ProActRail: Proactive Maintenance Prediction for Railway Air Production Units

|  |  |
| --- | --- |
| 1st Somayya Elmoghazy  *Electrical and Computer Engineering*  *Ontario Tech University*  Oshawa, Canada  somayya.elmoghazy@ontariotechu.net | 2nd Ammar Elmoghazy  *Electrical and Computer Engineering*  *Ontario Tech University)*  Oshawa, Canada  ammar.elmoghazy@ontariotechu.net |

*Abstract*—Predictive Maintenance (PdM) is an emerging maintenance technique for different industries that uses data analysis along with machine learning to facilitate the maintenance tasks of different equipment and predict failures before they happen. It aims to reduce the cost of scheduled maintenance that often include unnecessary repairing or replacement. It also aims to improve the lifespan of the equipment by maintaining them before they reach a complete malfunction. Different machine learning techniques have been utilized for this task especially the ones that use deep learning. Autoencoders specifically are being used since they can detect anomalies from largescale unlabeled data. We use a Sparse Autoencoder network for failure prediction of an Air Production Unit (APU) of a Metro in Portugal. We deploy our model on the cloud and simulate a real life IoT system by creating synthetic data and feeding it to the frontend application then use the trained model for failure prediction.

*Index Terms*—Iot, Predictive Maintenance

# I. INTRODUCTION

Various strategies and techniques are employed for vehicle maintenance such as corrective, preventive, and condition-based maintenance. In corrective maintenance [1], for instance, the repairing and maintenance only happens after a

malfunction of an equipment. On the other hand, preventive maintenance involves the repairing or replacing of equipment happens because of a predetermined and scheduled routine inspections [2]. This can lead to resource waste, as functional equipment may be unnecessarily repaired to avoid unexpected breakdowns. Time is also another precious asset that can be wasted on unnecessary maintenance. For this reason, condition-based maintenance focuses on monitoring the actual state of the system to determine the necessary maintenance tasks [3].

One technique of condition-based maintenance is predictive maintenance (PdM). PdM leverages data analysis tools to evaluate both real-time and historical data from different system components [4]. This allows for the detection of irregularities and potential flaws and for the facilitation of system repairs before an actual failure occurs. Several proposals for incorporating machine learning in PdM have emerged recently. Deep learning methods, in particular, such as Neural networks, Recurrent Neural Networks, Convolutional Neural networks, are being utilized for failure predictions. These methods help estimate the likelihood of equipment failure by automatically processing the historical data of the system.

Autoencoders are also being used as deep learning techniques for PdM. They are unsupervised learning methods that can automatically detect failures, anomalies, in unlabeled data. They do that by compressing data into lower-dimension representation then reconstructing it to capture the important patterns and so identify anomalies or irregularities in the future data. This allows for PdM to be applied on large-scale datasets from different sources without the need to waste time and effort labeling all the data entries. There are different types of autoencoders such as Sparse Autoencoders (SAE) and Variational Autoencoders (VAE). The authors in [3] have conducted a comparison of both techniques for anomaly detection and have shown that SAE are superior in this problem. Hence, we focus on implementing an SAE model for failure prediction on our dataset.

We propose a data-driven method for failure prediction using the MetroPT dataset [5]. Our solution utilizes a custom SAE model for failure prediction on the back-end side of our system. For real-time implementation, our system is composed of five parts, data generation, data ingestion, data storage, model inference and visualization. Our aim is to provide a complete IoT system for predictive maintenance where synthetic data is used to simulate real life data and the system is able to predict failures before they happen.

# II. RELATED WORK AND BACKGROUND

Predictive maintenance in transportation systems is important, as it touches millions of people who live in any big city. Faults in public transportation vehicles during regular service can lead to numerous issues, particularly when they interrupt journeys. These negative consequences impact both the operating company and the passengers. Early detection of such faults can prevent trip cancellations, service disruptions, and save money, which is highly valuable. In 2017 alone, over 170 trips were canceled due to these issues [5].

The Air Production Unit (APU), is one of the important systems located on roofs of train vehicles; it is responsible for multiple operational functionalities. For example the secondary suspension, which ensures the train is leveled probably regardless of the train load. The APU is in high demand throughout the train operation, and due to there being no usual backups for this unit any damage that occurs would require the train to go out of service. The type of failures that occur can’t be detected using basic methods like setting a threshold value, which explains the need to use machine learning models to analyze the patterns in the data.

Throughout the past years many research have been done in the use of different techniques to perform predictive maintenance tasks specifically in the transportation industry. These techniques can be divided into two sections: condition-based like [6] or data driven like [7]. The choice of which technique to be used depends completely on the type of data available and business objectives. For the MetroPT benchmark the dataset contains over 10 million data points and contains two types of failures that occurred over three different time periods. A technical report was published to provide the ground truth of the failures, which includes:

* Failure time
* Affected components
* Failure type

Two papers have used the benchmark so far, [8] took a rule-based approach, they defined a safe mode or a failure mode based on the peak reading values from the historical sensor readings. Although this approach could be simpler, the main goal of predicting the failure before it occurs is not met. The second paper [3] used two types of auto-encoders in their approach and added a low pass filter to the model output which is important to lower the number of false positives in the system. As it won’t be good to have frequent train stops based on the model prediction, but then it appears that the APU is in a good condition. In [3] they used data from March to July 2020, which is not publicly available as the dataset used in this project is between January and June 2022. However, the approach is still valid as the dataset is the same structure, the only main difference is in the number of failures in the dataset. The data is a combination of analog and digital sensors installed on an APU system is used to monitor real-time performance changes. Analog sensors record real values, while digital sensors capture binary values. The APU’s sensor layout is shown in figure 3 and the sensors include: Analog sensors:

* TP2 - Compressor Pressure
* TP3 - Pressure measurement in the pneumatic panel
* H1 - Activated when the pressure is above the normal values
* DV Pressure - This is the value of the pressure drop once the towers discharge water (if zero, the compressor is under load)
* Motor-Current - Current measurement of one phase in the three-phase motor
* Oil Temperature - the value of oil temperature inside the compressor

In this project, it appeared that the analogue values had more impact on the failure predictions; therefore, all the analog signals are used. In addition to, transforming the digital signals like the COMP signal (when it is turned off, means the compressor is not working). The main challenges in using machine learning in this type of application is the data streaming and data pre-processing. In this paper, we applied similar feature engineering and data post-processing types as [3]. However, it was more complicated as we are proposing a fully hosted application that could be used by the maintenance team for monitoring the system. The paper did conclude that the Sparse autoencoder achieved better results in the analog features 70% precision, 45% F1 score, and 33% recall. Having a trained machine learning model is one component of a predictive maintenance task. It is important to have it perform inference on the newly generated raw data and visualize the outputs, so that the operating company would be able to take the required actions in a timely manner.

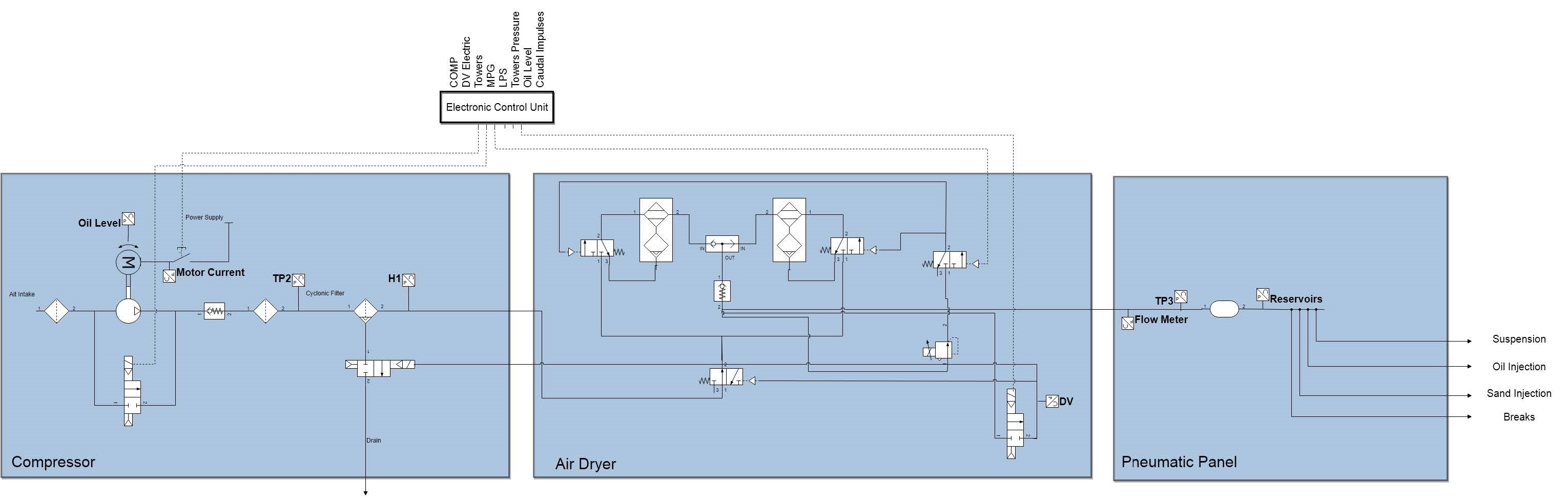
Generally, there have been more data-driven approaches taken in other studies, which includes traditional machine learning techniques like: Linear Regression [9], Support Vector Machine [10]. More approaches considered even using a combination of approaches like [11], which used ARIMA combined with SVM model. Although machine learning models would work perfectly for well labelled and small amount of data, In case of sensory data which is unlabelled and big amounts of data containing noise and a lot of pre-processing. It is encouraged to use a deep learning approach, as classical machine learning approaches would easily over-fit over this type of task [12]. The following review [13] explores the use of various deep learning techniques in the predictive maintenance task. One of the Deep learning approaches are the autoencoders, which is a form of neural networks usually used as a self-supervised as it performs unsupervised feature extraction. There is various variants of auto-encoders developed on the same concepts one of which is used in this project.

# III. PROPOSED SOLUTION - PROACTRAIL

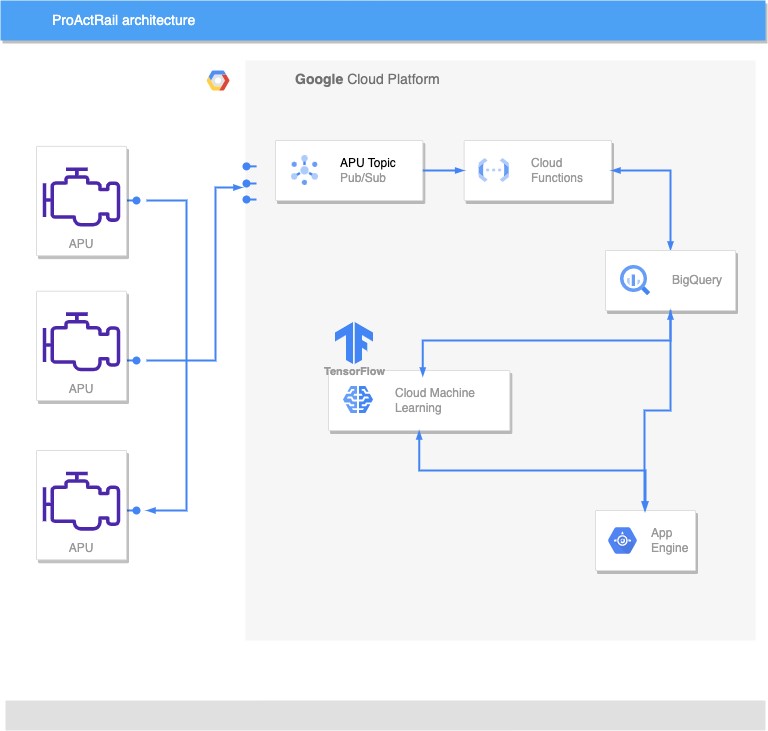
The growing demand for real-time data processing and analytics in the Industrial IoT applications has necessitated the development of robust and efficient systems that can manage the ingestion and processing of vast amounts of unstructured data. One such innovative system, leveraging the power of Google Cloud services, has been successfully implemented to facilitate seamless data ingestion, processing, storage, and visualisation. This section dives into the architecture, benefits, and potential applications of this cutting-edge system. The solution is divided as follows:

* Data Generation
* Data Ingestion
* Data Storage
* Model Inference
* Visualization

The implemented system is based on a serverless architecture that harnesses the capabilities of Google Cloud Platform (GCP) services, namely Google Cloud Pub/Sub, Cloud Functions, App Engine, BigQuery, and Vertex AI. The process begins with the ingestion of simulated data through Google



## Fig. 1. APU sensors diagram



## Fig. 2. APU sensors diagram

Cloud Pub/Sub, a messaging service that enables real-time *A. Synthetic data*

communication between different services and applications. The first step of the implemented system is the generation of synthetic data to simulate real-time data from train air production units. To accomplish this, a Python script was developed, which leverages various statistical and mathematical

techniques to create realistic sensor data that closely resembles actual measurements that follows the same normal distribution. This script considers factors such as historical trends, seasonal patterns, and random fluctuations to generate a diverse and representative dataset. By using this synthetic data, the system can be tested and validated in a controlled environment, ensuring the accuracy and reliability of the anomaly detection process without relying on actual sensor data. The flexibility of the Python script allows for easy customization of data generation parameters, enabling the simulation of different scenarios and conditions that may be encountered in real-world operations, such as Oil leaks or Air leaks which is already available in the original dataset. Overall, the Python script serves as a valuable tool for refining and optimizing the anomaly detection system before its deployment in live environments.

## B. Pub/sub service

The Cloud Pub/Sub service operates on a publish-subscribe model, where data producers (publishers) send messages to topics, and data consumers (subscribers) receive the messages by subscribing to the topics. In this system, the simulated data is published to a designated topic called APU in the form of Json messages, which are then consumed by a Google Cloud Function to transform it to tabular format to be inserted as a new row to the BigQuery database.

## C. Cloud Function

Google Cloud Functions, a key component in the proposed solution, is a serverless, event-driven computing platform that allows for the execution of custom code in response to specific events, in this case the ingestion of data through Google Cloud Pub/Sub. This serverless architecture provides numerous benefits, including automatic scaling, cost efficiency, reduced operational overhead, and rapid development. The automatic scaling feature ensures that Cloud Functions can handle varying workloads without any intervention or resource provisioning (which is very costly), enabling the system to adapt seamlessly to changes in data volume. Cost efficiency is achieved through the pay-as-you-go pricing model, which charges only for the resources used during function execution, minimizing overall expenses. The elimination of server management responsibilities results in reduced operational overhead, allowing developers to focus on refining data processing logic and improving system performance. Lastly, the ease of deployment and integration with other Google Cloud services streamlines the development process, accelerating the implementation of new features and enhancements.

## D. Data Storage

BigQuery, is the back-bone of our solution, is a fully managed, serverless data warehouse service that enables superfast SQL querying and analysis of large datasets. This powerful service offers numerous benefits, such as scalability, cost efficiency, real-time analytics, and seamless integration with other Google Cloud services. The scalability feature of BigQuery ensures that it can handle massive amounts of data, automatically adjusting to storage and processing demands without the need for manual intervention or resource provisioning. Cost efficiency is achieved through the pay-as-you-go pricing model, which charges only for data storage and query execution, minimizing overall costs. BigQuery’s real-time analytics capabilities allow for the ingestion and processing of data as it is generated, ensuring timely insights and decision-making. Furthermore, the ability to easily perform the required machine learning tasks, like pre-processing and inference using a deployed machine learning model on Vertex AI endpoints.

## E. Sparse Autoencoder

Sparse autoencoders (SAE) are a variation of traditional autoencoders that are designed to learn sparse features by incorporating a sparse penalty term in the learning process. This is done by adding a regularization component to the cost function to penalize the weights. In SAEs, a large number of neurons are included in the first hidden layers. This is different from other types of autoencoders where the number of nodes in their hidden layers is less than the input layer. From that large number of neurons, a select few of these neurons are allowed to be activated for encoding and decoding. It is important to note that the activation values of each neuron are data dependent since they change based on the input data during the training process. These values correspond to the network weights, which are typically regularized.

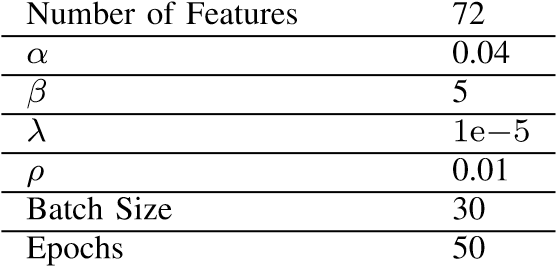
We adapt the SAE model developed for the early MetroPT dataset in [5] to our newer dataset. It is important to note that their model only incorporated two types of analog signals from the analog sensors, whereas we incorporate all eight of the analog signals. It is unclear why they only use two as the old dataset is no longer available. It could be that the older dataset did not include all the sensors that are now available. In general, the architecture of our model is slightly different from the architecture present in the paper as it is fine-tuned to the newer dataset. Table I shows the architecture of our SAE model. We follow the same feature selection process as in [3]. The data from each analog sensor is segmented according the run and idle, *Trun* and *Tidle*, times of the Compressor, which is represented by the digital signal COMP in the dataset. Then, both segments are further segmented into two, for *Trun*, and five, for *Tidle*, equal time intervals. Finally, the mean of each segment is calculated and added to the feature vector. The resulting feature vector is composed of seven numbers for each signal in addition to *Trun* and *Tidle*. Since there are eight analog signals in the dataset, the total length of the vector is 72.

## F. Real-time inference

In the project, a key milestone was the deployment of a model on Vertex AI Endpoint, a powerful and flexible service provided by Google Cloud Platform. Vertex AI Endpoint offers numerous benefits for the management, serving, and scaling of machine learning models. One of its primary advantages is the

TABLE I SAE PARAMETERS

|  |  |
| --- | --- |
| Parameter | Value |
|  |  |
| Neurons (2nd hidden layer) | 8 |
| Neurons (Bottleneck layer) | 8 |



ease of deployment, which simplifies the process of integrating the model into the existing system. Vertex AI Endpoint also provides built-in support for versioning and updating models, ensuring seamless transitions between model iterations without disrupting the application. Furthermore, there is two options to choose from either batch processing or real-time processing, in this solution real-time processing is used to be able to analyze the streamed data and visualize the model output in near real-time. This feature ensures that the model can effectively handle the demands of the anomaly detection system, regardless of data volume or processing requirements. Overall, the deployment of the model on Vertex AI Endpoint enhances the efficiency, reliability, and maintainability of the predictive maintenance system, while minimizing the operational overhead associated with traditional machine learning model management.

## G. Data Visualization

A Streamlit application is designed to monitor and predict the APU failures. The application is built with several components and functions to simulate, fetch, process, and visualize the data, as well as to predict potential failures. For the sake of demonstration, the data simulation is add as button at the top of the app page, in a real-life scenario the data would be directly streamed into the application from the BigQuery database.

Here’s a description of the main components and functions in the code:

* Function: get\_data\_from\_bigquery - Fetches and processes historical data from BigQuery, returning it as a pandas DataFrame.
* Function: get\_filtered\_data\_from\_bigquery - Filters historical data based on a time range, returning the filtered data as a pandas DataFrame.
* Function: preprocess\_data - Scales and transforms raw data for input to the prediction model.
* Function: predict\_failure - Feeds preprocessed data into a pre-trained model to predict equipment failure, returning a prediction array.
* Function: display\_prediction - Displays success or failure messages based on prediction results, using Streamlit session state to avoid duplicates.
* Function: send\_to\_pubsub - Sends a data row to a Google Cloud Pub/ Sub-topic as a JSON message.
* Function: simulate\_oil\_leak\_data\_to\_pubsub - Main function simulating oil leak data and sending it to Pub/Sub. Initializes session states for buttons, creates empty chart placeholders, filters historical data, and loops through filtered data. During the loop, it sends data to Pub/Sub, fetches, and visualizes updated data, preprocesses data, and predicts equipment failure. The loop is terminated when the” Stop Oil Leak Data” button is pressed.

The simulation can be stopped at any time by pressing the” Stop Oil Leak Data” button. The application will stop sending data to Pub/Sub and cease updating the charts and predictions.

Overall, this Streamlit application is a useful tool for visualizing and predicting equipment failures using historical data and a machine learning model. It can help operators and maintenance personnel identify potential issues and take appropriate action to prevent equipment failure.

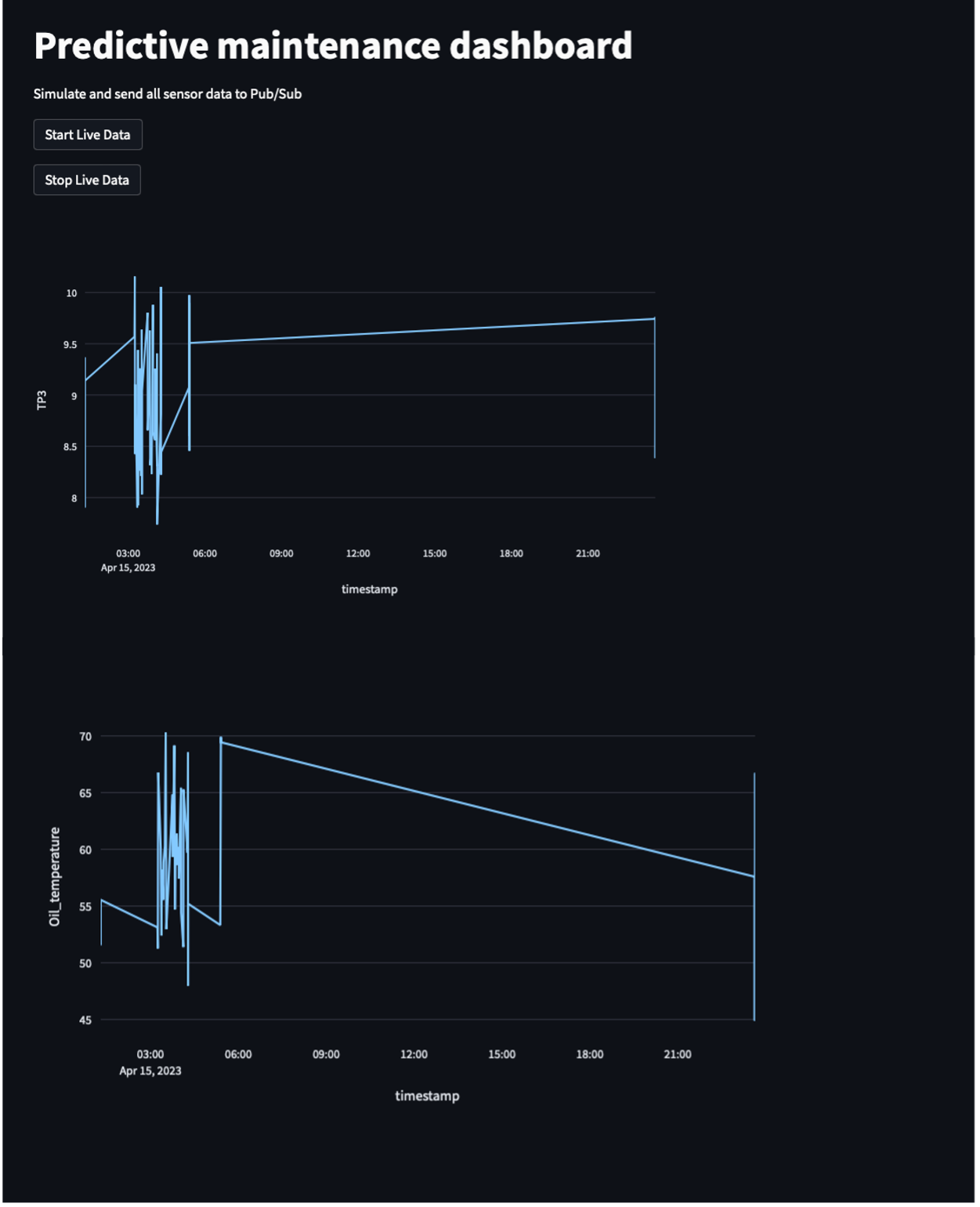
# IV. RESULTS

We carried out our experiments on Google Cloud which provided us with 16 vCPUs, 60 GB RAM. The training took almost 24 hours on all the normal data. We implemented our code in Python for both the SAE, and used Streamlit framework for the front end visualization. The dataset is available from January 2022 to July 2022. Three failure regions are provided in the dataset, which are excluded from the training stage and later included in the testing. We train our model in sequence taking the first three weeks first then testing on the next week, then move the training window by one week. The anomalies detected in the testing stage are excluded in the next training cycle. The first part of the results section discusses the results of the SAE model, and the second part illustrates our system as a whole.

## A. Sparse Autoencoder Results

The results of training and testing our SAE model are shown in Table II. It can be seen that our model is able to outperform the previous model in both the recall and F1-score, which means it is able to detect more anomalies in the data. It is worth noting that the old dataset had seven different failure regions known, while the newer one only had three. Our model doing better could be due to the fact that the training was done on all eight different analog sensors while the old model only used two analog signals. As said previously, it is hard to tell why the only trained their model on the first two analog signals.

There is still a huge room for improvement in terms of anomaly detection and prediction. The dataset is raw and different types of preprocessing could be applied on it to improve the performance of the model. Although the previous work buikt a different model only for the digital signals and produced decent results incorporating both the analog and digital signals is something that is yet to be tested. Moreover,



### Fig. 3. Dashboard

different types of model architectures can also be examined along with other types of autoencoders.

### TABLE II

SAE MODEL RESULTS

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Precision | Recall | F1-score |
| [3] | 70 | 33 | 45 |
| ours | 59 | 61 | 60 |

## B. ProActRail results

The above figure shows the output of the front-end app developed in Stream-lit, the dashboard allows the maintenance team to visualize the newly simulated data in real-time as well as any failures predicted by the model would show an alert on the dashboard to allow the user to take action. The is deployed on Google Cloud App Engine to cater for scalibility and availability of the app.

The implemented solution results in real-time data streaming, processing, and inference for a predictive maintenance task. The benchmark [5] aimed for the model to be successful it needs to predict failures before at least two hours to give the operating teams time to take actions. The developed system provided an interface that would allow the team to visualise the data as well as receiving data in a timely manner. Email notifications could easily be integrated with the system in the future the improve the response time to any failures.

Finally, the developed system can easily be integrated with the real-system though APIs, which will simply publish the real-time data the google pub/sub-topic and the system would function normally. More data with different types of failures and more details about them would highly increase the model precision specially if collected in real-time. The app code is available on GitHub.

# V. CONCLUSION

In conclusion, the developed solution harnesses the power of cloud computing to address the challenges associated with handling vast amounts of data and developing sophisticated machine learning models for anomaly detection in train air production units. By utilizing Google Cloud services, such as Cloud Pub/Sub, Cloud Functions, BigQuery, and Vertex AI Endpoint, the system effectively ingests, processes, and analyzes real-time sensor data, enabling proactive and data driven decision-making in train maintenance operations. In addition to training an auto-encoder model, which outperforms [3] in recall and f1-score. The model is deployed on Google cloud and runs real-time inference on newly simulated data.

## REFERENCES

1. C. Stenstrom, P. Norrbin, A. Parida, and U. Kumar, “Preventive and¨ corrective maintenance–cost comparison and cost–benefit analysis,” *Structure and Infrastructure Engineering*, vol. 12, no. 5, pp. 603–617, 2016.
2. R. Macedo, R. Benmansour, A. Artiba, N. Mladenovic, and D. Uro´ seviˇ c,´ “Scheduling preventive railway maintenance activities with resource constraints,” *Electronic Notes in Discrete Mathematics*, vol. 58, pp. 215– 222, 2017.
3. N. Davari, B. Veloso, R. P. Ribeiro, P. M. Pereira, and J. Gama, “Predictive maintenance based on anomaly detection using deep learning for air production unit in the railway industry,” in *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 2021, pp. 1–10.
4. J. Yan, Y. Meng, L. Lu, and L. Li, “Industrial big data in an industry 4.0 environment: Challenges, schemes, and applications for predictive maintenance,” *Ieee Access*, vol. 5, pp. 23484–23491, 2017.
5. B. Veloso, R. P. Ribeiro, J. Gama, and P. M. Pereira, “The metropt dataset for predictive maintenance,” *Scientific Data*, vol. 9, no. 1, p. 764, 2022.
6. E. Fumeo, L. Oneto, and D. Anguita, “Condition based maintenance in railway transportation systems based on big data streaming analysis,” *Procedia Computer Science*, vol. 53, pp. 437–446, 2015.
7. N. Davari, B. Veloso, G. d. A. Costa, P. M. Pereira, R. P. Ribeiro, and J. Gama, “A survey on data-driven predictive maintenance for the railway industry,” *Sensors*, vol. 21, no. 17, p. 5739, 2021.
8. M. Barros, B. Veloso, P. M. Pereira, R. P. Ribeiro, and J. Gama, “Failure detection of an air production unit in operational context,” in *IoT Streams for Data-Driven Predictive Maintenance and IoT, Edge, and Mobile for Embedded Machine Learning: Second International Workshop, IoT Streams 2020, and First International Workshop, ITEM 2020, Co-located with ECML/PKDD 2020, Ghent, Belgium, September 14-18, 2020, Revised Selected Papers 2*. Springer, 2020, pp. 61–74.
9. W.-J. Lee, “Anomaly detection and severity prediction of air leakage in train braking pipes,” *International Journal of Prognostics and Health Management*, vol. 8, no. 3, 2017.
10. A. Chaudhuri, “Predictive maintenance for industrial iot of vehicle fleets using hierarchical modified fuzzy support vector machine,” *arXiv preprint arXiv:1806.09612*, 2018.
11. C. Ordo´nez, F. S. Lasheras, J. Roca-Pardi˜ nas, and F. J. de Cos Juez, “A˜ hybrid arima–svm model for the study of the remaining useful life of aircraft engines,” *Journal of Computational and Applied Mathematics*, vol. 346, pp. 184–191, 2019.
12. M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, M. Hasan, B. C. Van Essen, A. A. Awwal, and V. K. Asari, “A state-of-the-art survey on deep learning theory and architectures,” *electronics*, vol. 8, no. 3, p. 292, 2019.
13. S. Khan and T. Yairi, “A review on the application of deep learning in system health management,” *Mechanical Systems and Signal Processing*, vol. 107, pp. 241–265, 2018.