui Pan, Charu Aggarwal, and Mahsa Salehi. Deep learning for time series anomaly detection: A survey. *ACM Computing Surveys*, 57(1):1–42, October 2024.

[2] Chunkai Zhang, Shaocong Li, Hongye Zhang, and Yingyang Chen. Velc: A new variational autoencoder based model for time series anomaly detection, 2020.

[3] Julian von Schleinitz, Michael Graf, Wolfgang Trutschnig, and Andreas Schröder. Vasp: An autoencoder-based approach for multivariate anomaly detection and robust time series prediction with application in motorsport. *Engineering Applications of Artificial Intelligence*, 104:104354, 2021.

[4] Alsubaei FS Alzahrani A Iqbal A, Amin R. Anomaly detection in multivariate time series data using deep ensemble models, 2024.

[5] Yingfeng Cai, Ruidong Zhao, Hai Wang, Long Chen, Yubo Lian, and Yilin Zhong. Cnn-lstm driving style classification model based on driver operation time series data. *IEEE Access*, 11:16203–16212, 2023.

[6] Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. Understanding the planning of llm agents: A survey, 2024.

[7] Philipp Schaefer. Fastf1 library documentation, 2018.