Time-Series Fault Detection in the Tennessee Eastman Process Using LSTM and GRU Networks

Jeswanth Naidu Padi
Department of Computer Science
Blekinge Institute of Technology
Karlskrona, Sweden
jepd23@student.bth.se

Ayyappa Mitta
Department of Computer Science
Blekinge Institute of Technology
Karlskrona, Sweden
aymi23@student.bth.se

***Abstract*—This study examines how Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models can identify and classify faults in the Tennessee Eastman Process, which is a complex industrial time-series dataset. Both models performed well, with the GRU slightly outpacing the LSTM by achieving 89.7% accuracy and faster training times. Key evaluation metrics, including precision, recall, and F1-score, confirmed reliable fault detection across most categories. To improve transparency, LIME was used to explain model decisions by highlighting important process features like reactor pressure and feed flow rates. The project followed the CRISP-DM methodology to ensure a clear and repeatable process. In conclusion, this research shows that combining deep learning with explainability methods can create effective and trustworthy fault diagnosis systems for real-time industrial use.**

***Index Terms*—Fault detection, Tennessee Eastman Process, Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU).**

I. Introduction

In recent times, the significance of efficient process fault detection in industrial environments has gained more recognition. The primary goal is to enhance the identification of process faults and gain a deeper understanding of the dataset's complex characteristics. The Tennessee Eastman process (TEP) represents a sophisticated chemical system that has been widely utilized as a standard for evaluating fault detection and diagnosis methods [14]. As the industry evolves, modern industrial systems increasingly depend on sensor networks to monitor key operational metrics like temperature, pressure, and flow rate in real time. The sensors are crucial for ensuring that processes run safely, efficiently, and within designated parameters. However, as these systems become more advanced, they become more vulnerable to unexpected issues such as equipment breakdowns, sensor failures, or external disturbances. If these faults are not detected promptly, they can lead to severe consequences, including safety hazards, a decline in product quality, and costly production interruptions [4].

Traditionally, multivariate statistical process monitoring (MSPM) techniques have been employed to analyze process data. Recently, a number of machine learning methods have been introduced to tackle the fault detection issue. These techniques rely on the process of feature engineering, which involves manually windowing, combining, and manipulating raw data to extract meaningful signals. Effective feature engineering necessitates a thorough understanding of both process technology and machine learning techniques. Well-designed features often help mitigate the challenges posed by limited data and depend significantly on the expertise and experience of process engineers. Lately, deep neural networks have garnered significant attention and popularity as a preferred technique in machine learning for processing images, sounds, and text, often achieving top-tier results. Essentially, these methods help eliminate the laborious feature engineering phase by utilizing raw sensor data as input. Throughout the training phase, the deep neural network identifies the most effective signal representations (embeddings) and incorporates them in place of manually crafted features, thereby training the whole learning system. Models like LSTM and GRU are great for understanding the basic dynamics of multivariate process data [13].

The goal of this project is to find and diagnose these issues using deep learning methods. We will focus on Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks, as they are effective in modeling time-based sequences. Long Short-Term Memory networks, known as LSTMs, are a special kind of recurrent neural network (RNN). They can learn connections over long periods. LSTMs are designed to avoid the problems that come with long-term dependencies [9]. GRU, or Gated Recurrent Unit, is a kind of recurrent neural network (RNN) architecture that bears resemblance to LSTM. Similar to LSTM, GRU is intended to handle sequential data by enabling the selective retention or discarding of information throughout time. Nonetheless, GRU features a more straightforward structure than LSTM, having a reduced number of parameters, which can enhance its training ease and computational efficiency [10].

The motivation for using LSTM and GRU is their design benefits in capturing long-term dependencies in sequential data. This is crucial for analyzing behavior over time. LSTM has memory cells and gating structures. It is particularly good at preserving important patterns over long sequences. GRU is a simpler version of LSTM. It often provides similar results while being more efficient in terms of computation. This makes GRU especially attractive for real-time or resource-limited industrial uses. The proposed project outlines a two-

phase approach aimed at detecting and classifying faults in the Tennessee Eastman Process (TEP), a traditional simulation used in chemical production systems. The process is divided into two distinct steps:

- Step 1: Employ an LSTM model to detect the normal conditions and anomalies from the dataset.
- Step 2: The detected anomalies are further forwarded to a GRU model for the classification of the specific category of fault from 21 potential types.

Additionally, Local Interpretable Model-agnostic Explanations (LIME) are used to improve model interpretability. They provide insights into which sensor features significantly affect classification results. This is an important step in adopting explainable AI (XAI) in safety-critical applications [3]. The project results are measured using established metrics like accuracy, precision, recall, F1-score, and confusion matrices to evaluate how effective and reliable the proposed models are.

II. Related Work

Here are some literature reviews of previous research papers that are related to fault detection in the Tennessee Eastman Process.

- In 2022, Rashi Verma et.al [15] worked on fault Detection and Diagnosis (FDD), which is crucial across all sectors, and recent advances show that Machine Learning, particularly Deep Learning, offers precise and effective solutions.
- The study done by Qishan Wang et.al [16] introduces a novel method, PCC-MGAF, that encodes multivariate time series as 2D images using Pearson correlation and an enhanced Gram Angular Field, improving fault detection in the Tennessee Eastman process.
- Yi Cao et.al's paper [11] proposes a new method using Canonical Variate Analysis (CVA) to better extract dynamic latent variables, validated on the Tennessee Eastman process.
- Xing Liu et.al [17] introduced a CGRU-AE-based fault detection method that effectively learns features from time-series process data. By generating T^2 and SPE statistics, it demonstrates superior performance on the Tennessee-Eastman process compared to traditional methods.
- Mingfei Hu et.al [12] propose a KELM classifier optimized with an adaptive variation sparrow search algorithm (AVSSA) and feature selection via XGBoost to enhance fault diagnosis in complex industrial systems. Tested on the Tennessee Eastman process, the method achieved a 91% average accuracy, outperforming traditional approaches.

Knowledge Gap: Prior studies on fault detection in the Tennessee Eastman Process have mainly used statistical methods and traditional machine learning techniques like PCA, PLS, and SVM. While these methods can be effective for some faults, they often struggle to capture long-term dependencies in multivariate time-series data. Recent research has begun to explore deep learning methods; however, many of these approaches act as black-box models, which makes them hard to interpret. In this study, we address these issues by comparing the effectiveness of two recurrent neural network architectures, LSTM and GRU, while using LIME for better explainability. This approach not only evaluates the model's accuracy but also emphasizes the need for transparency and practical application, especially in deciding whether the more efficient GRU can serve as a good alternative to LSTM for real-time fault detection.

III. Methodology

The methods section explains the project's workflow. We use the CRISP-DM (Cross-Industry Standard Process for Data Mining) framework to keep a systematic, repeatable, and goal-oriented strategy. The experimentation approach includes machine learning techniques to detect faults in time-series data within the Tennessee Eastman Process.

A. Business Understanding:

The goal of this project is to create a system that can automatically detect and categorize process faults in the Tennessee Eastman Process (TEP) using deep learning methods. In complex industrial settings like TEP, faults can happen without warning and may lead to serious problems, including equipment damage, production delays, and safety risks if not addressed quickly. By using deep learning algorithms that can analyze large amounts of multivariate time-series data, the system aims to identify faults early, before they escalate into critical issues. Quick and accurate fault detection not only boosts the overall safety of the facility but also decreases unplanned downtimes and improves operational efficiency.

B. Data Understanding:

The dataset [7] used is the Tennessee Eastman Process (TEP) dataset, which is available on Kaggle. It includes simulated multivariate time-series data that represents different process conditions.

- **Fault-free-training.csv:** This file contains data from standard operations without faults. It is used to train the model to identify normal patterns
- **Fault-free-testing.csv:** It consists of the data without any anomalies and is used to test the model to discover trends.
- **Faulty-training.csv:** These files provide training data for each of the 21 fault types, where XX ranges from 01 to 21. Each file simulates a specific process fault.
- **Faulty-testing.csv:** This file includes a mix of faults. It is meant to test how well the model generalizes to new, unseen time-series patterns.

Each file records 52 different sensor measurements over time, including pressures, temperatures, flow rates, levels, and other process variables. The fault classifications are derived from the filenames and help categorize the sequences into 22 groups, with 0 for normal and 1–21 for faults.

Train-Test Split Approach:

- All samples from fault-free-training.csv are labeled as class 0 (normal).
- All samples from fault-training-01.csv to fault-training-21.csv, designated as classes 1 to 21.
- Samples from fault-testing.csv contain varied and potentially overlapping fault sequences. This setup enables us to evaluate the model's capacity to recognize fault types it has encountered before in various patterns.

C. Data Preparation:

The process of preparing data for training deep learning models with the Tennessee Eastman Process (TEP) dataset includes several key steps. These steps ensure that the time-series data is structured and optimized for better model performance [6] [1].

1. Label Assignment: Each sample in the dataset is linked to a specific fault scenario, as shown by the filename. The fault types are extracted and assigned numerical class labels. For instance, "fault01" gets the label 1, "fault02" gets label 2, and so on. Normal operations are labeled as 0. This numerical labeling is important for supervised learning, where the model learns to match input data patterns to specific fault types.

2. Normalization: To bring all features to a common scale, we use **MinMax normalization**. This method adjusts each feature to fall within the range of 0 to 1. Normalization is crucial because it speeds up training and prevents features with larger numerical ranges from dominating the learning process.

3. Sequence Creation: Since LSTM and GRU models learn from sequential patterns, we need to split the continuous time-series data into smaller, fixed-length sequences. In this case, each sequence consists of 100 consecutive time steps. These overlapping sequences help the model understand the temporal dependencies and trends in process variables over time, which are vital for accurate fault detection.

4. One-Hot Encoding: Fault labels are then turned into **one-hot encoded** vectors. For example, if there are 22 possible classes (including normal operations), each label becomes a 22-dimensional binary vector. Only the index that matches the correct class is set to 1, with all other indices set to 0. This format is necessary for training classification models using the categorical cross-entropy loss function, which compares the predicted probability distribution to the actual class distribution.

5. Shape Transformation: Finally, we reshape the input data into a 3D format required by recurrent neural networks: (samples, time steps, features). Each training sample now represents a sequence of 100 time steps, with each time step containing multiple process variables (features). This setup allows LSTM and GRU models to process data sequences and learn how features change over time with various fault types.

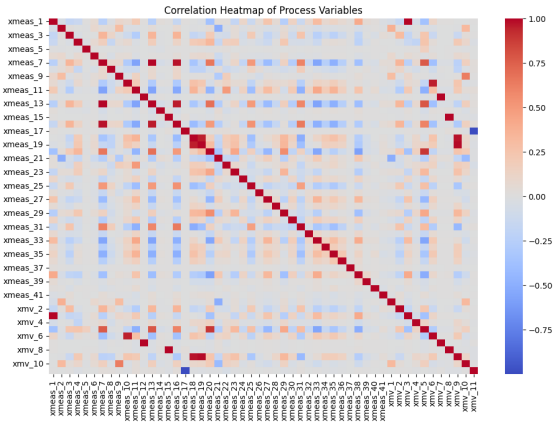

Fig. 1. Correlation HeatMap of Process Variables [5]

D. Model Training:

In this project, we used Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models, which are types of Recurrent Neural Networks (RNNs). These models are particularly good at time-series classification tasks, such as identifying faults in the Tennessee Eastman Process, because they can maintain temporal dependencies in sequential data. By using both LSTM and GRU, we aimed to evaluate their performance in terms of classification accuracy, training efficiency, and suitability for real-time applications.

GRU Architecture:

The GRU model is similar to the LSTM model but includes a GRU layer instead of an LSTM layer. Like the LSTM layer, the GRU layer has 64 units, followed by a dropout layer with a 0.3 dropout rate and a dense softmax output layer. GRUs are known for being computationally efficient compared to LSTMs since they use fewer gating mechanisms and have fewer parameters. This makes GRUs quicker to train and more lightweight, which is ideal for situations with limited computational resources or real-time applications. Despite its simpler structure, GRUs often perform as well as LSTMs, making them a strong choice for fault detection tasks.

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 50, 52)	0
gru (GRU)	(None, 64)	22,656
dense (Dense)	(None, 64)	4,160
dense_1 (Dense)	(None, 21)	1,365

Fig. 2. GRU Architecture

LSTM Architecture:

The LSTM model we developed has a simple but effective architecture. It starts with an input layer connected to an

LSTM layer with 64 memory units. This design captures long-term dependencies in time-series data, which is essential for identifying faults that evolve slowly. To reduce overfitting, we added a dropout layer with a rate of 0.3. This layer randomly turns off 30% of the neurons during training, helping the model generalize better to new data. The output layer is a dense layer using a softmax activation function. This is ideal for multi-class classification because it provides probability scores for all possible fault types, ensuring accurate classification. We also explored using autoencoders for feature extraction. By learning compressed representations of the input data, the autoencoder highlighted important patterns and reduced noise [8]. This improved the effectiveness of the fault classification that followed.

Layer (type)	Output Shape	Param #
input_layer_1 (InputLayer)	(None, 50, 52)	0
conv1d_1 (Conv1D)	(None, 50, 64)	10,048
max_pooling1d_1 (MaxPooling1D)	(None, 25, 64)	0
bidirectional_3 (Bidirectional)	(None, 25, 128)	66,048
bidirectional_4 (Bidirectional)	(None, 64)	41,216
repeat_vector_1 (RepeatVector)	(None, 50, 64)	0
bidirectional_5 (Bidirectional)	(None, 50, 64)	24,832
time_distributed_1 (TimeDistributed)	(None, 50, 52)	3,380

Fig. 3. LSTM Architecture

Model Parameters:

Both models were trained using the **categorical cross-entropy** loss function, which is common for multi-class classification tasks. Since our fault labels were one-hot encoded, this loss function allowed the models to accurately assess the difference between predicted probabilities and actual class labels.

For optimization, we used the **Adam optimizer**, which combines the advantages of both AdaGrad and RMSProp. Adam adjusts learning rates for each parameter during training, leading to faster convergence and better performance on sparse or noisy datasets, conditions often found in industrial process data.

We chose a batch size of 64, which strikes a good balance between training reliability and computational efficiency. The models were trained for up to 30 epochs, but we included early stopping to automatically halt training when the model's validation performance stopped improving. This approach helps prevent overfitting by ensuring that the models do not learn patterns that are too specific to the training data.

E. Evaluation Metrics:

To evaluate how well our fault detection models work, we used several evaluation metrics [2] that show different aspects of classification performance.

- **Accuracy:** This metric measures the total percentage of correct predictions made by the model across all fault categories. It provides a clear view of the model's overall performance but can be misleading if the classes are imbalanced.
- Precision, Recall, and F1-Score: Since the Tennessee Eastman Process includes several fault types with varying frequencies, it's important to assess performance for each class.
  - **Precision** shows the ratio of accurately predicted fault instances to all instances identified for that fault. It indicates the reliability of positive predictions.
  - **Recall** evaluates how well the model identifies true fault cases. It shows how many actual faults are detected correctly.
  - **F1-Score** combines precision and recall into a single measure, providing a harmonic mean that is useful for considering both false positives and false negatives.
  - To handle class imbalance, we calculated these metrics as macro-averaged scores, treating each class equally regardless of its size.
- **Support:** is the count of actual occurrences of each class in the dataset. It indicates the true instances available for that class. This metric helps contextualize precision, recall, and F1-score by illustrating class distribution.
- **Confusion Matrix:** This matrix offers a clear view of the model's performance across each fault type by showing the number of correct and incorrect predictions for each class.

Explainability with LIME:

In addition to measuring accuracy, we wanted to improve the clarity of our models' decisions. To do this, we used LIME (Local Interpretable Model-Agnostic Explanations). This method explains individual predictions by locally approximating the model with a simpler, easier-to-understand version. Through LIME, we identified which features significantly influenced the model's fault predictions. In many cases, variables like reactor pressure and component B feed flow frequently appeared as key factors. This insight not only confirms that the model makes decisions based on important process variables but also builds user trust by explaining the reasoning behind specific fault classifications.

IV. Results and Analysis

The evaluation of the GRU and LSTM models in the fault detection task showed strong results from both, with notable differences.

A. LSTM Model:

The LSTM model achieved a respectable overall accuracy of 84% . Although this is slightly lower than the GRU, it is still quite competitive. The training speed of the LSTM was slower compared to the GRU, but its structure is well-known for effectively capturing long-term dependencies, which is important for complex time-series data. Like the GRU, the LSTM showed balanced class-level performance with strong precision and recall for most faults. The same challenge in distinguishing between fault-00 and fault-09 was observed,

highlighting the difficulty in differentiating these instances rather than pointing to a specific problem with the model. The training and validation loss curves showed a steady learning pattern without significant overfitting, reinforcing the reliability of the model's predictions. Figure 4 shows the results of the trained LSTM model.

	precision	recall	f1-score	support
Normal	0.76	0.98	0.86	47998
Anomaly	0.98	0.69	0.81	47998
accuracy			0.84	95996
macro avg	0.87	0.84	0.83	95996
weighted avg	0.87	0.84	0.83	95996

Fig. 4. LSTM Results

B. GRU Model:

The GRU model achieved an impressive overall accuracy of about 89.7% slightly better than the LSTM. A major benefit of the GRU was its faster training speed and better computational efficiency. This makes it more suitable for real-time fault detection, where quick model updates are essential. At the individual class level, the GRU maintained high precision and recall scores for most fault categories, successfully identifying a wide range of faults with few errors. However, the model struggled to distinguish between the normal condition (fault-00) and fault-09. This difficulty likely arose from overlapping patterns in sensor readings for these classifications. Training stability for the GRU was also impressive, as shown by the close alignment of training and validation loss curves. This suggests that the model was well-regularized and managed to avoid overfitting, which helped it generalize to new data. Figure 3 shows the results of the trained GRU model.

	precision	recall	f1-score	support
fault_00	0.000	0.000	0.000	4800
fault_01	0.999	1.000	0.999	4800
fault_02	1.000	1.000	1.000	4800
fault_03	0.743	0.786	0.764	4800
fault_04	0.999	0.998	0.998	4800
fault_05	0.999	0.995	0.997	4800
fault_06	0.999	1.000	0.999	4800
fault_07	1.000	0.999	1.000	4800
fault_08	1.000	1.000	1.000	4800
fault_09	0.334	0.509	0.403	4800
fault_10	0.997	0.999	0.998	4800
fault_11	0.999	0.998	0.998	4800
fault_12	0.999	0.994	0.996	4800
fault_13	0.999	0.999	0.999	4800
fault_14	1.000	1.000	1.000	4800
fault_15	0.407	0.586	0.481	4800
fault_16	0.999	0.996	0.997	4800
fault_17	0.999	0.999	0.999	4800
fault_18	0.994	1.000	0.997	4800
fault_19	1.000	0.983	0.991	4800
fault_20	1.000	0.999	0.999	4796
accuracy			0.897	100796
macro avg	0.879	0.897	0.887	100796
weighted avg	0.879	0.897	0.887	100796

Fig. 5. GRU Results

C. LIME Interpretations:

LIME provided useful insights into the features influencing fault predictions for both models. Key variables, such as reactor pressure, feed flow from components B and D, and stripper level, were often identified as the most important. These findings align with domain knowledge, confirming that the models focus on crucial sensors that indicate the health and status of the Tennessee Eastman Process. This level of interpretability not only boosts the trustworthiness of the predictions but also offers actionable insights for process engineers managing plant operations.

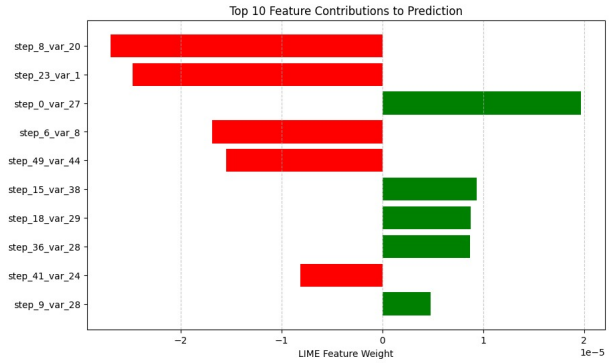

Fig. 6. LIME feature weights

In summary, both the GRU and LSTM showed strong performance in fault detection, displaying high accuracy and stable training. The GRU's faster training time and slightly better accuracy make it a desirable choice for practical use, while the LSTM's strengths in sequence learning remain valuable. The explainability analysis with LIME further emphasizes the significance of the models by highlighting important process variables.

V. Discussion

A. *RQ1: How accurately can the LSTM model distinguish between normal and anomalous operation in the TEP dataset?*

The LSTM model showed strong effectiveness in differentiating between normal and faulty conditions in the Tennessee Eastman Process dataset. By capturing long-term patterns in sequential data, the LSTM successfully identified subtle anomalies in normal process behavior across various related variables. With an overall accuracy close to 84% it proves reliable for detecting anomalies, which is crucial for identifying faults early. The model's precision and recall scores further demonstrate its good performance in accurately finding faults while minimizing false positives. However, some challenges remain, such as occasionally misclassifying normal situations and specific faults like fault-09, which share similar sensor patterns. This suggests that while the LSTM effectively models time-based relationships, there's still room for improvement in how features are designed or represented to enhance its ability to tell apart different conditions.

Importantly, the LSTM's ability to handle multivariate time-series data makes it ideal for real-world industrial settings where sensor readings vary over time. Its strength in capturing

complex process dynamics supports reliable monitoring, which can help prevent costly downtime and improve safety in plants. However, the longer training time and wider range of parameters compared to the GRU might limit its scalability in resource-constrained environments, making it better suited for places with plenty of computational resources where accurate anomaly detection is vital.

B. *RQ2: How effectively can the GRU model classify specific fault types from the detected anomalies?*

The GRU model not only matched the LSTM but slightly exceeded it in overall accuracy, reaching about 89.7%İt excelled particularly in classifying specific fault types among the detected anomalies. Its simpler structure, with fewer parameters, allowed the GRU to train faster and operate more efficiently, making it a great choice for real-time fault classification tasks. In terms of specific faults, the GRU maintained high precision and recall across nearly all fault classes, effectively distinguishing between different faults with few errors. This is especially important in monitoring industrial processes, where knowing the exact fault type allows for more focused and effective responses.

Moreover, using LIME for explainability showed that the GRU's predictions depend on key process variables, such as reactor pressure and feed flow rates, which align well with industry knowledge. This transparency builds user confidence and aids decision-making by providing clear insights into why faults are identified. The GRU's combination of accuracy, efficiency, and interpretability makes a strong case for its use in industrial settings where both performance and practical application matter.

C. *Reflections:*

In summary, the findings highlight the complementary strengths of LSTM and GRU models in detecting and diagnosing faults in complex industrial systems. The LSTM's ability to distinguish between normal and abnormal operations ensures early and reliable fault detection, which is vital for maintaining safe and stable plant conditions. On the other hand, the GRU's effective and accurate classification of specific fault types provides important details for efficient fault diagnosis and response. The use of LIME for interpretability addresses a key limitation of many deep learning methods by making model decisions clearer and easier to understand for users. Following the CRISP-DM methodology provides a structured and thorough approach, from data understanding and preprocessing to modeling and evaluation. This ensured that every step was purposeful and repeatable. Combining high-performing models with explainability and a structured process framework helps bridge the gap between academic research and practical, applicable solutions in industrial fault detection. Future research can build on this by exploring hybrid models or further improving interpretability to increase user confidence and operational efficiency even more.

VI. Limitations

This study presents encouraging results. However, several limitations should be considered when evaluating the findings. These limitations highlight areas where the approach may face challenges or need improvements to ensure reliability and use in real-world situations. Understanding these constraints is vital for guiding future research and practical application efforts.

- **Synthetic Data:** The TEP dataset is generated and cleaner than actual industrial data, which often contains more noise and missing values. This can impact model effectiveness in real-world situations.
- **Class Imbalance:** Some fault categories have very few instances. This leads the model to focus on more common faults and may cause it to miss less frequent ones.
- **Local Interpretability:** LIME gives explanations for individual predictions. However, it may not effectively show how the model behaves overall across the entire dataset.
- **No Extensive Hyperparameter Tuning:** The models were not fine-tuned with thorough hyperparameter optimization. This means that better performance might be achieved with more tuning.

VII. Conclusion and Future Work

A. *Conclusion:*

This project highlights the strong potential of recurrent neural networks, specifically the LSTM and GRU architectures, for identifying and classifying faults in complex industrial time-series data, such as that from the Tennessee Eastman Process. Both models effectively captured the dynamic and multivariate features of the data, demonstrating high predictive accuracy and good generalization to new fault scenarios. The slightly better performance and efficiency of the GRU make it a favorable option for real-time fault detection in resource-limited environments, while the LSTM remains useful for modeling longer time dependencies.

A key benefit of this work is the use of explainability techniques, particularly LIME, which clarified how these deep learning models make decisions. By identifying the main process variables that affect predictions, the project built a connection between model performance and interpretability. This connection is crucial for gaining user trust and encouraging adoption in industrial settings. Furthermore, following the CRISP-DM methodology created a clear and transparent workflow. It addressed understanding business issues, data, and deploying a reliable and interpretable model.

In summary, this study shows that combining effective deep learning methods with interpretability tools and a structured development process can lead to practical and trustworthy fault detection systems. These findings lay a strong groundwork for future advancements and real-world applications focused on improving safety, reducing downtime, and optimizing process operations in industrial environments.

B. Future Work:

- Incorporating attention mechanisms could help the models recognize and prioritize important time steps in the sequence data. This may lead to better fault detection accuracy. The next key step is to test and validate the models using real-world data from industrial plants. This will assess their performance in more variable and noisy environments compared to synthetic datasets.
- Thorough hyperparameter tuning using methods like grid search or Bayesian optimization could improve performance even further. Exploring different architectures, such as CNN-LSTM hybrids, could boost both feature extraction and temporal modeling abilities. In addition to LIME's local explanations, using global interpretability techniques like SHAP would provide better insight into how the model behaves.
- It's essential to integrate the developed models with industrial control systems like SCADA or PLC platforms for live monitoring and automated fault responses. This will shift focus from research to practical application. Such integration would enable real-time fault detection and decision support, improving safety, reducing downtime, and optimizing the efficiency of industrial processes.

REFERENCES

[1] "7.4. Imputation of missing values." [Online]. Available: https://scikit-learn/stable/modules/impute.html

[2] "Evaluation Metrics in Machine Learning," section: Machine Learning. [Online]. Available: https://www.geeksforgeeks.org/machine-learning/metrics-for-machine-learning-model/

[3] "Explain your model predictions with LIME." [Online]. Available: https://kaggle.com/code/prashant111/explain-your-model-predictions-with-lime

[4] "(PDF) Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation." [Online]. Available: https://www.researchgate.net/publication/262877889

[5] "seaborn.heatmap — seaborn 0.13.2 documentation." [Online]. Available: https://seaborn.pydata.org/generated/seaborn.heatmap.html

[6] "StandardScaler." [Online]. Available: https://scikit-learn/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

[7] "Tennessee Eastman Process CSV." [Online]. Available: https://www.kaggle.com/datasets/afrniomelo/tep-csv

[8] "Time-series forecasting with LSTM autoencoders." [Online]. Available: https://kaggle.com/code/dimitreoliveira/time-series-forecasting-with-lstm-autoencoders

[9] "Understanding LSTM Networks – colah's blog." [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[10] Anishnama, "Understanding Gated Recurrent Unit (GRU) in Deep Learning," May 2023. [Online]. Available: https://medium.com/@anishnama20/understanding-gated-recurrent-unit-gru-in-deep-learning-2e54923f3e2

[11] Y. Cao and R. T. Samuel, "Dynamic latent variable modelling and fault detection of Tennessee Eastman challenge process," in *2016 IEEE International Conference on Industrial Technology (ICIT)*, Mar. 2016, pp. 842–847. [Online]. Available: https://ieeexplore.ieee.org/document/7474861

[12] M. Hu, X. Hu, Z. Deng, and B. Tu, "Fault Diagnosis of Tennessee Eastman Process with XGB-AVSSA-KELM Algorithm," *Energies*, vol. 15, no. 9, p. 3198, Jan. 2022, number: 9 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/1996-1073/15/9/3198

[13] I. Lomov, M. Lyubimov, I. Makarov, and L. E. Zhukov, "Fault detection in Tennessee Eastman process with temporal deep learning models," *Journal of Industrial Information Integration*, vol. 23, p. 100216, Sep. 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2452414X21000145

[14] S. Nasif, A. Tasnim, and N. Sanzida, "A Collection of Exploratory Data Analysis Techniques for Process Fault Detection in the Tennessee Eastman Dataset," in *2024 2nd International Conference on Information and Communication Technology (ICICT)*, Oct. 2024, pp. 86–89. [Online]. Available: https://ieeexplore.ieee.org/document/10839661/

[15] R. Verma, R. Yerolla, and C. S. Besta, "Deep Learning-based Fault Detection in the Tennessee Eastman Process," in *2022 Second International Conference on Artificial Intelligence and Smart Energy (ICAIS)*, Feb. 2022, pp. 228–233. [Online]. Available: https://ieeexplore.ieee.org/document/9743021

[16] Q. Wang, G. Zhang, Y. Huang, and Y. Zhang, "Imaging Multivariate Time-Series to Improve Fault Detection: Application on Tennessee Eastman Process," in *2021 7th Annual International Conference on Network and Information Systems for Computers (ICNISC)*, Jul. 2021, pp. 196–201. [Online]. Available: https://ieeexplore.ieee.org/document/9603806

[17] J. Yu and X. Liu, "A Fault Detection Method based on Convolutional Gated Recurrent Unit Auto-encoder for Tennessee Eastman Process," in *2020 Chinese Automation Congress (CAC)*, Nov. 2020, pp. 1234–1238, iSSN: 2688-0938. [Online]. Available: https://ieeexplore.ieee.org/document/9326895