Leveraging Temporal Fusion Transformers and Domain-Specific LLMs for Real-World Industrial Sensor Forecasting and Decision Support

by
Ori Ben Yosef

B.S. Environmental Health Sciences (2013)
Hadassah Academic College, Jerusalem

Submitted to the System Design and Management Program
in partial fulfillment of the requirements for the degree of

Master of Science in Engineering and Management

at the
Massachusetts Institute of Technology

September 2025

Authored by: Ori Ben Yosef
MIT System Design and Management Program
August 8, 2025

Certified by: Dr. Eric Rebentisch
Research Associate, Sociotechnical Systems Research Center & Lecturer
Thesis Supervisor

Accepted by: Joan Rubin
Executive Director
MIT System Design and Management Program

Leveraging Temporal Fusion Transformers and Domain-Specific LLMs for Real-World Industrial Sensor Forecasting and Decision Support

by

Ori Ben Yosef

Submitted to the MIT System Design and Management Program on August 8, 2025, in partial fulfillment of the requirements for the degree of Master of Science in Engineering and Management

ABSTRACT

In process industries, large volumes of sensor data are generated continuously, yet much remains underutilized for proactive decision-making. This thesis explores a novel architecture that combines deep learning and large language models (LLMs) to forecast, interpret and prevent process threshold violations in an industrial process facility. A Temporal Fusion Transformer (TFT) model was trained on 3 months of real-world, multivariate sensor data (1-minute resolution across 31 sensors) to predict 12-minute-ahead process parameter exceedances. Forecast outputs were passed to a costume-built domain-specific GPT-4.1 model, configured using prompt engineering, graph interpretation capabilities, and a retrieval-augmented generation (RAG) system incorporating expert literature and process knowledge. The GPT model synthesized probabilistic forecasts into well-structured team-based Five Whys root cause analyses, where virtual domain experts questioned each other to refine the diagnosis, a long term mitigation plan to remove the root causes found, and simulation-driven, per-unit prevention plan, generated by testing alternative process settings with the trained deep learning model and selecting the minimal production disturbance configuration that prevented the predicted violation, all while leveraging domain-specific knowledge to ensure operational feasibility and engineering trustworthiness by explicitly referencing authoritative sources from its RAG library, such as procedures and technical text books, to maintain compliance with stakeholder needs. Evaluation showed that GPU-trained deep learning model significantly outperformed CPU-trained equivalents in mean quantile loss metrics. Subject matter expert evaluation of the LLM's responses indicates that the LLM's insight quality improved as more domain knowledge was added leading to greater specificity, unit level differentiation in recommendations. This dual-model system demonstrates a scalable approach to combining forecasting and interpretability in one pipeline, offering preventative, actionable, domain-specific support for engineers, operators, and managers in complex industrial environments.

Thesis Supervisor: Dr. Eric Rebentich
Title: Leveraging Temporal Fusion Transformers and Domain-Specific LLMs for Real-World Industrial Sensor Forecasting and Decision Support

Acknowledgments

First, I want to acknowledge my wife, Nofar, who was brave enough to accept the challenge and move with our two kids 5,500 miles to a different country, with different language and different culture and enter the gates of MIT together with me, undeniably a true challenge. during this time, with strengths springing from what feels like an endless source, you have pushed me and the kids forwarded to finish the race, half the intended time. And on top of that also find time to organize family trips like Disney World and even creating new life bonding friendships. Even though, I was closed in my office most of the year, I was always aware of it and will never forget it. You are a true lioness, and without you this journey would not have been possible, I love you deeply for that.

To my kids, Daniel who is six years old now, and Emmy who is 4 years old, you both gave me the power to continue with happy and funny moments I will never forget. Both of you moved to a new country with different language and absolutely flourished and defeated every challenge in the way. Your journey reminded me every day how small my challenge is compared to yours, and it put things into perspective for me. I love you and thank you for this year. I truly hope you will read this work one day.

I want to thank Chevron for sponsoring me and sending me to the MIT SDM program, thru the digital scholarship program. This shows strategic thinking that helps Chevron be one of the world's leading companies.

I want to thank my friends, who are also known as the night watchers: Victor Momoh, Dawit Dagnaw, and Brent Del-Mundo, who supported me in countless days (and nights!) and gave me a place to release some of the tension with a funny story or just talking during our long, seemingly endless nights together in the virtual meetings.

Lastly, I would like to thank my advisor, Dr. Eric Rebentich, who made the unthinkable (in my perspective) an actual reality. I am not sure if Eric remembers, but during one of the gala events in the boot camp of the SDM program (right at the start), I talked with him about my idea to use sensor data to gain insights for process engineers, and he told me that's it is a valid idea. This gave me powers to go in this path. At this point, it was just an idea, but Eric guided me in this thesis in the right direction to make it a reality.

Table of Contents

List of Figures

List of Tables

Acronyms, Terms, and Definitions

(In Alphabetical Order)

Definitions

- CUDA - CUDA is NVIDIA's parallel computing platform and programming model that allows software like PyTorch or TensorFlow to run computations directly on GPU instead of the CPU
- DARTS - Darts is an open-source Python library designed for time series forecasting and analysis. It is built on PyTorch for deep learning models, enabling GPU acceleration via CUDA.
- Deep learning - also called a deep neural network (DNN), is a machine learning approach using multi-layered neural networks to automatically learn complex patterns and representations from large datasets.
- Domain Specific - Tailored to a particular subject or industry area.
- Forecasting - Predicting future values based on past data trends.
- GPT - Generative Pretrained Transformer, a type of LLM.
- Knowledge graphs - Structured data representing relationships between entities.
- LLM - Large Language Model trained to understand and generate text.
- Long-term Forecasting: This refers to multi-step ahead forecasting problems where the prediction length P is typically greater than the threshold for short-term forecasting, often above 48 steps [1]
- Multistep - Generates multiple future points, not just one.
- Multivariate Time Series - a time series is multivariate if all its samples are vectors. This implies that at each time step, multiple related values or features are recorded simultaneously [1].
- Short-term Forecasting - One-step-ahead forecasting is included in short-term forecasting. More broadly, multi-step ahead forecasting can be divided into short-term and long-term.

The typical threshold value for prediction length (P) between short-term and long-term forecasting generally ranges between 2 and 48 [1].

- Stationarity - Stationary time series where "the dynamical process that generated it does not change over time [1].
- System prompt/message - Initial instructions guiding model behavior and tone.
- Task-agnostic - Works across many tasks without specific customization.
- Time Series - A time series is a sequence of data points collected or recorded at successive, evenly spaced points in time. Each data point represents a value observed at a specific moment, and the order of observations is essential, as it reflects how the variable evolves.
- Transformer - Neural network architecture for processing sequential data.
- Univariate Time Series: A time series is considered univariate if all its samples are scalar. This means that at each point, only a single value is recorded or measured for the observed phenomenon.
- User query/message - Input from user seeking a specific response.

Acronyms:

- CNN – Convolutional Neural Network
- CO2 – Carbon Dioxide
- DNN – Deep Neural Network
- ESN – Echo State Network
- GAN – Generative Adversarial Network
- GNN – Graph Neural Network: Deep learning on graph-structured data.
- GRU – Gated Recurrent Unit
- LSTM – Long Short-Term Memory
- MPS (on macOS) – Metal Performance Shaders: Apple GPU-accelerated deep learning framework.

- NO2 – Nitrogen Dioxide
- O3 – Ozone:
- PM10 – Particulate Matter <10μm
- PM2.5 – Fine Particulate Matter <2.5μm
- SO2 – Sulfur Dioxide
- TCN – Temporal Convolutional Network

1 Introduction

1.1 Background

Process industries are a type of production that adds value by mixing, separating, forming, and/or performing chemical reactions [1]. Examples for process industries: chemicals, plastics, tobacco, pharmaceuticals, wood and wood products, rubber, base metals, coal, petroleum, ceramics, food, beverages, textiles, paper, and paper products [2].

These industries employ various sensors that monitor variables like temperature, pressure, flow rate, and emissions [3]. As a result, they generate large amounts of historical and real-time data [4], [5], often sampled at high frequency [6] and stored in process historians' databases (e.g., PI System by AVEVA).

Despite this abundance of data, much remains underutilized [7]. In many facilities, decision-making depends on rule-based control systems, operator intuition, or lagging indicators like alarms [8].

Innovations in artificial intelligence, particularly in deep learning [9] and natural language processing, present new opportunities to unlock the full potential of operational data [4]. Deep Neural Networks (DNNs) have shown success in finding hidden correlations and detecting complex patterns in time-series data and outperforming traditional machine learning techniques [9]. Meanwhile, Large Language Models (LLMs), such as GPT-based systems, have shown the ability to synthesize information, interpret results, and generate meaningful insights in natural language[10], [11].

Sometimes, due to the complexity of the industrial process, it is hard to diagnose why problems occur [12]. These problems can have significant financial implications. A four-year report (2019-2023) by SIEMENS [13], based on organizations surveyed in different countries worldwide across sectors - automotive, fast moving costumer goods (FMCG), heavy industry and oil & gas suggests that unplanned downtime costs of the world's 500 largest manufacturing companies are estimated to be 11% of their annual revenues, totaling USD 1.4 trillion. Deep learning and domain-specific large language models can potentially diagnose and reduce the scale of the problem and therefore can save a portion of this cost every year.

1.2 Research objectives

The objectives for this research are as follows:

Design and build

- Design a (multivariate, long-term, time-series) deep-learning architecture and build a working model whose inputs are multivariate time-series sensor values from industrial process operationI .Its output is the prediction of unwanted values.
- Design a domain-specific GPT architecture and build a working model to interpret the deep-learning predictions and respond with in-context root cause analysis and mitigation steps to prevent the unwanted values and their future recurrence.

Evaluate

- Evaluate the accuracy (Quantile loss) of the deep-learning model forecasting.
- Evaluate the correctness, usefulness, strength, and limitations of the domain-specific GPT responses by human SME (Subject Matter Experts).

1.3 Research

Deep Neural Network (DNN):

Q1: Are the DNN's results accurate enough to be helpful in a real-world industrial scenario?

Q2: Can DNNs be useful in real-time scenarios?

Q3: Can you train the model without feature engineering and receive accurate results?

Q4: Can a model highlight the most contributing features to the exceedance?

Domain-Specific GPT:

Q1: Can the model give a correct root cause analysis (RCA) based on the input from the deep neural network?

Q2: Can the model formulate precise, context-aware operational recommendations that balance constraint compliance and production optimization?

Q3: Does the model improve the decision-making of the process engineer?

1.4 Scope and limitations

This research addresses a real-world operational challenge encountered in the process industry sector, aiming to forecast a violation in a specific parameter that directly impacts revenue streams in a dynamic and complex process environment.

The research uses multivariate time series process sensor measurements in the process industry. Specifically, the process of removing condensate from natural gas. In this process, a material stream is processed by seven parallel trains and measured for various parameters using 31 sensors.

Consequently, the deep neural network and the domain-specific GPT are models developed and trained exclusively on measurements and literature for this environment. It should be noted that

applying models to other industrial contexts without proper adaptation might reduce their performance.

In addition, the model is trained on historical data and does not operate in real time. Therefore, the applicability of integrating the model into live operations is not in the scope of this research.
The study does not consider the trade-offs associated with maintaining situation awareness, which may be relevant in real-time deployment scenarios.

1.5 Personal Motivation

My whole career led me to this moment. My career started as a consultant, making process surveys and emission surveys. During this time, I learned about the process industry hands-on. Understand the different types of instruments, including sensors with which I have interacted for environmental threshold violations investigations. The idea for the thesis was seeded in my mind afew weeks before I joined the MIT SDM. It was focused on environmental compliance. Can you use all the process sensors to predict or get new insights by looking at correlations in this vast amount of data? If something happens in the process that affects this violation, you should see it in the sensors. The problem is that so many of them are used mainly for checking process health, not for any special insights or forecasts. Basic rule-based alarms of linear correlations (if this measurement is above X, show alarm). There must be something you can do with all this data. Can an LLM help with this also?
This was the start of my journey at MIT, where I took as many classes as possible to gain enough knowledge to build this model, including deep learning and machine learning courses.

1.6 Structure of the thesis

The thesis is structured based on the following chapters: the introduction outlines the motivation, research questions, and the proposed AI-based framework for forecasting and decision making in industrial systems. The literature review surveys prior work in deep learning for time-series forecasting, large language models in decision support, and the history of sensors in the process industry. The methodology chapter details the system components, including the Temporal Fusion Transformer training details, data preprocessing, and domain-specific GPT prompt engineering. The case study presents the scope and location of the sensors within the industrial process. The results chapter reports quantitative metrics results for the deep learning model and qualitative metrics results for the domain-specific GPT received by the SME. The discussion interprets the results and reflects on practical implications. Lastly, the conclusion recaps the main findings and outlines guidelines for future work.

2 Literature Review

2.1 Sensor-Based Data in Industrial Processes

In industrial processes, various variables are being measured to control the operation of their systems. Those can be divided by the measured variable that needs to be measured. Below is a partial list of measurement variables and application examples [14]:

Flow: flow rates of gases, liquids, and solids; Level: liquid and solid level measurement in atmospheric vessels and pressurized vessels; Temperature: liquid and gas temperature measurement; Pressure: measuring pressure in pipes and vessels; Density: concentration, composition and calorific value in fuel requires density measurement to be determined; Noise: acoustic emission analysis for nondestructive testing of vessels and pipes; Vibration: mechanical health monitoring by measuring vibration in rotary machinery; Weight: scaling of hoppers and tanks; Analysis: (1) ambient air quality, (2) turbidity, pH and total organic carbon (TOC) in wastewater streams, (3) carbon monoxide (CO) as indicator for incomplete combustion in boilers, (4) moisture in process gas, and (5) nitrogen oxide (NO_x), Sulphur oxide (SO_x), carbon monoxide (CO), carbon dioxide (CO_2) and particulate in industrial emission stationary sources such as stacks.

In a modern process industry plant, there are tens of thousands of sensors [15], monitoring many crucial variables along the process, and enabling factory automation and operation in real-time. Generally, the production data is generated on the field level, acquired by a supervisory control and data acquisition software (SCADA), and stored in a data historian. A data historian is sometimes called a real-time information system and process information system[16]. An example of data historian software is AVEVA™ PI System™ [17].

2.2 Deep Learning in Industrial Analytics

The type of data that is stored in the process data historians is time series. The process variables' results are stored with a timestamp that includes the date and time and are ordered promptly. This allows operators, engineers, and managers access to a long-term process data history used for analysis, reporting, and decision making.

Industrial companies are already in the significant data era. According to a report written back in 2011 by McKinsey & Company [18], "manufacturing stores more data than any other sector - close to 2 exabytes of new data stored in 2010." In the same vein, GE [19] in their report published in 2018 gives an example of a Consumer Packaged Goods (CPG) manufacturer that generates "5,000 data samples every 33 milliseconds". That's 13 billion samples per day. According to a note from the report, this data is generated from only one machine out of many that produce a specific personal care product. This is consistent with the data obtained in 2023 by [4], which gives an example of Air Liquide that collects 3.5 billion data points daily across the company. This underscores the vast number of data sets produced by industrial companies daily.

Even though a vast amount of data is technically available, this does not mean it is easily accessible or applicable. According to [7], Industrial big data analytics faces numerous challenges, as there is a lack of an effective and efficient online large-scale machine learning framework that can analyze and give insights from high-volume heterogeneous data sources: sensor measurements, event logs, and geospatial data. Some suggest that the big data collected by the industry sector is very valuable. McKinsey and Company has published a report [18] stating that industrial big data has the potential to revolutionize economies and deliver a new era of productive growth.
However, literature shows that big data in the process industry is generally underutilized. The underlying reason might be the process that the data represents. Manufacturing in the process industry, such as the chemical sector, is notoriously complex. Mega complexes of pipes and

reactors, mills, cooling towers, heat exchanges, pressurized tanks, silos, furnaces – mixing, crushing, and reacting multiple types of raw materials into a finished product. Effects in the process can be nonstationary, nonlinear, multivariate, long-range, and short-range, time-series, with billions of data points collected daily. One example can be trying to forecast a non-planned recurring plant shutdown, based on a multi-year labeled history of the plant's sensors. Logically, the unplanned shutdown of a furnace has a cause. The causes might be overlooked and not obvious; otherwise, it would not be a recurring issue. Sometimes, small phenomena in different places can have a correlation, potentially suggesting a shutdown is coming. Can this be forecasted 30 minutes in advance, for example? Literature suggests that this challenge has been faced for almost a century now, by traditional statistical methods, machine learning, and lately, deep learning [20]

Traditional statistical models often dictate restrictive *a priori* assumptions: (1) Many traditional statistical models, such as Autoregressive models (ARMA, AR), are fundamentally "based on the assumption that time series are stationary [21]." A stationary time series is one where "the dynamical process that generated it does not change over time [9]", meaning its statistical properties like mean and variance. Remain constant over time [20]. (2) Another assumption is that traditional statistical approaches often require assumptions of a "normal distribution" for the time series data [21]. (3) Classical methods frequently assume "linear correlation" and "independence" among data points [21]. However, real-world time series data sets are often non-stationary with non-linear relations [21] and complex dependencies among the different data points. Several machine learning models, like random forest and decision trees, offer strong predictive power in statistical forecasting. A Random Forest builds many decision trees and combines their outputs, enabling it to manage high-dimensional data and capture complex, nonlinear relationships among features [20]. Previous studies [20] have shown that emerging technologies such as the Internet of Things (IoT) have significantly improved the efficiency of data acquisition and storage. With the arrival of the big data era, the volume and complexity of generated data continue to grow rapidly. Statistical forecasting models, particularly those utilizing machine learning, must evolve to handle

large-scale, high-dimensional data and meet the growing demand for precise predictions across diverse sectors. Practical applications often deal with complex, nonlinear data patterns, necessitating more adaptable and precise modeling approaches.

In contrast, deep learning models emerge as a potential solution due to their flexibility and inherent capability to grasp complex, non-linear relations among input data and effectively manage high-dimensional time series without requiring such rigid a priori assumptions [9], [20]. Deep learning can automatically study complex time features and patterns from large-scale data, removing the need for manual feature engineering and improving prediction accuracy by catching long-term dependences and weak signals [9], [20]. This enables deep learning algorithms to outperform traditional machine learning methods in forecasting applications consistently [9], [22].

Deep learning has shown time-series prediction ability in many domains, including energy, meteorology, air pollution monitoring, finance, health, traffic, and industrial production [9]. Below are a few examples from [9]:

1. Traffic Flow Forecasting (Multivariate, Long-term): Models like Graph Neural Networks (GNNs) and various Transformer variants are used for predicting traffic flow and speed on roads.
2. Air Pollution Data Forecasting (Multivariate, Short-term): Deep learning models, including Temporal Convolutional Networks (TCNs), Echo State Networks (ESNs), Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) have been applied to predict air pollution data, such as PM2.5, PM10, NO2, O3, SO2, and CO2 concentrations.
3. Stock Market Price Prediction (Univariate, Short-term): Various deep learning models, including Convolutional Neural Networks (CNNs), LSTMs, GRUs, and Generative Adversarial Networks (GANs), have been used to forecast stock market indices, shares, and prices.

With the focus of the study, I give below a few examples of time series forecasting using deep learning in the industrial process concept that was found in the literature (multivariate, short-term):

4. Forecasting production rates of oil fields [23];
5. Forecasting the remaining useful life (RUL) of an airplane turbofan engine [24], [25]
6. Forecasting fuel cells' durability [26];
7. Optimizing the exploitation of process off-gases in an integrated steel work [27]

2.3 Large Language Models (LLMs) and Domain-Specific LLMs

2.3.1 Examining the Use of LLMs in Technical Domains

Natural language processing techniques have witnessed considerable advancements in the last decades, starting with rule-based systems in the 1950s and 1960s, evolving to statistical techniques in the 1990s, and being affected by the development of neural networks in the 2010s. Lately, this has been due to the development of self-attention mechanisms and transformer-based neural networks [28]. Studies have found that training the model and scaling up the data can improve the model, based on the scaling law [29], which eventually leads to the generation of Large Language Models (LLMs) with emergent capabilities that were not found in smaller models [30].

Large Language Models (LLMs) are task-agnostic and used in various applications [31]. However, LLMs may encounter many challenges when facing domain-specific complex problems [31]. These challenges originate from the divergence of domain data, the intricacy of domain knowledge, distinctive domain intents, and various constraints such as social norms, cultural conformity, and ethical standards [31]. Domain-specific LLMs can be developed to address these challenges by modifying non-domain-specific LLMs employing distinct methodologies.

One study [31] analyzed the methodologies of domain specialization, categorized into three groups based on the accessibility of LLMs.:

- External augmentation (black box): adding capabilities to the LLM that were not available before, by retrieving specific missing knowledge from an external source to the input data. For example, Retrieval-Augmented Generation, also known as the RAG library, enables LLMs to call domain-specific tools [31]. For instance, LLMs can create code for Python interpreters or database queries by calling specific tools [31]. The black box assumption usually indicates we only have access to the model API (e.g., ChatGPT) without knowing any information about the generated output.
- Prompt engineering (grey-box): Customizing LLMs through task-specific input texts (prompts) to produce desired responses, including discrete and continuous prompts [31]. With discrete prompts, we can have a one-shot discrete prompt, where we tailor one discrete prompt until we are satisfied with the output results, and a few-shot discrete prompt, where we give our preferred answer in several discrete examples inside the prompt. Another approach suggests that instead of manually designing text instructions (discrete prompts), continuous prompting employs learnable vectors (soft prompts) attached to the input sentence. The grey box assumption means we have developer access to the LLM and information on the process. Such information can guide us in designing and refining prompts that effectively elicit domain knowledge.
- Model fine-tuning (white-box): Modifying the internal parameters of the LLM using smaller, domain-specific datasets, often through adapter-based or task-oriented methods [31]. The white box assumption implies that we have complete access to the LLM (e.g., LLaMA and its variants), encompassing the parameter settings, training data, and the model architecture.

With the rise of LLMs and the techniques developed to modify them to specific needs, literature shows that a variety of different industries have found ways to utilize LLM successfully by modifying them to their specific needs. The following are selected examples:

- Health and Medicine: Nuance Dragon Medical One employs LLMs to transcribe doctor-patient conversations with up to 98% accuracy, leading to a 30-40% reduction in documentation time [31].
- Automotive Industry: In legal document analysis (regulatory compliance), Mercedes-Benz employs LLMs to analyze regulatory changes related to vehicle safety and emissions, helping streamline compliance reporting and reducing associated costs by 10-15% [11].
- E-commerce: Customer Service and Support: Amazon uses LLM-powered chatbots to manage and process customer inquiries, successfully resolving 70-80% of queries without human intervention [11]
- Education: In a personalized learning application, Duolingo integrates LLMs to enhance learning and user engagement, reporting a 20% improvement in these areas [11].
- Finance and Banking: In fraud detection, JPMorgan Chase uses LLMs to monitor transactional data, contributing to a 40% reduction in fraud chances [11]
- Manufacturing Industry:
    - Knowledge sharing: This academic paper, [32], Examines the application of Large Language Models (LLMs) to augment knowledge sharing. in manufacturing environments. The authors developed an LLM-based system designed to retrieve information from factory documentation and expert knowledge, aiming to efficiently answer operator queries and facilitate new knowledge capture. Through a user study, they identified benefits like quicker information retrieval and issue resolution yet also observed a preference for human expert interaction. Furthermore, benchmarking various LLMs revealed GPT-4 as the top performer, with open-source models offering good alternatives due to data privacy and customization benefits. The study provides initial insights and a system blueprint for factories considering LLM integration into their knowledge management strategies. This study used a combination of RAG and prompt engineering to modify the LLM model.

- Controlling the production using LLM and digital twin: This academic paper [33] explores a framework that integrates Large Language Models (LLMs) with digital twin systems to enhance industrial automation, aiming for more flexible and adaptive production processes in smart factories. The authors propose using LLM agents, acting as intelligent managers and operators, to interpret complex information from digital twins and control physical production systems through service interfaces. By employing prompt engineering, these LLM agents can autonomously plan and execute diverse tasks, even un-predefined ones, showcasing the potential for human-like problem-solving in manufacturing. The research highlights the value of digital twins in bridging the gap between LLMs' reasoning capabilities and real-world industrial environments, enabling an artificial "brain" with mechatronic "hands" and "eyes." This study used a combination of RAG and prompt engineering to modify the Multi-agent LLM model.
- Mining Industry - Domain-Specific Question Answering: MiningGPT [34], a 7B parameter LLM, was specifically fine-tuned for the mining industry. This model was built by fine-tuning an instruction-tuned foundational model using a question-answer pair dataset generated from the MiningPile dataset. MiningPile is a comprehensive mining domain-specific open dataset comprising 167,857 rows and 120.6 million tokens derived from open data, books, and thesis reports. This study uses the fine-tuning methodology to modify the LLM.
- Root Cause Analysis for Industrial Process Anomalies: This document [35] outlines a novel approach to root cause analysis for industrial process anomalies, focusing on manufacturing safety. The core idea involves integrating knowledge graphs (KGs) with large language models (LLMs) to process the numerous alarms generated during industrial incidents. A knowledge graph, built from historical operational data, stores industrial safety facts. An LLM obtains named entities from real-time alarms and

reclaims relevant knowledge from the graph. This context allows the LLM to understand root causes and provide clear and accurate explanations.

LLMs are showing impressive capabilities in natural language tasks[36]; however, some challenges are described regarding LLMs' ability to interact with numerical figures. This research paper [36], "Exposing Numeracy Gaps: A Benchmark to Evaluate Fundamental Numerical Abilities in Large Language Models," introduces NumericBench, a new benchmark designed to assess the often-overlooked numerical reasoning capabilities of Large Language Models (LLMs). While LLMs excel at natural language tasks, the authors in [36] highlight their surprising weaknesses in fundamental numerical operations like arithmetic, comparison, and retrieval, attributing this to their reliance on "surface-level statistical patterns" rather than a true understanding of numbers as "continuous magnitudes.", the paper reveals that even advanced LLMs consistently "struggle with simple numerical tasks." The paper concludes by identifying key shortcomings, such as tokenizer and training paradigm limitations, and proposes future directions to support numerically aware language modeling, a necessity for real-world applications.

2.3.2 Gaps in Current Deep Learning and Domain-Specific GPTs Methods

The literature review of both deep learning and LLMs has pointed out significant successes across a variety of industries. While deep learning's main contribution is in comprehending complex, multivariate, nonlinear, numeric time series, the contribution of LLMs is in textual knowledge management. While deep learning has been tested on many different data sets from various domains (for example, weather, stocks, electricity, etc.) for multivariate, univariate, long, and short-term time series forecasting, there is a gap in testing them on very complex real-world industrial sensor data sets, at least in published literature. While you can draw lines that suggest

similarities between the published datasets and industrial process datasets, they are not inherently the same. The industrial data set, as tested in this paper, is a dataset of real-world operation of 31 sensors for 3 years, 1-minute interval, labeled and complete. Based on the best knowledge of the author, there are no published industrial data sets of this kind within the process industry context. Testing deep learning techniques on a process industry dataset of this kind can give novel insights into the capabilities of using deep learning in real-world applications. Regarding LLMs, real-world applications in many industries are available, like finance, e-commerce, and banking. However, real-world example applications within the process or manufacturing industry are underrepresented and elusive in current literature. Available literature shows applicability in using LLMs for information sharing or root cause analysis in a manufacturing environment.

In summary, the paper contributes to the following matters:

- Testing a deep learning forecasting model on real-world timeseries, multivariate, and complex sensory data from the process industry.
- A novel approach of integrating a deep learning time series forecaster with an LLM for actionable insights.

3 Methodology

3.1 Research Design

This study uses a quasi-experimental case study research design.

The overarching goal of this study is to research the value of the architecture of a deep neural network and a domain-specific GPT in forecasting and giving insights (root cause analysis and mitigation plan) into unwanted events based on a real-world, multivariate, non-stationary industrial process dataset extracted from process sensors. Using a deep learning architecture, the study will try to discover temporal relationships in historical measurements that can affect forecasting the target. This data will be taken from a specific real-world operating process, from the industrial process industry. After doing so, the chosen model's performance evaluation will occur. So, the modeling and analysis are done on a specific dataset**.** The research will include a downstream reasoning component (Large Language Model) that will be the interpretive layer of the forecasting model. That adds an interpretation layer using domain knowledge. The LLM will have more domain knowledge added to it and will be evaluated on its response to each level of expertise added.

3.2 Data Source Confidentiality and Preprocessing

3.2.1 Confidentiality

The data used in this research originates from a real-world, large-scale industrial process. While real-world datasets are highly valuable, they often come with trade-offs due to confidentiality constraints. To ensure anonymity, the data owner applied complete masking. All sensor name tags were removed, and a numerical transformation function was applied to the measurements before the data left the company. As a result, the numerical values presented in this study do not reflect actual values, and any resemblance to real figures is purely coincidental. However, to preserve the

dataset's relevance to real-world operations, the transformation function retained the fundamental patterns in the data. Consequently, the conclusions drawn from this research remain generally applicable to real-world scenarios.

3.3 Preprocessing Sensors Data

3.3.1 Scale Of the Complete Data Set

The dataset comprises historical measurements of 31 sensors, measuring 3.5 years (42 months) and a measurement resolution of 1 min. Totaling 1,843,199 chronological time steps, measures for each sensor. The whole data set consists of 57,139,169 measurements.

Sensor 1 is the forecasting target, while sensors 2-30 are the historical measurements used in the model to forecast sensor 1. Eventually, only a small portion of this vast data set was used for the research due to computational constraints.

3.3.2 Note on Feature Engineering

The preprocessing is minimal, and its goal is to process the data to a point that it is ready for deep learning modeling and not for better performance. Practically, it is safe to say that no feature engineering was done. In the context of real-world applicability, using an out-of-the-box dataset without the need to feature engineer it can be very valuable and more scale-safe. However, using feature engineering on a similar data set could improve forecasting performance.

3.3.3 Raw Data

Raw data from the sensors is composed of 4 CSV (Comma Separated Values) files, each for a different year: 2022, 2023, 2024, and January-June of 2025.

3.3.4 Handling Text

As this is a real-world data set, the data is included with real sensor missing measurements, such as loss of communication, maintenance, time-outs, etc. Those were handled by attaching the value NaN (Not a Number) [37]. It is a special floating-point value defined by the IEEE 754 standard. The model must have a value to work, so to handle this, there are two main approaches: removing the whole timestamp from the dataset. The second approach is imputation. This involves replacing missing values with estimated ones based on existing data. There are several imputations: forward fill/backward fill, interpolation, rolling mean/median. The chosen type is Forward Fill/Backward Fill: Replacing NaNs with the immediately preceding or succeeding non-missing value.

Table 3-1 - Text count and percentage in the dataset per year

Year	Text Values Count	Text Values %
2022	374,870	2.23%
2023	109,333	0.65%
2024	42,532	0.25%
2025	28,926	0.34%
All Years	555,661	0.94%

Table 3-2 - Types of texts and their count in the dataset

Text Value	2022	2023	2024	2025	All Years
Bad	311	2,265	182	29	2,787
Comm Fail	193,724	80,560	20,019	0	294,303
Configure	311	0	0	0	311
I/O Timeout	59,986	3,605	5,072	28503	97,166
Intf Shut	110,674	22,903	17,259	394	151,230
Not Connect	9,864	0	0	0	9,864

3.3.5 Handling “0”

Zeros were not eliminated or impaled. In the real world, zero results can suggest that a unit is not working, or the value is below the measurement sensitivity. This does not necessarily indicate a failure in the analyzer.

Table 3-3 - Non-zeros and zeros in the dataset

Year	Non-Zero Numeric Count	Non-Zero Numeric %	Zero Numeric Count	Zero Numeric %
2022	15,052,552	89.50%	866,147	5.15%
2023	14,439,749	85.85%	1,744,487	10.37%
2024	15,163,774	89.91%	1,131,903	6.71%
2025	8,015,922	94.56%	167,021	1.97%
All Years	52,672,087	89.30%	3,909,561	6.63%

3.3.6 Timestamps

In time series data, daylight saving times can create doubled time steps and gaps between time steps. This happens twice each year. There is a gap in the Timestamp when daylight saving time requests to move the clock forward by 1 hour. So, we can see a gap of 1 hour each year for daylight savings in our timeseries, for example, if the next time stamp after 04:59 am is becoming 06:00 instead of 05:00. The same happened when we moved our clock backward 1 hour, but here we get a double timestamp. For example, the next time stamp after 04:59 am is 04:00 instead of 05:00.
To handle this, the dataset was converted from the local time zone to UTC (Coordinated Universal Time).

Table 3-4 - Timestamps count in the dataset

Year	Expected Frequency	Expected Timestamps	Actual Timestamps	Missing Timestamps	Duplicate Timestamps
2022	1.00 min	525,600	525,600	0	0
2023	1.00 min	525,600	525,600	0	0
2024	1.00 min	527,040	527,040	0	0
2025(Jan-Jul 3 23:59)	1.00 min	264,959	264,959	0	0
All Years	1.00 min	1,843,199	1,843,199	0	0

3.3.7 Data Completeness

The data set includes 1,843,199 timestamps per sensor. The data set has 31 sensors, totaling 57,139,169 measurements. Data completeness can be calculated using the following formula:

$$\text{DataCompleteness}(\%) = \frac{N_{usable}}{N_{total}}$$

Where:

- N_{usable} = Number of usable observations
- N_{total} = Total number of observations

For this study, usable observation is a numeric value (meaning it is not a text; Zeros are included as usable observation). Using the data extracted from the dataset, the following table shows the completeness of the dataset per year.

Table 3-5 - Dataset quality

Year	Ready for model
2022	97.7%
2023	99.3%
2024	99.7%
2025	99.6%
All Years	99.0%

This is considered a high-quality dataset by industry standards.

A full breakdown of texts, zeros, timestamps, and numeric values can be seen in Appendix A.

3.4 Deep Learning Model

3.4.1 Choosing a Model

Deep learning for time series is a relatively new concept. Many new models are being suggested, each with its strengths and weaknesses. Several studies have tried to compare the models [9], [20], [21], [22], [38], each with its point of view. Using the comparative studies, the chosen model was picked based on the following principles:

- Length of the prediction: If the application is required to forecast more than 1 step, then some models cannot do so. In industrial processes, decisions are made by operators and management, and this preferably requires time to act. For example, suppose the data resolution is 1 minute, and the need is to forecast a variable exceedance directly affected by an upstream process. In that case, a 1-minute alert might not be sufficient to act unless the facility has an automatic computerized response based on the alarm. This study presents a more agile model that can forecast more than 1 step. This is the reason this study is focusing on a multi-step ahead model.

- Multivariate or univariate: Univariate time series forecasting involves predicting the future values of a single variable based on its historical data. It focuses solely on the patterns and trends within that one time series, without considering any other variables. This contrasts with multivariate time series forecasting, which uses multiple related time series to make predictions. The study aims to predict a future value based on historical upstream multivariate data.
- Forecasting length: How many steps should the model predict in the future? A multistep short-term forecast is considered anything between 2 and 48 steps ahead [9]; anything above it would be viewed as a long-term forecast. It is, of course, based on one's specific needs or application and how many steps ahead he wants or needs to forecast. For example, assuming the time is 10:00 am, forecasting the temperature in Boston, Massachusetts, at 10:00 am 7 days from now requires 168 steps, which is a clear long-term forecast. The data owner did not specify a forecasting window in our case study, so a 12-minute ahead forecast was chosen based on a hyperparameter optimization study. But in real-world applications, things are more complex. Twelve minutes forecast might be enough for the current process engineer, but not enough for the next one in his role; he might want 30 30-minute forecast. This type of change might require an architectural change in the model, which is why this study preferred to run the model on a long-term-based architecture, giving enough agility to make changes in the future.

Let's summarize our type of model: Forecasting Long-Term Multi-Variate Time-Series. Based on the work done by [9], transformers are the only true long-term forecasting architecture. Transformers excel in long-term forecasting by overcoming the short-term memory limitations of recurrent neural networks. Unlike other architectures where information must pass through many steps sequentially, Transformers utilize attention mechanisms to capture global dependencies between distant samples in a sequence directly. This allows them to maintain crucial information over long periods, making them more effective for extended predictions. Based on the table below,

all transformers can analyze multivariate datasets. Risk management is another crucial aspect, specifically in real-world operations like the process industry. No one, yet can predict the future with perfect accuracy, so all current forecasting models are, by definition, wrong, but this does not mean that they are not helpful. In the process industry, the risk of wrong prediction must be considered when deciding. For example, if one wants to forecast a value that, if it exceeds, he must reduce the production rate to prevent it from exceeding this value, then, to decide, one needs to know the probability of this event. Is it 90% certainty, or 20%? It can affect the decision. [20] shows in Table 1 which models are point-type forecasting (just one number) and which give a probability estimate. From the transformers list, only three are probabilistic: temporal fusion transformer (TFT), TACTiS, Scaleformer, and Infomaxformer, as shown in Table 3-6, which is taken from [20]

Why I Chose TFT Over TACTiS, Scaleformer, and Infomaxformer for Industrial Use:

In industrial time series forecasting, accuracy alone is insufficient. Real-world, large-scale decisions have profound financial, environmental, safety, and quality implications. Decision-makers carefully evaluate these implications before acting and therefore demand clear explanations behind numerical predictions. Merely stating, “this is what the model predicts,” even from a highly accurate model, is inadequate. Explainability is essential. While TACTiS, Scaleformer, and Infomaxformer offer advanced architectures tailored for long-term forecasting, they fall short in transparency—an indispensable requirement for industrial decision-making.

The Temporal Fusion Transformer (TFT) was selected because it is a novel attention-based deep learning architecture designed for multi-horizon time series forecasting. TFT uniquely combines strong performance with interpretability, providing insight into the temporal dynamics driving predictions.

Key advantages of TFT include:

- Built-in Interpretability: TFT's architecture supports explainability through specialized components. These components use variable selection networks to identify the most impactful input features at each time step. Additionally, the modified interpretable multi-head attention mechanism highlights relevant time points. This allows end-users to pinpoint precisely what inputs drive predictions, unlike traditional post-hoc methods, which disregard time ordering. TFT integrates interpretability directly into its design.
- Handling Heterogeneous Input Types (Static and Temporal Features): Multi-horizon forecasting involves diverse inputs - static covariates (unit tags, status, product numbers), known future inputs (holidays, scheduled maintenance), and historical exogenous series. Unlike many models that inadequately handle such heterogeneity, TFT explicitly integrates these varied inputs. Its static covariate encoders generate context vectors to condition temporal dynamics. A sequence-to-sequence layer locally processes known and observed inputs, while the temporal self-attention decoder captures long-term dependencies. This careful handling of input diversity effectively mirrors real-world complexity.
- Enhanced Trust and Debugging Capabilities: The inherent interpretability of TFT fosters trust and accountability. By enabling predictions to be audited and traced to specific features or time points, TFT addresses the challenges posed by opaque "black box" models. Its insights - such as global feature importance, persistent temporal patterns, and significant event identification - empower human experts to verify and improve forecasts. Understanding model behavior helps refine input pipelines and assumptions, particularly when performance adjustments are required.

Table 3-6 DTSF Model Architecture Paradigm [20]

Architecture	Model	Multi/Uni	Output	Loss	Metrics	Year
Encoder - Decoder	COST (Woo et al., 2022)	Multi & Uni	Point	contrastive loss	MSE, MAE	2022
	TS2Vec (Yue et al., 2022)	Multi & Uni	Point	contrastive loss	MSE	2022
	ACT (Li et al., 2022c)	Multi & Uni	Point	cross-entropy	Q50 loss, Q90 loss	2022
	SimTS (Zheng et al., 2023)	Multi & Uni	Point	cos-similarity loss, InfoNCEloss	MAE, MSE	2023
	DeepTCN (Chen et al., 2020b)	Multi	Pro	quantile loss	NRMSE, SMAPE, MASE	2020
	STEP (Shao et al., 2022)	Multi	Pro	MAE	MAE, RMSE, MAPE	2022
	DCAN (He et al., 2022)	Multi	Point	RMSE	MAE, RMSE	2022
	FusFormer (Yang and Lu, 2022)	Multi	Point	–	RMSE, RMSE Decrease	2022
	HANet (Bi et al., 2023)	Multi	Point	–	MAE, RMSE	2022
	D^3VAE (Li et al., 2022b)	Multi	Pro	–	MSE, CRPS	2022
	TI-MAE (Li et al., 2023b)	Multi	Point	MSE	MSE, MAE	2023
	AST (Wu et al., 2020b)	Uni	Pro	cross-entropy	Q50, Q90 loss	2020
Transformer	TFT (Lim et al., 2021)	Multi & Uni	Prob	quantile loss	P50, P90 quantile loss	2021
	Informer (Zhou et al., 2021)	Multi & Uni	Point	MSELoss	MSE, MAE	2021
	ETSformer (Woo et al., 2022b)	Multi & Uni	Point	MSELoss	MSE, MAE	2022
	FEDformer (Zhou et al., 2022b)	Multi & Uni	Point	MSELoss	MSE, MAE, Permutation	2022
	TACTiS (Drouin et al., 2022)	Multi & Uni	Pro	log-likelihood	CRPS-Sum, CRPS-means	2022
	Autoformer (Wu et al., 2021)	Multi & Uni	Point	L2 loss	MSE, MAE	2022
	NSTformer (Liu et al., 2022b)	Multi & Uni	Point	L2 loss	MSE, MAE	2023
	Dateformer (Young et al., 2022)	Multi & Uni	Point	MSE	MSE, MAE	2023
	Crossformer (Zhang and Yan, 2023)	Multi & Uni	Point	MSE	MSE, MAE	2023
	Scaleformer (Shabani et al., 2022)	Multi & Uni	Pro	MSE	MSE, MAE	2023
	BasisFormer (Ni et al., 2023)	Multi & Uni	Point	MSE	MSE, MAE	2023
	CRT (Zhang et al., 2022a)	Multi	Point	–	ROC-AUC, F1-Score	2021
	Pyraformer (Liu et al., 2021)	Multi	Point	MSE	MSE, MAE	2022
	TDformer (Zhang et al., 2022b)	Multi	Point	MSE	MSE, MAE	2022
	FusFormer (Yang and Lu, 2022)	Multi	Point	–	RMSE, RMSE Decrease	2022
	Scaleformer (Shabani et al., 2022)	Multi	Point	MSE, Huber, Adaptive loss	MSE, MAE	2022
	Infomaxformer (Tang and Zhang,2023)	Multi	Pro	MSELoss	MSE, MAE	2023
	PatchTST (Nie et al., 2022)	Multi	Point	Adaptive Loss	MSE, MAE	2023
	iTransformer (Liu et al., 2023c)	Multi	Point	L2 Loss	MSE, MAE	2023
	MCformer (Han et al., 2024)	Multi	Point	MSE, MAE	MSE, MAE	2024
	SAMformer (Ilbert et al., 2024)	Multi	Point	MSE	MSE, MAE	2024
	TSLANet (Eldele et al., 2024)	Multi	Point	MSE	MSE, MAE	2024
	MASTER (Li et al., 2024a)	Multi	Point	MSE	IC, ICIR, RankIC	2024
	TimeSiam (Dong et al., 2024)	Multi	Point	L2, Cross-Entropy	MSE, MAE, Recall, F1 Score	2024
	Chronos (Ansari et al., 2024)	Multi	Point	Cross Entropy	WQL, CRPS, MASE	2024
	TimeXer (Wang et al., 2024c)	Multi	Point	L2 loss	MSE, MAE	2024
	Time-SSM (Hu et al., 2024)	Multi	Point	MSE	MSE, MAE	2024
	SageFormer (Zhang et al., 2024)	Multi	Point	MSE	MSE, MAE	2024
	TIME-LLM (Wang et al., 2024a)	Multi	Point	MSE, SMAPE	MSE, MAE, SMAPE	2024
	CARD (Wang et al., 2024b)	Multi	Point	MSE, MAE	MSE, MAE	2024
	Pathformer (Chen et al., 2024)	Uni	Pro	L1 loss	MSE, MAE	2024
GAN	ForGAN (Koochali et al., 2019)	Multi & Uni	Pro	RMSE	MAE, MAPE, RMSE	2019
	COSCI-GAN (Seyfi et al., 2022)	Multi & Uni	Pro	Global loss = local + central	MAE	2022
	RCGAN (Esteban et al., 2017)	Multi	Pro	cross-entropy	AUROC, AUPRC	2017
	TimeGAN (Yoon et al., 2019)	Multi	Pro	Unsupervised, Supervised, Reconstruction, Loss	Discriminative and Predictive Score	2019
	PSA-GAN (Jeha et al., 2022)	Multi	Point	Wasserstein loss	–	2022
	AEC-GAN (Wang et al., 2023)	Multi	Point	MSE	ACF, Skew / Kurt, FD	2023
	ITF-GAN (Klopries and Schwung, 2024)	Multi	Point	MSE	MSE, STS, Pearson, Hellinger, Pred.	2024
	MAGAN (Ferchichi et al., 2024)	Multi	Point	–	MAE, MAPE	2024
	TSGAN (Xu et al., 2024)	Multi	Point	–	MAE, RMSE, MAPE	2022
	AST (Wu et al., 2020b)	Uni	Pro	cross-entropy	Q50 loss, Q90 loss	2020
Integrated Module	ConvLSTM (Shi et al., 2015)	Multi & Uni	Point	cross-entropy	Rainfall-MSE, CSI, FAR, POD	2015
	Bi-LSTM (Du et al., 2020)	Multi	Point	MSE	MAE, RMSE	2020
	(Fu et al., 2022a)	Multi	Point	MAE	MAE, RMSE, MAPE	2022
	(Asiful et al., 2018)	Uni	Point	L2 loss	MAE, MSE, MAPE	2018
	TATCN (Wang and Zhang, 2022)	Uni	Point	–	MAE, RMSE, sMAPE	2022
	LST-TCN (Sheng et al., 2022)	Uni	Point	Pinball loss	MAPE, RMSE	2022
Cascade	TimesNet (Wu et al., 2022)	Multi & Uni	Point	MSE, SMAPE	MSE, MAE, SMAPE, MASE	2022
	TreeDRNeT (Zhou et al., 2022c)	Multi & Uni	Point	Lp Regularized Loss	MSE, MAE	2022
	Triformer (Cirstea et al., 2022)	Multi & Uni	Point	–	MSE, MAE	2022
	SCINet (Liu et al., 2022a)	Multi & Uni	Point	L1 loss	RSE, CORR, MSE, MAE, MAPE, RMSE	2022
	HTSF (Duan et al., 2023)	Multi	Pro	L2 loss, HyperGRU	MAE, RMSE	2023
	CIPM (Yolcu and Yolcu, 2023)	Multi	Point	–	RMSE, MAPE, MdRAE	2023
	MACN (He et al., 2023)	Multi	Point	RMSE	RMSE, MAE	2023
	CasCIFF (Zhu et al., 2024)	Multi	Point	–	MSLE, MAPE	2024
	FCPM (Guo et al., 2024)	Multi	Point	RMSE	MAE	2024
	N-BEATS (Oreshkin et al., 2019)	Uni	Point	MAE	SMAPE, OWA, MASE	2020

3.4.2 Temporal Fusion Transformer and Basic Definitions

To forecast our target value, a Temporal Fusion Transformer (TFT) [39] model from the Darts library [40] was selected for its ability to handle multivariate time series with temporal attention and uncertainty estimation. Based on a hyperparameter optimization study, the model was configured to receive 96 minutes of historical sensor data as input and predict the next 12 minutes of values (Sensor 1) as output. All available sensor streams served as past covariates. Scaling the numbers is crucial in time series forecasting when using deep learning. It ensures model stability and performance, particularly when dealing with variables of varying magnitudes [22].

Following the Temporal Fusion Transformer (TFT)[39] paper, this implementation applies z-score normalization to both the target and covariate time series. Specifically, each measurement **x** is transformed using the formula:

$$x_{(\text{scaled})} = \frac{x - \mu}{\sigma}$$

Where **μ** and **σ** are the mean and standard deviation computed from the training set, this standardization is crucial for stabilizing the learning process and ensuring consistent feature scaling across the network. The Scaler class from the Darts library is used to perform this operation, and the same fitted statistics are applied to the validation and test sets, as described in the TFT paper. This preprocessing step ensures compatibility with the model's requirements.

One of TFT's novelties is the ability to provide new types of interpretabilities for multi-horizon forecasting. Multi-horizon forecasting predicts multiple time steps in the future, compared to one-step-ahead forecasts, which only forecast one step into the future each time. The authors [39] provide the following list of interpretability options in the TFT model: "(i) globally-important variables for the prediction problem, (ii) persistent temporal patterns, and (iii) significant events."

By default, the model also uses Quantile Regression likelihood, which means that its forecasts are probabilistic, an essential feature for interpretability.

Below is a list of the inputs to the model and definitions for the different inputs:

- Target Variable: the future variable we want to forecast.
- We use covariates to analyze their interdependency to predict the target variable.
- Input_chunk_length: number of historical steps provided to the model for forecasting.
- Output_chunk_length: number of future steps to forecast.
- Quantile regression [0.1, 0.5, 0.9] - The quantiles [0.1, 0.5, 0.9] represent uncertainty intervals in the model's probabilistic forecasts using quantile regression.
    - 0.1 Quantile (10^{th} percentile): This is the value below which 10% of predicted outcomes fall. Represents a lower bound of your uncertainty interval. In practice, 90% of future outcomes are expected to exceed this value.
    - 0.5 Quantile (50^{th} percentile or median): This is the most likely (central) forecast. Half the outcomes are expected to fall above, and half below.
    - 0.9 Quantile (90^{th} percentile): This is the value below which 90% of outcomes fall. Represents the upper bound of your uncertainty interval. Only 10% of outcomes are expected to exceed this value.
- Timestamp alignment: Ensuring every time point (e.g., every minute) has a consistent and valid timestamp.
- Fill missing timestamps: filling any missing timestamps with forward and backward fill.
- Float32 conversion: Converting all numerical values (e.g., sensor readings) to 32-bit floating-point format (float32). Some devices (e.g., MPS on macOS) and deep learning libraries only support float32.
- Scaling: Standard practice to speed up convergence and improve model stability, especially with different sensor scales.

- Hidden size: Size of vectors used in self-attention layers. Determines the capacity of the representation [28].
- LSTM layers: The number of stacked Long Short-Term Memory (LSTM) layers in a model. Each layer processes temporal sequences and passes its output to the next layer to capture increasingly complex patterns.
- Attention heads: number of discrete attention mechanisms in multi-head attention [28].
- Dropout: fraction of neurons dropped during training to reduce overfitting.
- Batch size: number of training examples processed before updating model weights.
- Epochs: One epoch is a complete pass through the entire training dataset. The model's parameters are updated after each batch within the epoch.
- Learning rate: A hyperparameter that controls the step size at each iteration while moving toward a minimum of the loss function during optimization.
- Optimizer: The algorithm that updates model weights during training based on gradients computed from the loss function. Standard optimizers include SGD, Adam [41], and RMSProp.
- Likelihood: the assumed probabilistic distribution of the model's output. In forecasting, this defines how the model predicts and evaluates uncertainty. Quantile Regression allows the model to predict multiple quantiles (e.g., 0.1, 0.5, 0.9) for uncertainty-aware forecasts, instead of just point estimates.
- Early stopping: A training strategy that stops model training when performance on a validation set stops improving for a defined number of epochs.
- Gradient clipping: A technique that limits (or "clips") the gradients during backpropagation to a maximum absolute value (e.g., 0.5) to avoid exploding gradients.
- Add_relative_index = False: Disables numeric indexing of data points, meaning the model won't explicitly use their positional indices.

- Add_encoders = {"cyclic": {"future": ["month"]}}: Applies cyclical encoding specifically to the “month” feature for future periods, helping the model capture seasonal patterns.

3.4.3 Hyperparameter Optimization Study

In this study, I conducted hyperparameter optimization using Optuna, a Python-based automated optimization framework, to identify the optimal settings for a Temporal Fusion Transformer (TFT) model applied to time series forecasting. The study was done on a Google Colab Cloud NVIDIA A100 GPU.

The hyperparameter search space included:

- Input sequence length (input_chunk_length): [24, 48, 96]
- Output forecast length (output_chunk_length): [12, 24, 30]
- Hidden layer sizes (hidden_size): [10, 20, 40, 80, 160, 240, 320]
- Number of LSTM layers (lstm_layers): [1, 2]
- Attention heads (num_attention_heads): [1, 4]
- Dropout rates (dropout): [0.1, 0.2, 0.3, 0.4, 0.5, 0.7, 0.9]
- Training epochs (n_epochs): [5, 10]
- Batch sizes (batch_size): [64, 128, 256]
- Learning rates (optimizer_kwargs): [{‘lr’: 0.0001}, {‘lr’: 0.001}, {‘lr’: 0.01}]
- Gradient clipping values (gradient_clip_val): [0.01, 1.0, 100.0] *

*Note: Although gradient clipping was part of the hyperparameter search space, it was not applied during training due to limitations in the Darts model implementation, which did not support passing gradient_clip_val through the configuration parameters. As a result, gradient clipping was not evaluated for its impact on model performance.

Each trial within the Optuna optimization process tested a distinct combination of these parameters. The TFT model was trained using scaled training data and evaluated using scaled validation data. The performance was assessed using the Mean Quantile Loss (Pinball Loss) metric, which is relevant for probabilistic forecasts. Specifically, the model was trained with a fixed random seed (random_state=42) for reproducibility.

After conducting 50 trials or reaching a 2-hour runtime limit, with a total of 5 trials completed, the optimal hyperparameter combination identified was:

- input_chunk_length: 96
- output_chunk_length: 12
- hidden_size: 40
- lstm_layers: 1
- num_attention_heads: 4
- dropout: 0.7
- n_epochs: 5 (100 epochs were used for the training)
- batch_size: 256
- learning rate: 0.0001

Other inputs to the tuning model:

- Target Variable: Sensor 1
- Past Covariates: All other sensor readings (Sensor 2-31).
- Add_relative_index = False
- Add_encoders = {“cyclic”: {“future”: [“month”]}}
- Quantiles = [0.01, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 0.99]

- Data set = train_raw.csv (See results chapter for details about train, validation, and test datasets)

3.4.4 Training Process

The training process was done on a CPU (MacBook Pro M3 16 GB unified memory)

3.4.4.1.1 Preprocessing

- Read train_raw, val_raw, test_raw .csv files for (See results chapter for details about those data sets)
- Convert to Darts Timeseries objects.
- Fill missing timestamps with forward and backward fill and convert types to float32.
- Scaling both target and covariates using Scaler (z-score).
- Load train_raw and val_raw CSV files and combine them for the training.

3.4.4.1.2 Model Instantiation

Using TFTModel from Darts with hyperparameters found in the optimization study:

- input_chunk_length: 96
- output_chunk_length: 12
- hidden_size: 40
- lstm_layers: 1
- num_attention_heads: 4
- dropout: 0.7
- n_epochs: 100
- batch_size: 256

- learning rate: 0.0001
- Target Variable: Sensor 1
- Past Covariates: All other sensor readings (Sensor 2-31).
- Add_relative_index = False
- Add_encoders = {“cyclic”: {“future”: [“month”]}}
- Quantiles = [0.01, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 0.99]
- Data set = combined and scaled train+val dataset.
- Optimizer = Adam (default)

3.4.4.1.3 Outputs

- A 100-epoch trained TFT model that predicts future values (Sensor 1) over a 12-minute horizon, based on 96 minutes of 30 past covariates.

3.5 Domain-Specific GPT

To augment the forecasting model with interpretable and actionable recommendations, a domain-specific Large Language Model (LLM) was configured to emulate the reasoning of a senior process engineer. This setup aimed to bridge the gap between quantitative time-series predictions and qualitative engineering decision-making.

3.5.1 System Prompt

A structured and role-specific system prompt was developed to define the LLM’s behavior. The prompt structure was created based on the GPT 4.1 prompting guide [42]. The following structure

should give the LLM a context overlaying the user query, potentially providing more helpful answers.

The following structure was used, as seen in the list below. The list explains the system prompts given for the LLM in this study. Using prompt engineering, a dedicated system prompt version was created for each new version of the LLM (when extra knowledge was added to the LLM, such as rag and professional domain textbooks. A Total of 6 different versions were used, one for each LLM version. For masking the relevant process behind the real-world data that is used in this study, parts of the actual text are omitted, but following the structure below should give similar results:

 # Role and Objective

What is the role of the LLM? (e.g., a senior process engineer who specialized in…) And what are his objectives (e.g., root-cause analysis, mitigation plan)?

 # Instructions

Detailed instructions on what needs to be done to comply with the objectives

Sub-categories for more detailed instructions

e.g., sub-category for root-cause and mitigation plan, an instruction to double-check his answers, and an instruction to always cite sources when providing answers.

 # Output Format

specific request for formatting (concise, professional language, etc.)

 # Examples

Past studies showed that giving the LLM examples of how a query and answer should look improves the LLM response and provides more control over the structure of the answer [43]

 # Context

Any critical information to give the LLM may help it comply with its objectives.

 # Final instructions and prompt to think step by step

Giving instructions to the LLM to think step by step significantly improves the LLM's answers [44].

3.5.2 User Message

The user message is a fixed query that the LLM gets immediately when it is called. Different user queries were used for each LLM version (like the system prompt methodology). The structure below resembles the one that was used in all the versions:

Hello Senior Process Engineer,

I am the facility operator, and I need your expertise. Our deep learning model has forecasted a threshold violation within the next 12 minutes. The attached CSV file provides the forecasted 12-minute multivariate time-series data and a 96-minute window preceding the breach.

CSV DATA:

{csv_content}

Please provide the following outputs:

1. Adjustment Plan:

- Determine the minimum reduction needed (across sensors 2, 6, 10, 14, 18, 22, and 26) to prevent Sensor 1 from exceeding the threshold as predicted by the model.
- Specify the start and end time for the reduction window.
- Explain your reasoning and calculation steps clearly.

2. Root Cause Analysis:

- Use the Five Whys technique to identify the most probable root cause for the violation.
- Provide a short explanation of your reasoning.

3. Corrective Actions

- Based on the identified root cause and your professional knowledge, suggest realistic and relevant corrective actions to prevent future occurrences.

3.5.3 Data Input

The LLM receives a multivariate time-series file in CSV format containing:

- The forecasted values for the next 12 minutes (Sensor 1)
- Past measurements for 96 minutes for sensors 1-31.

The model reads the file content, interprets the forecast trends, and generates expert insight accordingly.

3.5.4 Model Execution

The model is called using OpenAI's GPT-4.1 API (model gpt 4.1-2025-04-14) with the prepared system and user prompts. The key steps include:

- Loading the system prompt instructions
- Reading the user message and attaching the forecast CSV. The user message is a fixed prompt of an operator asking for help from the process engineer (the LLM), and a request to provide a set of outputs that is needed in real-time.
- Executing the LLM call using client.chat.completions.create() with the file embedded or referenced.

3.6 Hybrid Deep Learning – LLM Model Architecture

3.6.1 Passing The Forecasting Results from The TFT Model to the LLM

To create a real-world scenario, I assumed that the test data is a real-world data set that is entering the TFT and being analyzed by him. once the TFT model forecast a specific target value that exceeds a curtain threshold he automatically creates a CSV file with a “snapshot” of the forecast, meaning the 96 min leading to the value, and the 12 min sensor prediction. The LLM script loads this file from the folder and gives insights based on it (the file is given to the LLM in the user message).

3.6.2 Retrieval-Augmented Generation (RAG)

As part of the metrics evaluation, the LLM results will be compared by adding knowledge and comparing the results to the model with less knowledge. The review is done by the process SME (subject matter expert). To do so, a RAG library is built. In the RAG library, the following needs to be added. The full contents of the academic textbook and process cheat sheet are not displayed here to avoid potential privacy concerns:

.

- An academic textbook that explains the fundamental principles of the applied process.
- The Five Whys Technique [45] (also shown in Appendix B)
- Instrument Engineers' Handbook, Vol 1, Process Measurement [14]
- Process cheat sheet – a brief and high-level explanation about the specific industrial process being analyzed by the LLM.

3.7 Evaluation Metrics

For evaluating the model, I used two evaluation metrics – quantitative for the TFT and qualitative for the LLM. To comply with current literature for TFT, I used the same performance evaluation metric as the TFT in its original paper [39]: Mean-Quantile-Loss. For evaluation, lower is better. A detailed explanation of the metric is provided in Chapter 3.1.7.1. For the LLM, I used a qualitative metric with the help of a subject matter expert. The phenomenon that the model is exploring is not fully understood by the data owner, so using a quantitative method like BLEU and ROUGE makes less sense, because the way they are calculated is by comparing the LLM's response with a reference that is regarded as the "correct answer". In our case, there is no clear one correct now for "root cause analysis" or correct mitigation plan" or "flow rate reduction plan" – so using those techniques will not give us a good performance metric. Instead, I used a qualitative method. I asked the data owner to provide a subject matter expert (SME) for this phenomenon and asked him four direct qualitative questions, detailed in chapter 3.1.7.2.

3.7.1 Temporal Fusion Transformer

The TFT is a quantile forecaster. Meaning he predicts a number in a specific quantile. So, when the model gives you a result, you get a result in a particular percentile, for example, P90 = 3.78. A P90 means "I'm 90% sure the real value will be lower than 3.78." An example of quantile forecasting can be seen in Figure 3-1.

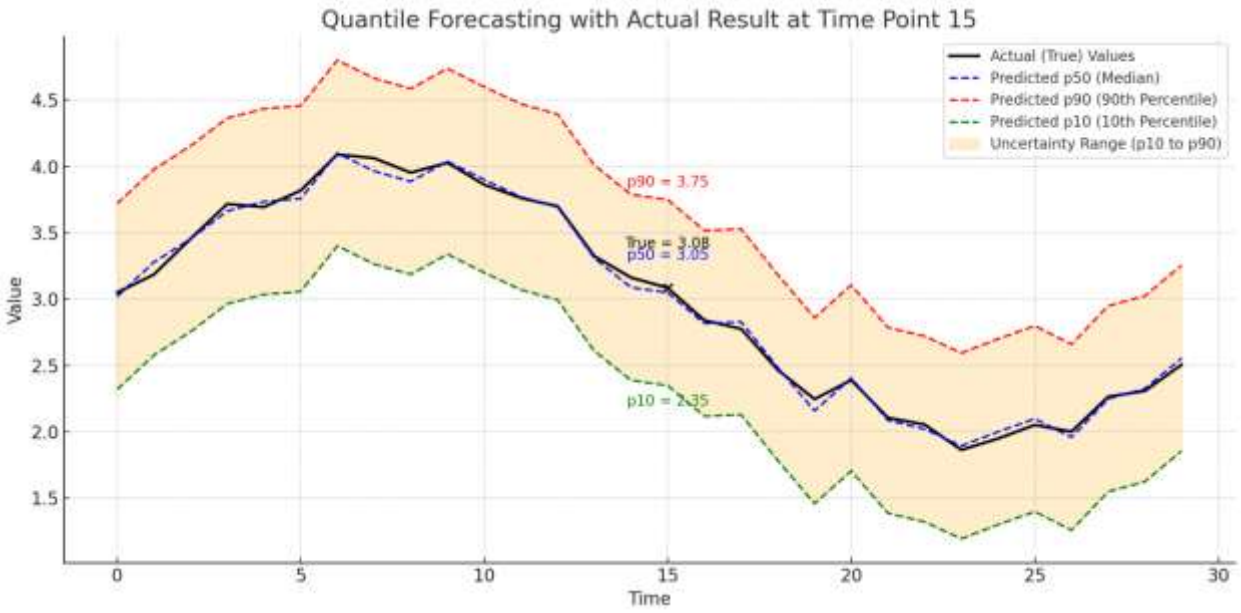


Figure 3-1 - Quantile Forecasting in a Time Series

We can see in Figure 3-1 that for a specific point in time, in the case of the figure, it is point 15, the P90 = 3.78, P50 = 3.08, and P10 = 2.38. P50 means the median, so the model thinks there is a 50% chance that the actual value will be higher than this number or lower. Meanwhile, P10 means that there is a 90% chance that the actual number will be higher than this.

To calculate the P90, P50, and P10, TFT uses the quantile loss formula during its training. Quantile loss penalizes the prediction based on how far it is from the actual value:

- If the guess is too low, it penalizes more when the quantile is high.
- If the guess is too high, it penalizes more when the quantile is low.

So, the quantile loss gives you an evaluation metric; it tells you how far you are from the actual value, but if the number is, let's say, 0.5 points away from the exact number, the quantile loss will not be exactly 0.5 because this number is penalized based on the quantile. A calculation example is seen in the following formula. The mean quantile loss is the average across the whole data set. It can be shown as MQL P50, MQL P90, or MQL for all quantiles.

Quantile Loss Formula:

$$L_q(y, \hat{y}) = \begin{cases} q \cdot (y - \hat{y}) if y \geq \hat{y} \\ (1 - q) \cdot (\hat{y} - y) if y < \hat{y} \end{cases}$$

- y The actual value
- $\widehat{y_q}$ is the predicted value for quantile q (e.g., 0.1, 0.5, 0.9)
- $q \in (0, 1)$ is the quantile level

Let's look at Figure 3. The examples below show how the quantile loss differs for P90, even when the distance equals 10.

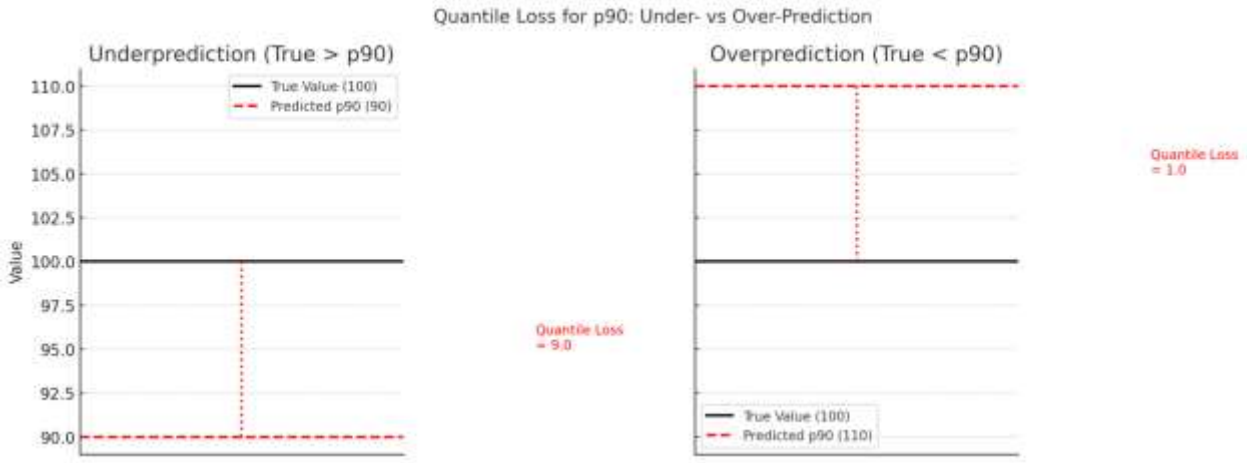


Figure 3-2 - Quantile Loss Example for P90

Example:

- Let's say the actual value is 100 (y). and the predicted number at the 0.9 percentile is 90 $\widehat{y_q}$
- $\widehat{y_q}$ is the expected value for quantile q (e.g., 0.1, 0.5, 0.9)
- $q \in (0, 1)$ is the quantile level
- The quantile level is equal to the percentile 0.9.
- Since y > $\widehat{y_q}$ we use the top line of the formula:

$$q \times (y-\widehat{y_q}) = 0.9 \times (100-90) = 9$$

Let's do the opposite:

- y = 100, $\widehat{y_q}$= 110 at the 0.9 percentile.
- Because y < $\widehat{y_q}$we use the bottom line of the formula:
-

$$(1-q) \times (\widehat{y_q}y) = (1-0.9) \times (110-100) = 1$$

Summary:

- Percentile results (P90, for example): the model predicts a number within a percentile. This is the forecast result.
- Quantile Loss: an evaluation metric that tells you how far the predicted result is from the actual result and penalizes the difference.
- Mean Quantile Loss (MQL): an evaluation metric, the average quantile loss across the whole data set.

3.7.2 Domain-Specific GPT

A subject matter expert (SME) conducted a qualitative evaluation of six response versions as listed in Table 3-7 to assess the performance of domain-specific GPT outputs. Each version was reviewed using the following criteria:

- Is it accurate?
- Is this valuable information?
- What are the strengths?
- What are the limitations?

Table 3-7 - Six LLM versions to be evaluated by the human SME

Version
NAÏVE
NAÏVE + Process Cheat Sheet
NAÏVE + Process Cheat Sheet + 5whys
NAÏVE + Process Cheat Sheet + 5whys + domain textbook
NAÏVE + Process Cheat Sheet + 5whys + domain textbook + IE textbooks
NAÏVE + Process Cheat Sheet + 5whys + domain textbook + IE textbook + deep learning explainability outputs

I interpreted the SME's feedback in consultation with the thesis advisor, and my interpretation is outlined in the discussion chapter (Chapter 5). To ensure transparency and trustworthiness, the results chapter (chapter 4) includes the complete, unedited SME responses, with only minor redactions to comply with company privacy policies and protect proprietary information.

4 Results

4.1 Case Study Description

The dataset originates from a real-world, large-scale industrial process. To respect the privacy of the company, no identifying information or specific process details are disclosed.

However, to interpret the results presented in this thesis, it is recommended to have a basic understanding of the process structure and the sensor layout as shown in Figure 4-1:

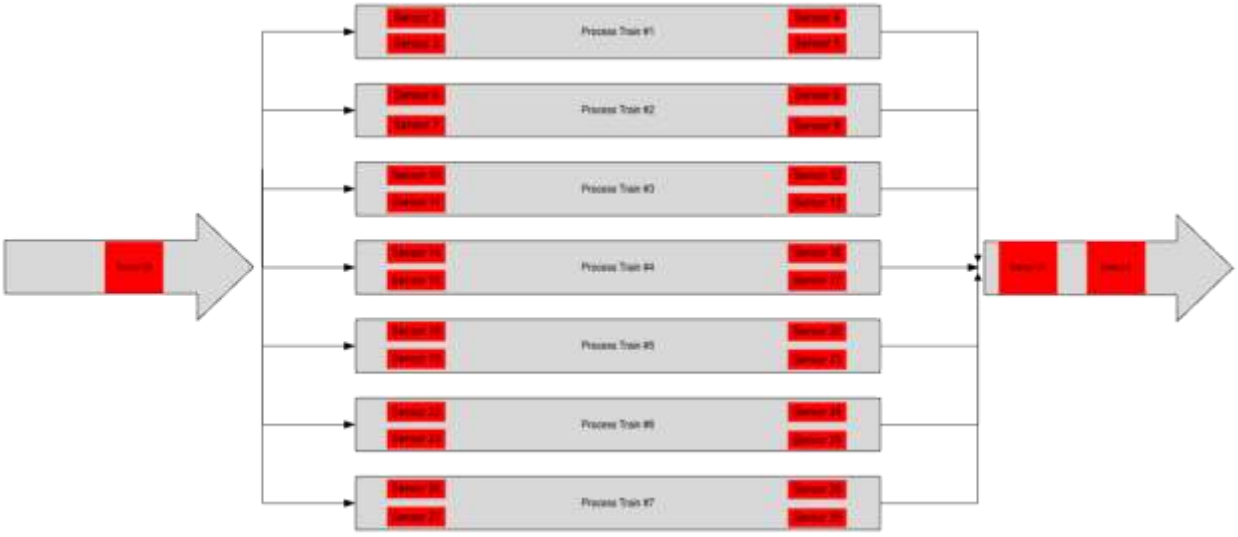


Figure 4-1- Process Flow and Sensors

The process begins at Sensor 30, where incoming material is measured before distribution into seven parallel identical process trains. Each train receives the material, processes it internally, and outputs it to a central collection point. Sensors 31, located downstream, measure the same parameters as sensor 30 for system-level evaluations. Within each train, Sensors 2, 6, 10, 14, 18, 22, and 26 monitor the same parameter; Sensors 3, 7, 11, 15, 19, 23, and 27 a second; Sensors 4, 8, 12, 16, 20, 24, and 28 a third; and Sensors 5, 9, 13, 17, 21, 25, and 29 a fourth. Sensor 2 is the unique forecasting target. This setup enables consistent monitoring of material as it enters, undergoes processing, and exits each train.

4

4.2 Results and Analysis

4.2.1 Deep Learning Model Results

4.2.1.1 Training Session

Model Train Time on train + validation set lasted 16 hours on MacBook Pro M3 16GB (CPU). Many try-and-error models were created during the modeling phase in different environments. At least 10 were trained on a private computer via the computer CPU, and eight were trained on a cloud GPU via Google's Colabratory (also known as Colab) environment. Surprisingly, it was found that the training environment can affect the model prediction performance; a comparison is shown later in this chapter. The only difference is that one was trained on a **GPU** (NVIDIA A100, 48 GB RAM via Colab). In contrast, the other was trained on a personal computer **CPU** (MacBook Pro M3, 16 GB RAM), using the same hyperparameters and datasets.

4.2.1.2 Data Statistics in the Training, Validation, and Test Set

The training was done on the train_raw.csv set and the val_raw.csv set (for the actual training, they were combined into a one long set). The combined set comprises 119,232-time stamps – from June 1, 2023, 00:00 until August 22, 2023, at 19:11 and 31 variables.

```
--- Statistics for Combined Train + Validation
Set ---
Number of timestamps: 119232
Number of variables: 31
Number of measurements: 3696192
Number of time steps 'Sensor 1' > 5.40: 404
Time in minutes 'Sensor 1' > 5.40: 404
Percent of time 'Sensor 1' > 5.40: 0.34%

--- Statistics for Test Set ---
Number of timestamps: 13248
Number of variables: 31
Number of measurements: 410688
Number of time steps 'Sensor 1' > 5.40: 90
Time in minutes 'Sensor 1' > 5.40: 90

Percent of time 'Sensor 1' > 5.40: 0.68%
```

Figure 4-2 - Statistics for combined train + validation and test set

The table below compares this study's training database statistics with the original TFT paper training set [39]:

Table 4-1 - TFT paper training set comparison with this study

Dataset Details	Electricity	Traffic	Retail	Volume	This Paper
Target Type	$\mathbb{R}$	[0,1]	$\mathbb{R}$	$\mathbb{R}$	$\mathbb{R}$
Number of Entities	370	440	130k	41	31
Number of Samples	500k	500k	500k	~100k	3.6M

Where:

- $\mathbb{R}$: All real numbers ($-\infty$ to ∞).
- [0,1]: Real numbers between 0 and 1, inclusive (often probabilities or normalized values).

From the data set perspective, the amount of data that was used was 8 times larger than the most extensive data set in [39].

As stated in the methodology chapter, hyperparameters were chosen using Optuna [46], which lets you test many different hyperparameter configurations and combines the best performance once. This study takes a similar amount of time to training (about 13 hours), but it reduces time by preventing the trial-and-error sessions you do in training.

Let's compare our study hyperparameters with the [39] hyperparameters:

Table 4-2 - [39] hyperparameters comparison with this study

	Electricity	Traffic	Retail	Vol	This Paper
Network Parameters					
K (input_chunk)	168	168	90	252	96
Tmax (output_chunk)	24	24	30	5	12
Dropout rate	0.1	0.3	0.1	0.3	0.7
State Size	160	320	240	160	40
Number of Heads	4	4	4	1	4
Training parameters					
Minibatch Size	64	128	128	64	256
Learning Rate	0.001	0.001	0.001	0.01	0.0001
Max Gradient Norm	0.01	100	100	0.01	N/A

Network Parameters

- k (Input Length = 96): Shorter than Electricity, Traffic, and Volatility datasets.
- Tmax (Forecast Horizon = 12): shorter than most benchmarks.
- Dropout Rate = 0.7: higher than others
- State Size = 40: smaller than all benchmarks
- Number of Heads = 4: Matches most benchmarks.

Training Parameters

- Minibatch Size = 256: Largest among all
- Learning Rate = 0.0001: Lowest value.

  Max Gradient Norm (gradient clipping) = N/A (see methodology chapter)

4.2.1.3 Prediction Results on the Test Set – GPU Vs CPU

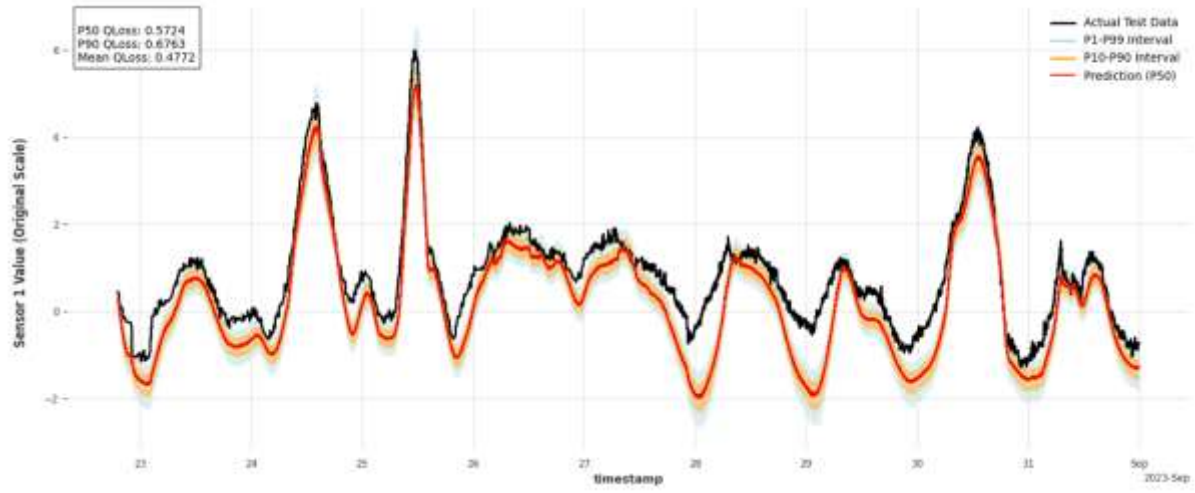


Figure 4-3 - Prediction VS Test Data (Trained on CPU)

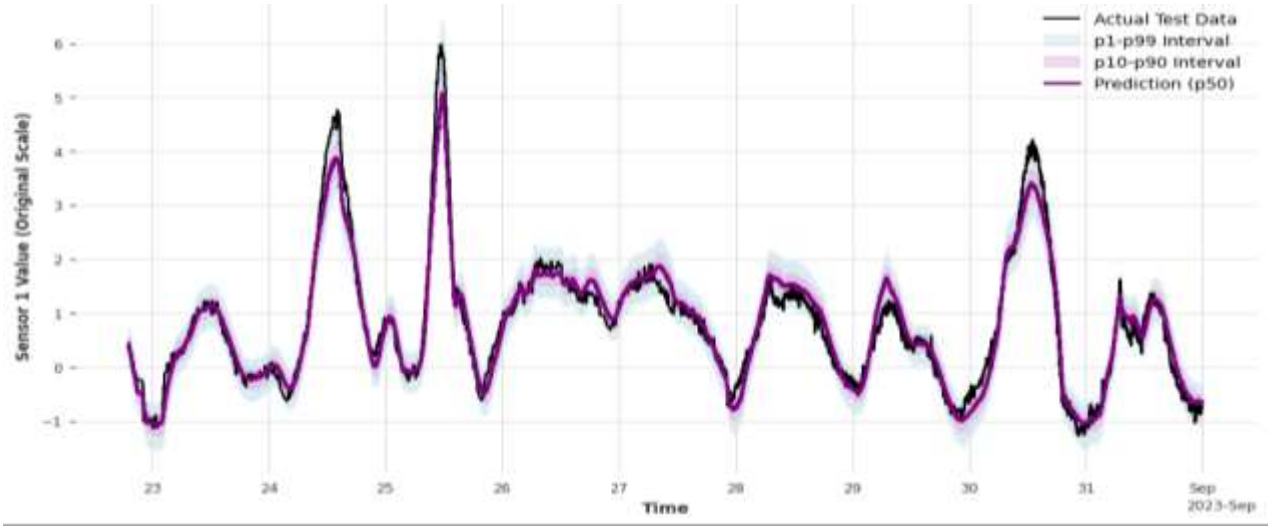


Figure 4-4 - Prediction VS Test Data (Trained on GPU)

Despite using the same model architecture, Figures 4-3 and 4-4 show differences in prediction between a model that is trained on a GPU and a model that is trained on a CPU:

Table 4-3 - prediction performance comparison CPU vs GPU

Metric	CPU	GPU
Mean Quantile Loss (MQL)	0.4771	0.1387
MQL P10	0.1554	0.0875
MQL P50	0.5724	0.1911
MQL P90	0.6762	0.1229
MQL P99	0.4667	0.0235

Table 4-3 compares how well the model predicted different quantiles when trained on a CPU versus a GPU. The GPU-trained model performed better overall, with a lower average quantile loss (0.1387 compared to 0.4771). This means the GPU model made fewer and minor errors across all percentiles.

The CPU model had a higher loss at P10, which represents the lower edge of the prediction range (0.1554 vs. 0.0875). This doesn't mean the CPU consistently predicted too high. Instead, it shows that its predictions around the lower bound were less accurate and more spread out. The GPU model gave tighter, more reliable lower-bound predictions, especially when detecting low values related to faults, failures, or safety limits.

At P90, where errors are more heavily penalized if the model underestimates, the CPU model had a much higher loss (0.6762 vs. 0.1229). This shows that the CPU model often failed to capture the peaks correctly, a pattern also visible in the graph, where actual values exceed the predicted upper range.

Overall, the GPU-trained model provided more accurate predictions across all the quantiles. The complete list of the mean quantile results is displayed in Table 4-5. The table shows "pinball loss" –a different name for Mean Quantile Loss.

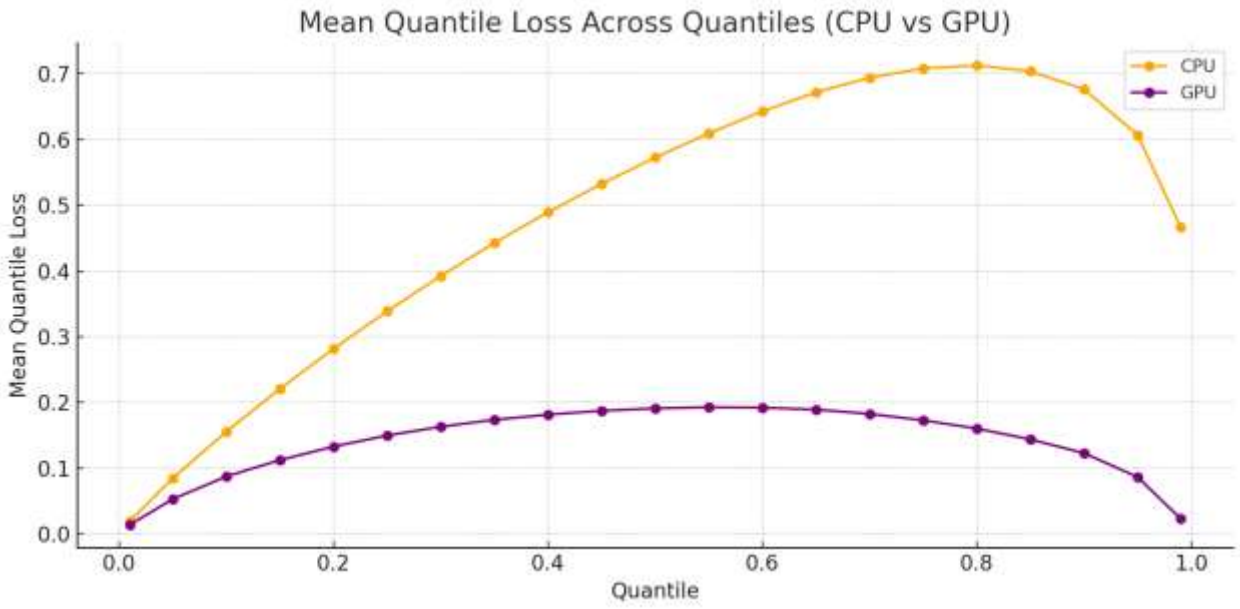


Figure 4-5 - Mean Quantile Loss Across Quantiles (CPU vs GPU)

This graph shows the Mean Quantile Loss (pinball loss) for each quantile (from 0.01 to 0.99), based on the values reported in Table 4-5. For each quantile, the value represents the penalized average error between the predicted and actual values across the test set; smaller values mean better performance.

Two lines are shown:

- Orange line = CPU-trained model
- Purple line = GPU-trained model

The CPU model's curve is more curved, especially between quantiles 0.5 and 0.95, indicating larger average errors. This means the CPU model struggled particularly in predicting upper quantiles like P90 and P99, often underestimating peaks, a pattern we saw in the previous time-series plots where the actual values exceeded the upper prediction bands.

At the lower end (P10 and below), the CPU model also shows noticeably higher loss compared to the GPU model. This doesn't mean it always overpredicted, but rather that its predictions were less accurate and more scattered around the lower bounds, a sign of unreliable estimates in the

tails. In contrast, the GPU model showed lower and more stable loss at P10, producing tighter and more trustworthy predictions in that range.

The GPU model's curve is consistently lower and flatter, meaning it predicted the central trend better and maintained strong calibration at the extremes (P10, P90, P99), all aligned with the tighter, more centered forecast bands in the earlier plots.

This graph confirms that GPU training results in better-calibrated and more accurate probabilistic forecasts across the entire range of uncertainty. It also shows that the GPU model handles both central trends (P50) and rare edge cases (P10, P90, P99) more reliably, essential for risk-aware forecasting in industrial settings.

Table 4-4 - Quantile loss (pinball loss) in our study – test set

Trained on CPU	Trained on GPU
Pinball Loss q=0.01,0.0192734120974321	Pinball Loss for Quantile 0.01: 0.0140
Pinball Loss q=0.05,0.0845217192159652	Pinball Loss for Quantile 0.05: 0.0528
Pinball Loss q=0.10,0.15540903155700184	Pinball Loss for Quantile 0.1: 0.0875
Pinball Loss q=0.15,0.2206029537659764	Pinball Loss for Quantile 0.15: 0.1124
Pinball Loss q=0.20,0.28154380435512427	Pinball Loss for Quantile 0.2: 0.1329
Pinball Loss q=0.25,0.3387624806510334	Pinball Loss for Quantile 0.25: 0.1497
Pinball Loss q=0.30,0.39230209532911925	Pinball Loss for Quantile 0.3: 0.1631
Pinball Loss q=0.35,0.4424862920659531	Pinball Loss for Quantile 0.35: 0.1736
Pinball Loss q=0.40,0.4891740834288782	Pinball Loss for Quantile 0.4: 0.1815
Pinball Loss q=0.45,0.5324556523530697	Pinball Loss for Quantile 0.45: 0.1873
Pinball Loss q=0.50,0.5724230738050814	Pinball Loss for Quantile 0.5: 0.1911
Pinball Loss q=0.55,0.6091960180859984	Pinball Loss for Quantile 0.55: 0.1927
Pinball Loss q=0.60,0.6427937120972179	Pinball Loss for Quantile 0.6: 0.1922
Pinball Loss q=0.65,0.6716030527230338	Pinball Loss for Quantile 0.65: 0.1892
Pinball Loss q=0.70,0.6939001671198257	Pinball Loss for Quantile 0.7: 0.1826
Pinball Loss q=0.75,0.7081280376542255	Pinball Loss for Quantile 0.75: 0.1729
Pinball Loss q=0.80,0.712494779817627	Pinball Loss for Quantile 0.8: 0.1603
Pinball Loss q=0.85,0.7037708317910174	Pinball Loss for Quantile 0.85: 0.1439
Pinball Loss q=0.90,0.676286753156839	Pinball Loss for Quantile 0.9: 0.1229
Pinball Loss q=0.95,0.606698237213269	Pinball Loss for Quantile 0.95: 0.0865
Pinball Loss q=0.99,0.466736697264262	Pinball Loss for Quantile 0.99: 0.0235
Mean Quantile Loss,0.4771696612165691	Mean Quantile Loss: 0.1387

Table 4-5 presents the quantile loss (pinball loss) across a range of quantiles (from 0.01 to 0.99) for models trained on CPU and GPU. The final row reports the Mean Quantile Loss (MQL), the average pinball loss across all quantiles.

From this table, several key insights emerge:

- The model trained on a GPU outperforms the CPU-trained version across all quantiles. The MQL drops from 0.4772 (CPU) to 0.1387 (GPU), dramatically improving overall forecasting performance.
- The GPU model achieves low error on extreme quantiles, such as P99 (0.0235 vs. 0.4667) and P90 (0.1229 vs. 0.6762). This indicates better handling of rare events.
- In contrast, the CPU model shows steadily increasing loss from lower to upper quantiles, with particularly high losses at the tails (e.g., P75–P95), suggesting underestimating peaks and poor uncertainty modeling.

This table quantitatively supports the earlier visual insights from the graphs: GPU training improves central forecasts (P50) and leads to tighter, more accurate prediction intervals across the full distribution.

4.2.1.4 Test Data Insights

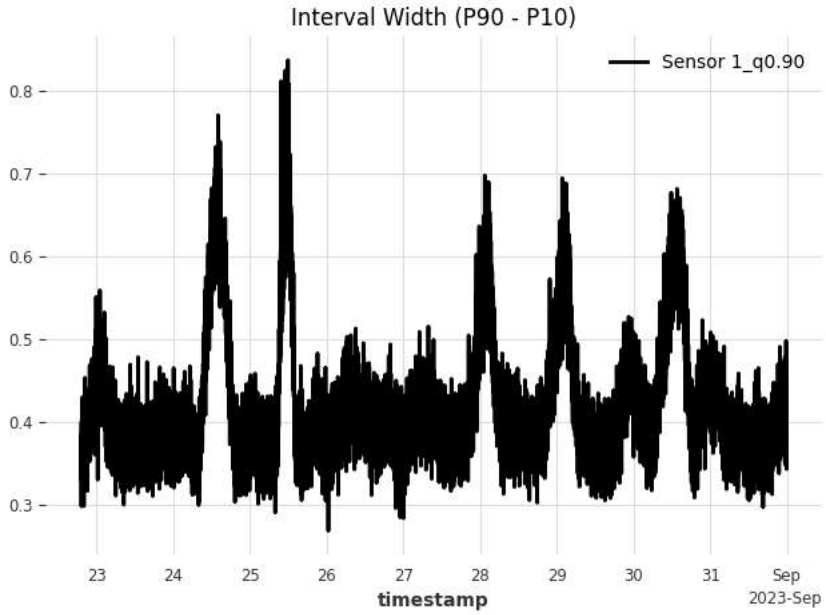


Figure 4-6 - Interval Width (P90-P10) - Test set (CPU)

Figure 4-7 presents the prediction interval width between the 90th and 10th quantiles (P90–P10) over time for Sensor 1. This interval represents the model's estimated uncertainty range. The width fluctuates notably across the forecast horizon, with peaks aligning to periods of higher variability or signal spikes, particularly around the 24th, 26th, 28th, and 30th of September. In contrast, lower and more stable intervals occur during quieter signal phases.

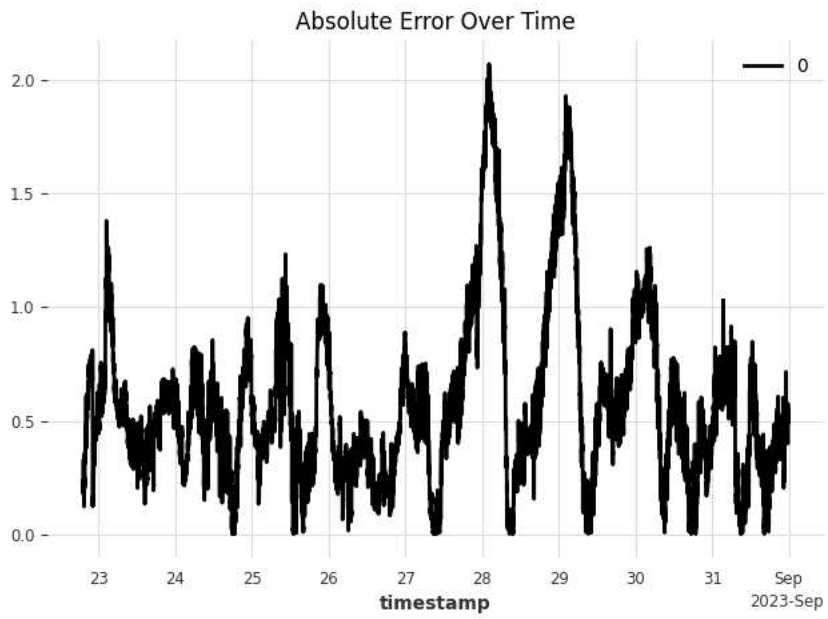


Figure 4-7 - Absolute Error Over Time - Test set (CPU)

Figure 4-8 illustrates the absolute error over time between the model's predicted median (P50) and the actual observed values. The error fluctuates dynamically, with noticeable spikes on September 24th, 28th, and 30th, corresponding to periods of rapid signal change or higher volatility. There were sudden drops on 28 and 29, suggesting that the model predicts the peaks better than the lows. The results in this graph are calculated with the actual quantile result, not the quantile loss. This gives us a more intuitive understanding of the model performance – it is what we care about – how far our prediction is from the actual value. We can see three noticeable spikes (~1.5, 2, ~2) on September 23, 28, and 29. Those correspond to deep valleys illustrated in Figure 4-3 on those dates, which shows that the model struggles to be accurate in the lowest points of the actual results. In our specific case, we care more about the peaks than the valleys. Figure 4-3 shows us two prominent peaks: between the 24^{th}-25^{th} and 25^{th}-26^{th}. The absolute error in those times looks inverse, indicating better performance for peak prediction than valleys. Even though the prediction at the peaks is better overall, it is still underestimating at the highest values. Those peaks are critical because we are trying to predict the exceedances, specifically those above 5.4. This specific one

point is that the actual value hits 6, the model underpredicts the result by 1 point. This suggests that a more robust training may be needed for more accurate peak predictions.

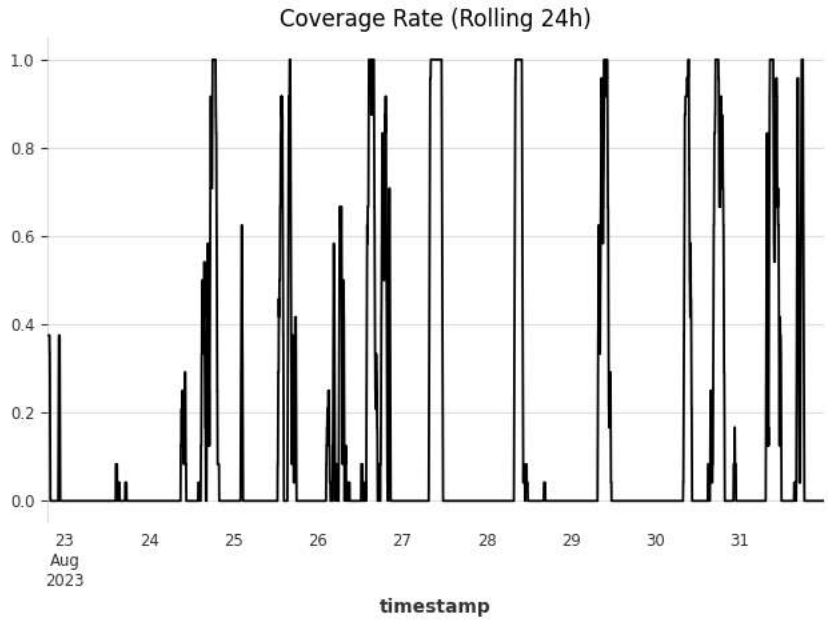


Figure 4-8 - Coverage Rate (Rolling 24h) – Test set (CPU)

Figure 4-9 presents the Coverage Rate (Rolling 24h), which tracks the proportion of actual test values that fall within the model's predicted confidence interval (e.g., P10–P90) over the past 24 hours. This metric serves as an indicator of how well the model's uncertainty estimates align with tangible outcomes. A value of 1.0 indicates perfect coverage, i.e., all actual values in the past 24 hours were captured within the prediction interval, while a value of 0.0 reflects a complete miss. In an ideal scenario, the coverage rate would remain close to the nominal level of the prediction interval (e.g., 0.80 for a P10–P90 interval), indicating that the model is neither under- nor over-confident. In this graph, sharp spikes and dips suggest instability in the model's uncertainty calibration, sometimes predicting too conservatively (overly broad intervals), and other times too narrowly (missing absolute values). This inconsistency may point to no stationarities in the data or moments of regime shifts that the model did not capture well.

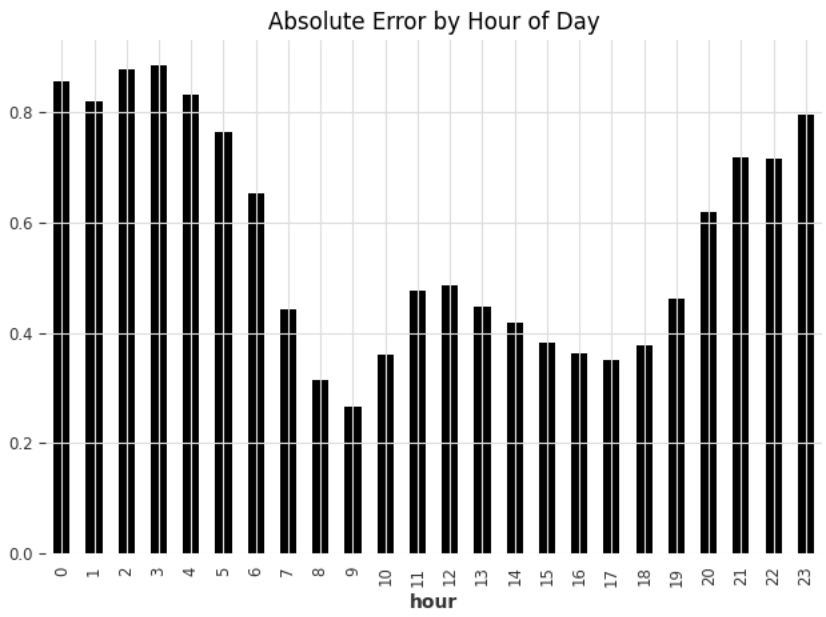


Figure 4-9 - Absolute Error by Hour of Day – Test set (CPU)

Figure 4-10 presents the absolute forecasting error grouped by hour of the day, revealing a clear diurnal pattern in model performance. The highest errors occur during late-night and early-morning hours (00:00–06:00), with values exceeding 0.8, suggesting reduced model accuracy during low-activity periods. In contrast, the error decreases significantly during midday (08:00–18:00), reaching a minimum around 09:00–17:00, indicating improved performance when signal behavior is more regular and structured. Error increases again in the evening hours.

The following figures (4-9, 4-10, 4-11, 4-12) present internal outputs generated directly from the Temporal Fusion Transformer (TFT) model architecture. The model's built-in interpretability mechanisms make these insights – variable importance attention scores-possible. Unlike many deep learning models that function as “black boxes,” TFT provides transparent, structured interpretability through dedicated components that explain its decision-making process. This unique feature enables a deeper understanding of model behavior and the ability to validate its alignment with domain knowledge. All interpretability plots in this chapter are native outputs from the TFT implementation and cannot be reproduced without it.

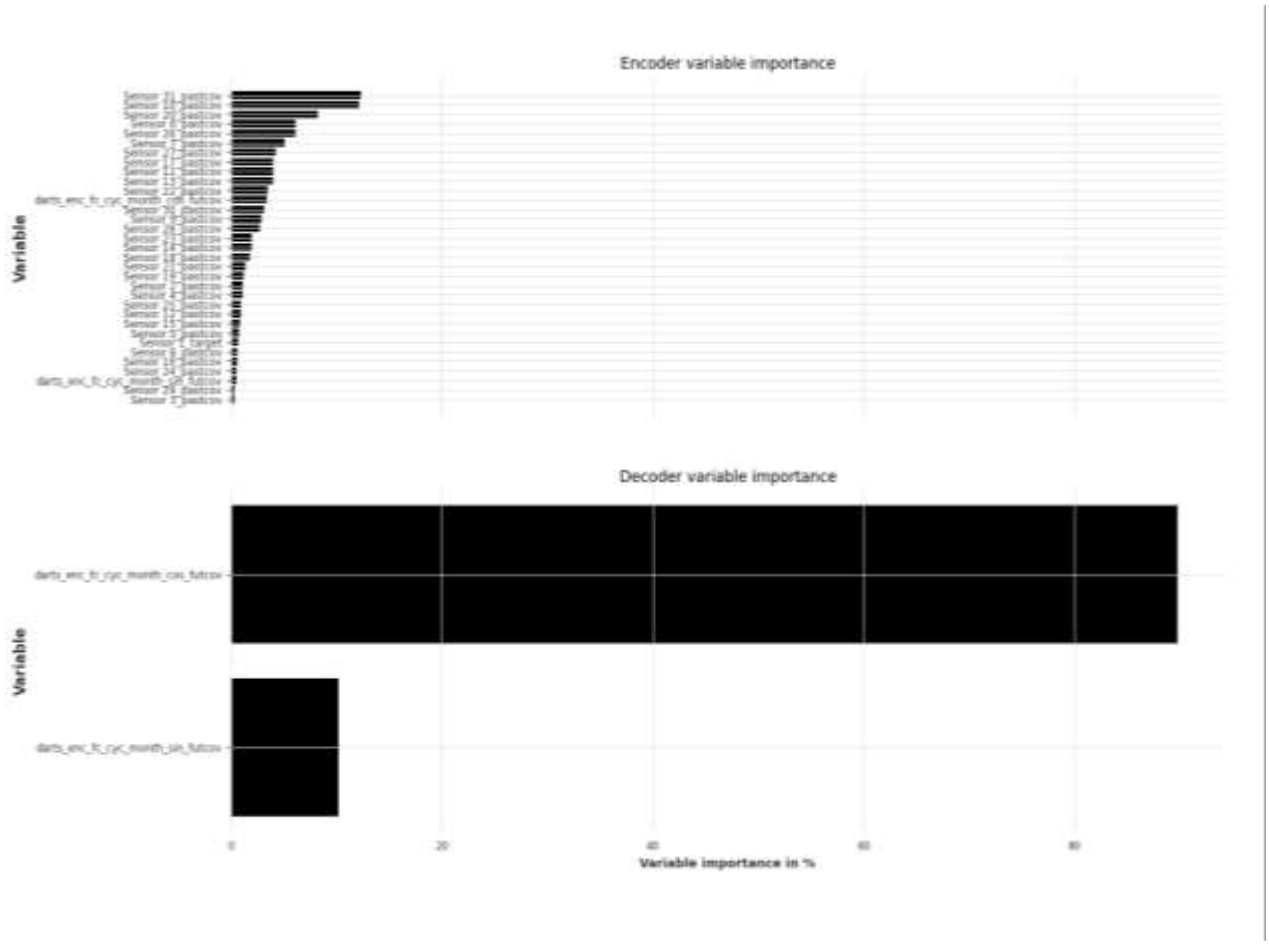


Figure 4-10 - Encoder and Decoder Variable Importance – Test set (CPU)

Figure 4-9 presents the variable importance for encoder and decoder inputs in the Temporal Fusion Transformer model. The encoder variable importance plot shows that Sensor 31, 10 (Train 3), and 20 (Train 5) contribute most significantly to the model's forecasts.

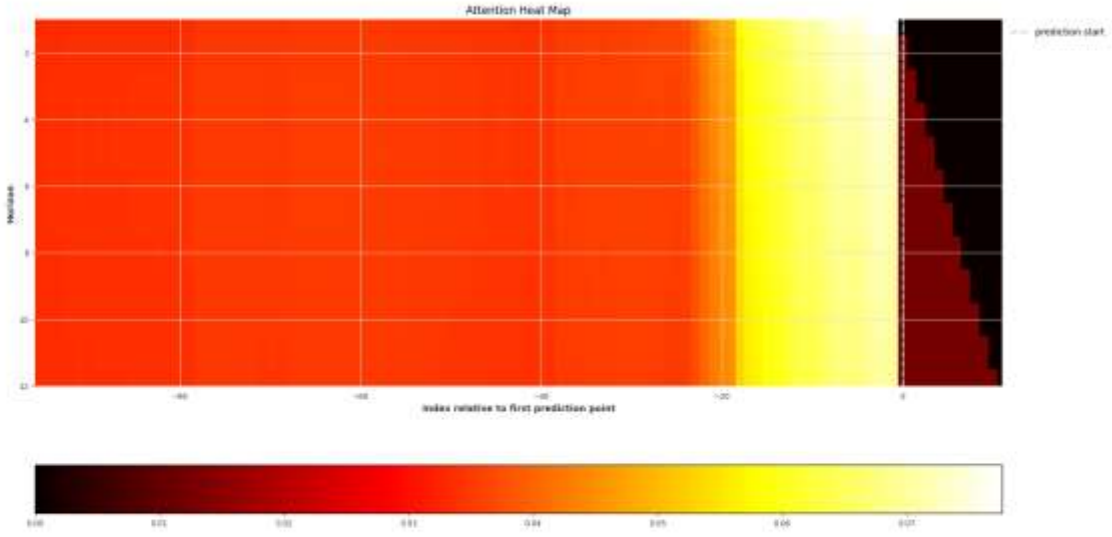


Figure 4-11 - Attention Heat Map - Test set (CPU)

Figure 4-10 (Heat Map): This shows attention weights by time index (x-axis) (96 minutes) and forecast horizon (y-axis) (12 minutes). The model strongly focuses on time steps just before the prediction starts (near index -20 to -), indicated by the bright yellow band. Attention decreases steadily as input points move further into the past.

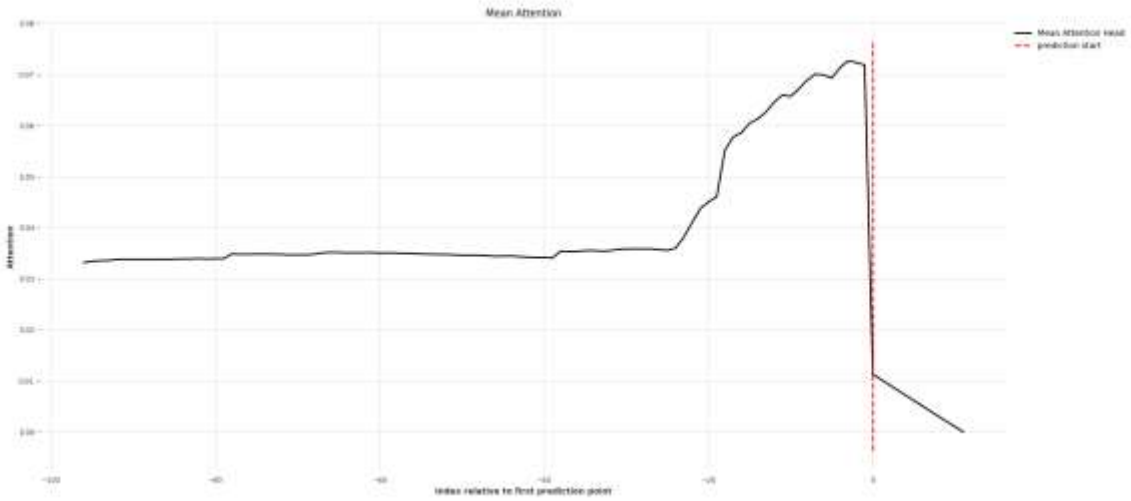


Figure 4-12 - Model Mean Temporal Attention - Test set (CPU)

Figure 4-11 (Mean Attention): This plot aggregates attention across all horizons. It confirms that the model increasingly focuses on recent history, peaking just before the forecast starts, then sharply dropping to near-zero for decoder steps (post-prediction inputs).

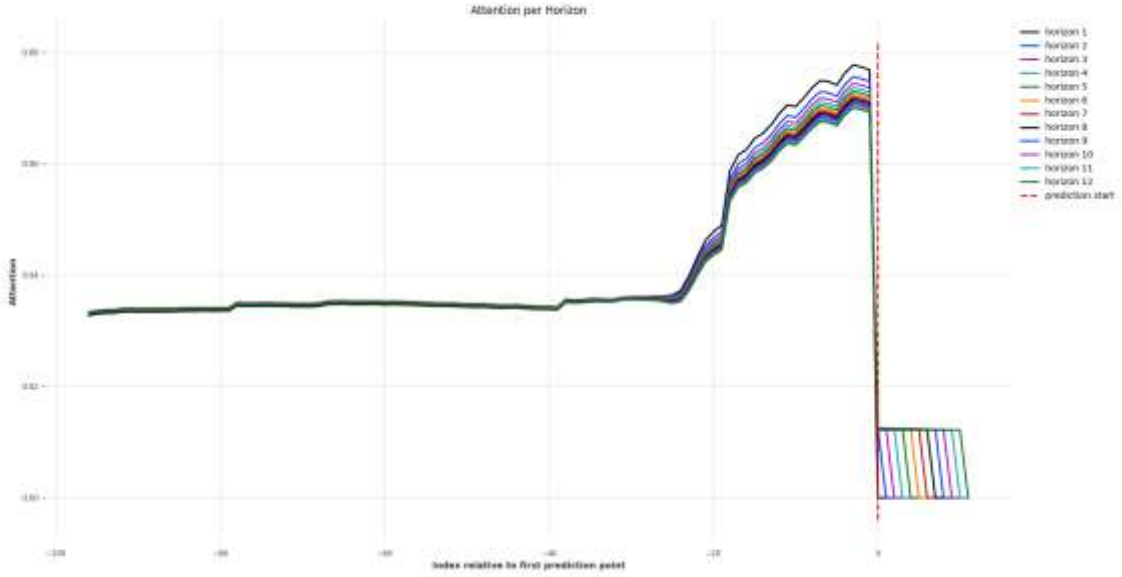


Figure 4-13 - Temporal Attention per Horizon

Figure 4-12 (Attention per Horizon): Each line represents attention weights for a specific forecast step (horizon 1 to 12). All horizons assign the highest attention to the immediate past, with consistent patterns across forecast steps.

4.2.2 Domain-Specific GPT Combined with A Deep Learning Forecaster (Case Study)

The data set used for the hybrid approach is based on dates different from the test data. The original model was trained and tested on data from June 1st to August 31, 2023. A fresh new data set was chosen for the hybrid approach to test the model's generalization. The data of the data set is June 19th, 2025. And it's a window of 96 minutes, from 11:27 am until 1:02 pm. The forecast of 12 minutes is from 1:03 pm until 1:14 pm. The data set had an exceedance in the actual results at those 12 minutes, and this is the reason it was chosen. Those 12 minutes are hidden from the model (deleted), and it has never been trained on this date.

Below are the explainability outputs of this data set from the TFT prediction:

4.2.3 Case-Study

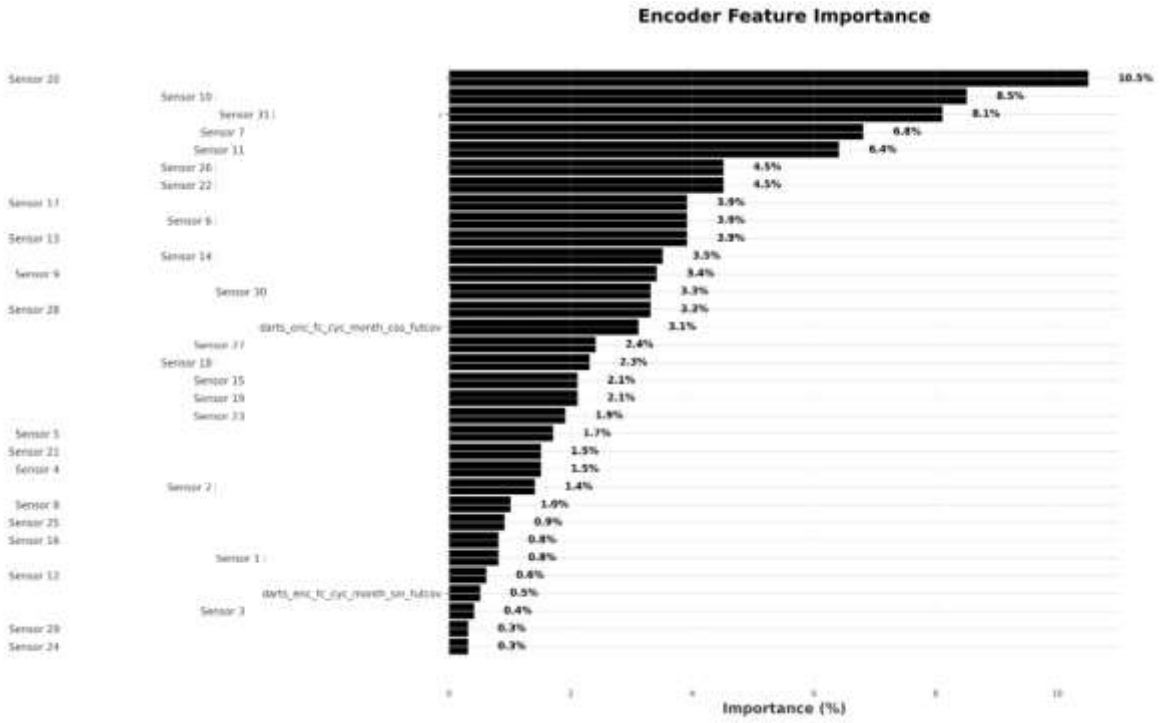


Figure 4-14 - Encoder Feature Importance (Case Study) (CPU)

The blank spaces between the graph bar and the Sensor tag are the result of blanking the full name of the sensor.

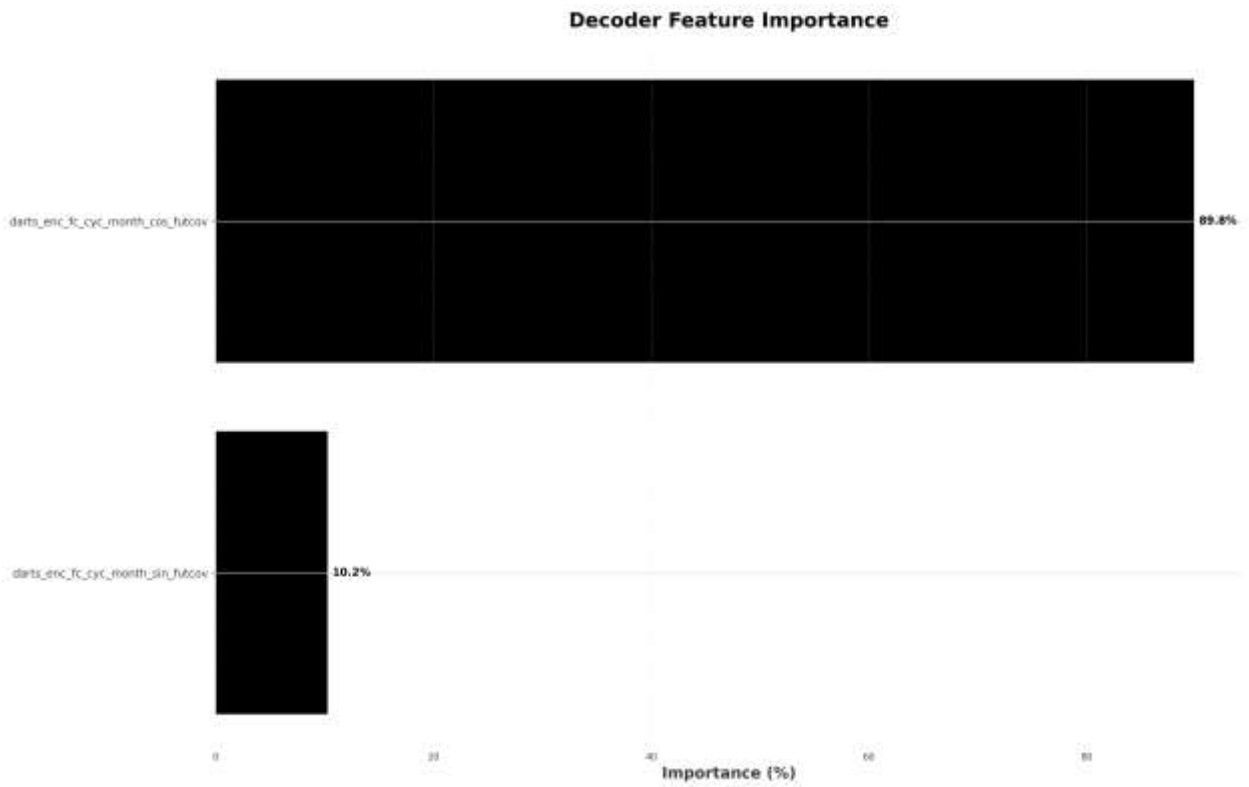


Figure 4-15 - Encoder Feature Importance (Case Study) (CPU)

Figures 4-13 4 and 14 show the encoder and decoder feature importances, respectively, offering insights into which variables the Temporal Fusion Transformer model relies on most for forecasting. The graph shows that sensors 20, 10, and 31 are the most impactful (like the test data, but in a different order).

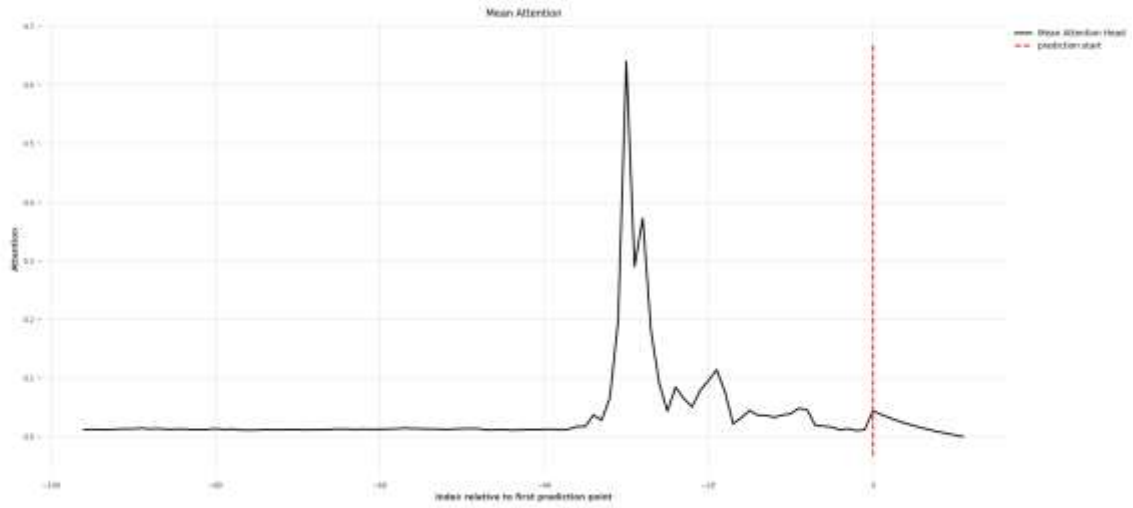


Figure 4-16 - Mean Attention (Case Study) (CPU)

Figure 4-15 (Mean Attention): This graph shows the average attention across all decoder steps. The attention sharply peaks around 30 steps before the prediction point, indicating that this segment of historical data is most influential for the model's forecasts. Attention gradually decays as we move farther from the peak, with minimal weight assigned to inputs far in the past or right before the prediction starts.

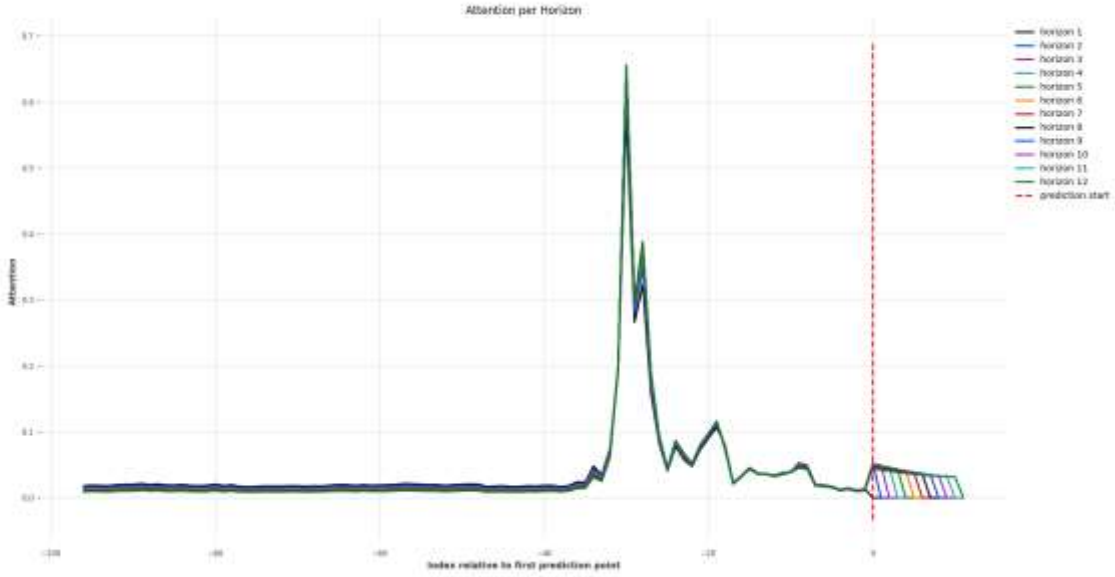


Figure 4-17 - Attention per Horizon (Case Study) (CPU)

Figure 4-16 (Attention per Horizon): Each colored line corresponds to a specific forecast horizon. All horizons consistently focus on the same high-attention window -roughly 20–30 steps before prediction - reinforcing that this mid-range historical segment carries the most predictive value.

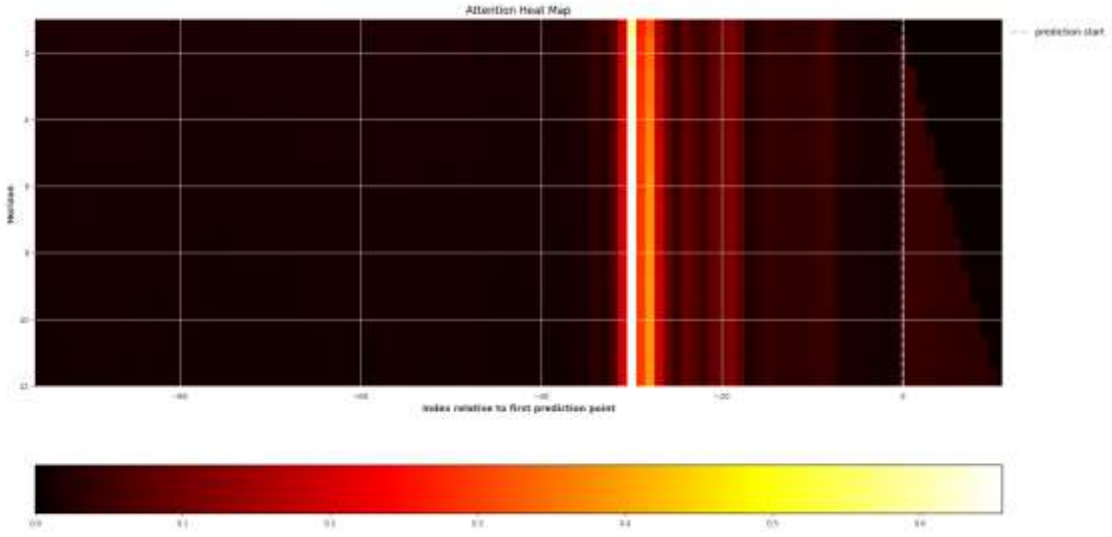


Figure 4-18 - Attention Heat Map

Figure 4-17 (Attention Heat Map): The heat map visually emphasizes the attention peak near index -30, where the intensity shifts from dark red to bright yellow/white. This transition signifies maximum attention concentration and aligns with Figures 4-15 and 4-16 patterns. Beyond this window, attention is diffuse and relatively uniform, confirming the model's focus on a narrow temporal region before forecasting begins.

Overall, we can summarize the TFT's model performance as follows:

- We saw performance differences between the GPU model and the CPU model, when the GPU model performs better at all quantiles.
- Coding environment differences prevented me from using the GPU model. Eventually only the CPU model was used, as the full pipeline code (forecaster and LLM) was written on my private computer that is running on a CPU. Dart's documentation [47] is stating that it is possible to train on a GPU and load on CPU, but in reality the problem was found to

be persistence and was not solved in the needed time, so a decision was have t be made and the CPU model was chosen. The following is an explanation of the possible causes that created the compatibility issue: The GPU model was trained in a Google's Colab environment. Google Colab is cloud-based environment for writing and running Python code. Colab let you run your code on a Google-hosted virtual machine equipped with an NVIDIA GPU and CUDA drivers + libraries preinstalled. This setup enables deep learning code in Colab to offload tensor operations to the GPU, and to reduce the training time. When trying to load the GPU trained model on my local machine (MacBook Pro M3) the model refused to load or execute predictions. The inability to load the GPU-trained Darts model on a CPU system was most likely due to differences in the underlying software environment. Models trained on GPU often depend on specific versions of libraries (such as PyTorch and CUDA) that are not directly compatible with CPU-only setups. If the versions of these libraries differ between the training environment (Google Colab) and the deployment environment (Cursor), incompatibility errors can occur during model loading. Another possible reason is that the model checkpoint may contain GPU-specific tensor allocations, which require explicit conversion to CPU format before loading. Without this conversion, the CPU environment cannot interpret the saved model state correctly. The CPU model used in this thesis is the CPU model, and the following bullets refer to the CPU model.

- The model under-predicts valleys (~ by 2 points) and peaks (~1 point).
- Even with the underpredictions, the mean quantile loss for the P50 is 0.572, meaning, on average, at the 0.5 quantile, the predicted results are about 0.5 points away from the actual results, suggesting that on stable durations, the prediction is more accurate.
- In our case, we care more about peaks, specifically results above 5.4. Under-prediction means that we may miss actual exceedances. This suggests a different training approach is suitable for better accuracy at those peaks, training on a data set with more frequent peaks

so that the model can learn more patterns in those peaks. Also, using a GPU-trained model should give overall better accuracy.

To summarize, the accuracy of the CPU model is generally sufficient for this thesis. Still, to use it in production, the SME should evaluate it to determine if this accuracy is adequate or if more training is needed.

4.2.3.1 GPT-Generated Reports

The LLM that was called to generate insights was gpt-4.1-2025-04-14 [48]. The communication with the model was done via OpenAI's API Application Programming Interface (API) with a dedicated API key, using a Python environment.

Total USD spent for communication with the model, including the engineering phase of the model, was $12.20 with 6,764,676 Total tokens used, and 251 calls.

4.2.3.1.1 The User Message

Throughout all the versions, the LLM was given a CSV file, with 12 minutes of forecasted data and the 96 minutes past covariates (the 30 sensors) that were used to predict the 12 minutes. Via the user message, the LLM was instructed to review the CSV file (attached as a file to the message) to generate an output that would include the following 4 items:

- Identification of the violation timestamp in the data set.
- To find the minimum reduction needed to reduce the sensor 1 measurement below the threshold, and to specify the start and the end time of the reduction window.
- To investigate the root cause of the violation with the use of the 5whys technique [49]
- To suggest corrective actions based on the root cause that was found.

The role that was provided to the LLM is "Senior Process Engineer."

A deep model is forecasting a measurement violation (in Sensor 1) within the next 12 minutes.
The facility operator requests the senior process engineer (the LLM) to generate an engineering response with the above outlined outputs that will help to prevent this violation and mitigate future violations.
An example of a user message is outlined in the methodology chapter 3.5.2.

When the retrieval-augmented generation library was added to the LLM, the following dedicated section was added to the user message:

```
---

### Reference Knowledge

- Professional knowledge to support your analysis: {rag_context_text}
"""
```

Figure 4-19 - Injecting RAG context to the LLM via the user message

Six different user messages were created, 1 for each more knowledgeable LLM.

4.2.3.1.2 The System Message (System Prompt)

The system messages are guidelines that the developer gives the LLM. This message guides the LLM to a desired behavior. Per OpenAI's chain-of-command [50], the system message overwrites the user message; it is higher in the chain of commands but under only to the Platform message, which are guidelines that OpenAI gives the LLM, generally to prevent any harm to other people, and they are out of reach of the developer or the user.
Five versions (and one more for the integration of the deep learning model) of system messages (messages sometimes are referred to also as "prompts") were created, each progressively better informed (more information about the specific process that created the violation) or knowledgeable (more engineering knowledge in the process domain) than its predecessor.

This iterative prompt design aimed to identify whether more information and knowledge would result in more valuable and accurate responses by the LLM.

The evaluation of the LLM response was qualitative and was done by an engineer who is a subject matter expert (SME) for this specific process that created the violation.

The SME was requested to answer the following questions, for each report version, relative to the previously generated report by the less knowledgeable LLM:

- Is it accurate?
- Is this useful information?
- What are the strengths?
- What are the limitations?

Below is a table that describes the changes in the system message in the first five versions:

Table 4-5 - Five System Prompts

Version	System prompt
NAÏVE	Basic system prompt (see below)
NAÏVE + Process Cheat Sheet	Added process description
NAÏVE + Process Cheat Sheet + 5whys	Added 5whys methodology from the literature, using RAG library.
NAÏVE + Process Cheat Sheet + 5whys + domain textbook	Supplemented the knowledge base with a foundational textbook addressing key principles in the target domain, using RAG library.
NAÏVE + Process Cheat Sheet + 5whys + domain textbook + IE	incorporated Instrument Engineers' Handbook, Volume 1: Process Measurement and Volume 3: Process Software and Digital Networks to

	expand coverage of instrumentation and control systems.

The basic system prompt was generated following the guidelines of the prompt engineering cookbook of GPT 4.1 [42]. The system prompt was written in Markdown [51] (markup language). A system prompt example can be found in the methodology chapter 3.1.5.1. Five versions of the system prompt were created (and one more for the deep learning model), one for each more knowledgeable LLM.

Resolving obstacles with prompt engineering:

Prompt engineering was a game of trial and error. Many versions were made until a final version was good enough. For example, 59 versions were created for the version "NAÏVE + Process Cheat Sheet + 5whys + domain textbook" until a final version was ready to be used. Prompt engineering is centered on helping the model detect the first exceedance and correctly aggregate flow rate values. Also, focus was given in the prompt engineering on adding references from the RAG library when using their knowledge. This was done to create a more trustworthy engineering report. Key strategies included clarifying the timestamp format, defining strict threshold rules, mapping column headers, isolating relevant sensors, and enforcing step-by-step extraction templates. Followed by self-validation prompts to ensure alignment with the CSV structure. Those were only instructions to verify his answers without giving the LLM any information.

4.2.3.1.3 SME Evaluation of the Engineering Reports

Below are the SME responses to each of the models. Some words (marked *) were omitted from the response to maintain compliance with information confidentiality.

4.2.3.1.3.1 NAÏVE

SME Response:

> ***"The report is accurate in identifying that the * generally decreases when the * decreases and that the correlation isn't linear. It is also correct in its analysis that other process conditions contribute to the increased *, and there is no prediction controlling or preventing exceedances. The reports suggest variations between * contributing to the changes, but generally, we don't see any significant differences in * composition between the * that would contribute to this.***
>
> ***The general analysis performed provides information already available, and I don't feel it adds to our understanding of the issues within the system. However, the specific instructions for * reduction could be useful in limiting the production reduction, since today we respond by reducing production in a very basic manner, with no variation between possibly different scenarios. If we can reduce the deferred production caused by being out of spec, it could be financially beneficial."***

4.2.3.1.3.2 NAÏVE + Process Cheat Sheet:

Adding a new chapter to the system message called "Facility Overview":

- Facility Overview:
    - 20 words describing the process in a very general and high-level way.
    - Design capacities and specifications.
    - Process Descriptions – describing the central units in the process (3 bullet points, about 10 words each)
    - Process Flow – direction of the flow between units (5 bullet points, about 10 words each)
    - Key Process relationships

Note: The design capacity and specification are not real, and their goal was to resemble only a real-world operation with real process constraints.

SME response:

> ***"For the second report (Naive + Process Cheat Sheet), it maintained a good logical conclusion process. The model was better and more specific in identifying systems that can relate to the performance and the rise in * compared to the first (naive) model.***
> ***The interpretation of possible * fouling didn't seem adequately supported, as it also mentions a change in * that is much more directly related to the outlet *.***
>
> ***While the report (and previous report) correctly mentions no prediction controlling or preventing exceedances, it only gives very general suggestions on how to improve and predict the exceedances, which do not necessarily add to our knowledge on the matter."***

4.2.3.1.3.3 NAÏVE + Process Cheat Sheet + 5whys:

Adding a Retrieval Augmented Generation library:

- The following structure was used to generate the RAG:
    - LangChainRAGLibrary: Core RAG component that handles individual PDF documents. It uses PDFPlumberLoader to extract text, RecursiveCharacterTextSplitter for chunking (1000 chars with 200 overlap), OpenAIEmbeddings for vectorization, and FAISS for similarity search. Stores both vector index and raw chunks for dual access patterns.
    - The PDF file added to the RAG library is a description and guidelines on the five whys technique by Olivier Serra [52].
    - The actual injection of the rag knowledge into the LLM is done in the user message, as described in the user message chapter.

The following was added to the system prompt:

- Adding more instructions and details to the root cause analysis chapter, mainly:

- Instructing the LLM to do a 5whys analysis exactly as instructed in the 5whys literature in the RAG library.
- Referring to the specific place in the RAG library with the following structure in the prompt: [Relevant Reference Material: Five Whys Technique]

SME response:

> ***"For the Third report (Naive + Process Cheat Sheet + Five Whys)-***
> ***Again, it is consistent with the others in presenting a clear and logical process. The model is better at identifying plausible reasons than the second (Naive + Process Cheat Sheet) report, which is important since there are a number of variables leading to an exceedance, and it is helpful to have different investigation angles called out. Additionally, the report is also more detailed in the corrective action suggestions."***

4.2.3.1.3.4 NAÏVE + Process Cheat Sheet + 5whys + domain textbook:

A domain textbook was added, describing the domain's fundamental principles. To support the RAG library with multiple files, the following was added to the code:

- MultiLangChainRAGLibrary: Orchestrator that manages multiple RAG libraries simultaneously. It initializes separate LangChainRAGLibrary instances for each domain source (Five Whys, Fundamentals, Instrumentation Handbooks) and provides unified retrieval methods that aggregate results across all sources with source labeling. The architecture supports semantic similarity search and page-specific retrieval, with automatic index persistence and loading for efficiency.

The following was added to the system prompt:

- Role:
  - You have deep expertise in:
    - [Relevant Reference Material: Fundamentals] - available in the RAG library.
- Instructions:

- Add a dedicated References section at the end of your report. In this section, cite relevant excerpts from the file [Relevant Reference Material: Fundamentals] available in the RAG library.
- In the reference section, link each cited excerpt to the specific conclusion or decision it supports in your report to enhance engineering credibility. Do not include any reference if the material does not explicitly support your point.

- Output Format
  - 6. Reference

SME Response:

"Though I cannot easily judge the accuracy of the prediction, it is helpful that this report gave more explanation as to why the specific * reduction was selected. The recommended reduction itself is also reduced compared to previous reports, which, if it indeed achieves the * spec, is significant in terms of our goal of minimizing disturbances to production.

I think the model is improved in its focus on the process more than specific equipment fouling, but I would have found it useful to leave some information on equipment that could play a factor in the exceedance, even if it would not explain the specific event. The report gave similar recommendations for preventative actions to the previous report."

4.2.3.1.3.5 NAÏVE + Process Cheat Sheet + 5whys + domain textbook + IE

Added the following textbooks to the rag library:

- Instrument Engineers' Handbook, Volume 1: Process Measurement and Volume 3: Process Software and Digital Networks to expand coverage of instrumentation and control systems.

Added the following to the system prompt:

- Role:
  - You have deep expertise in:

        - [Relevant Reference Material: Instrument Engineers Handbook Vol 1 - Process Measurement] - available in the RAG library.
        - [Relevant Reference Material: Instrument Engineers Handbook Vol Three - Process Software and Digital Networks] - available in the RAG library.

- Instructions:
    - Add a dedicated References section at the end of your report. In this section, cite relevant excerpts from the files:
        - [Relevant Reference Material: Instrument Engineers Handbook Vol 1 - Process Measurement] - available in the RAG library.
        - [Relevant Reference Material: Instrument Engineers Handbook Vol Three - Process Software and Digital Networks] - available in the RAG library.

SME Response:

"As with the previous report, I appreciate the explanation it provides of the * reduction compared to earlier versions and that the numbers are once again reduced.
The model focused on process parameters while also paying attention to instrumentation, which I believe is a notable improvement. Instrumentation could indeed be an important factor that was not adequately addressed in earlier reports.
However, similar to the last report, I would have preferred it to include some information on equipment that could play a factor in the exceedance, even if it would not explain the specific event.
The report gave similar recommendations for preventative actions similar to those provided in the previous report."

4.2.3.1.3.6 Combined Deep Learning Model and Domain-specific GPT

I used a multimodal approach to connect the deep learning model to the domain-specific LLM. Leveraging the vision capabilities of GPT 4.1.

This creates a model for a decision pipeline where deep learning insights are translated into engineering actions through LLM interpretation.

I added the following outputs from the deep learning model to the user message. For clarity, both the domain-specific GPT and the deep learning model used the same data set (the CSV file injected into the LLM in the user message is generated by the deep learning model, from June 19, 2025). This enables us to develop a 6th version of the report:

Table 4-6 - Version six of the system prompt for adding deep learning insights

Version	System prompt
NAÏVE + Process Cheat Sheet + 5whys + domain textbook + IE + deep	Added variable importance, attention plot and flowrate forward simulator

The following approach was taken to integrate the models:

4.2.3.1.3.6.1 TFT Plots

Temporal fusion transformers have the ability for interpretability, a crucial feature that gives more trust to the model by humans. The model generated two main plots:

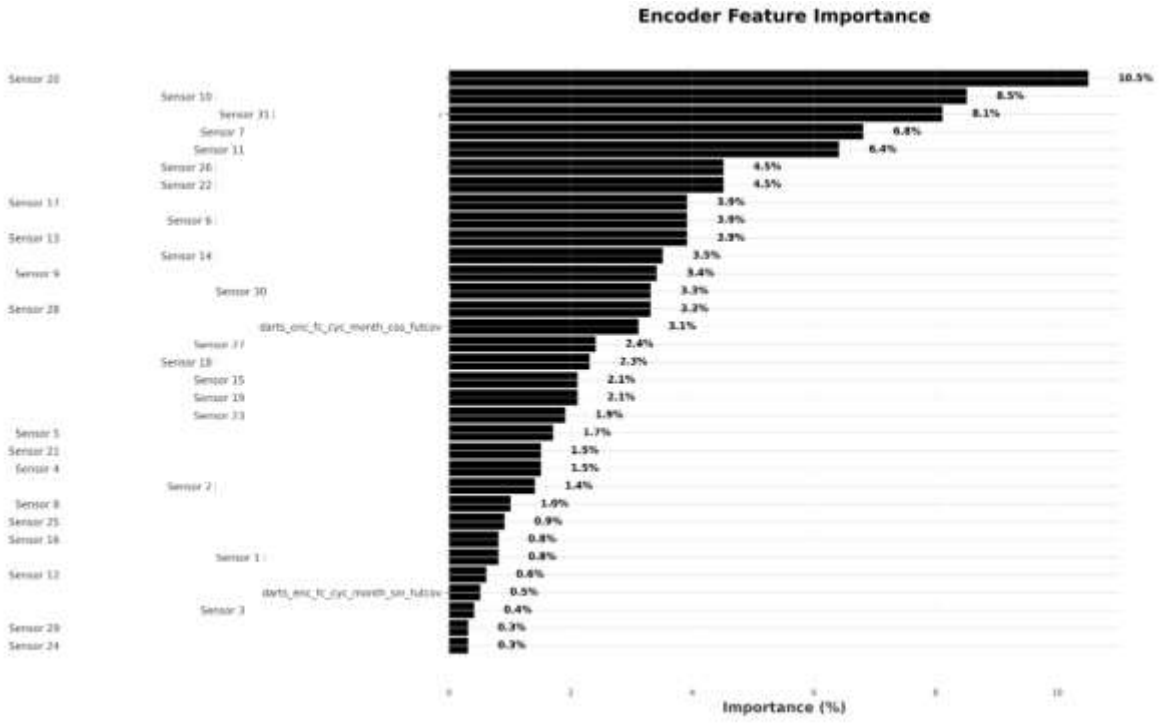


Figure 4-20 - Encoder Feature Importance (Case Study) (CPU)

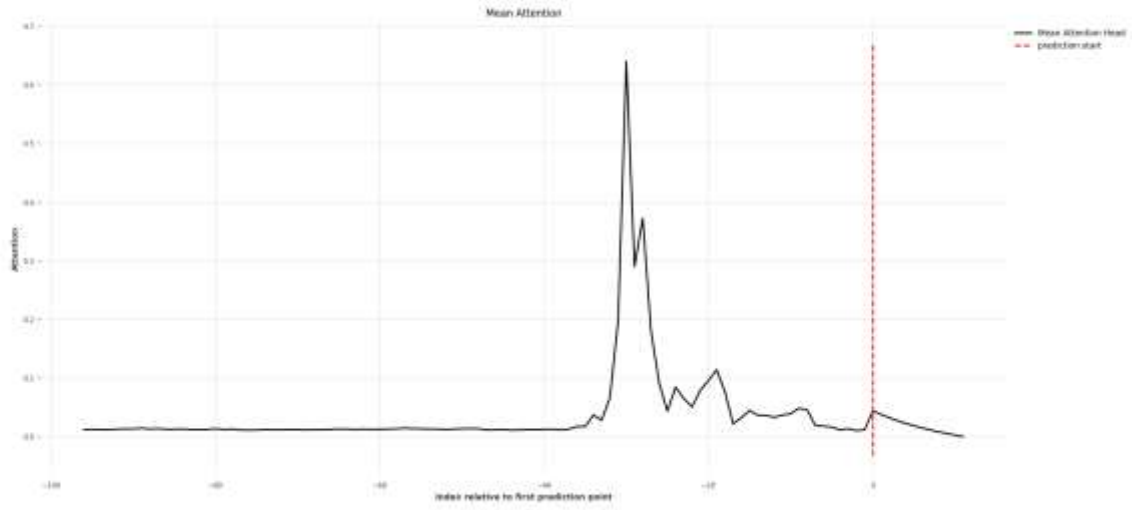


Figure 4-21 - Mean Attention (Case Study) (CPU)

To send the files to the LLM, it is necessary to upload them to a web URL and share the URL with the LLM. For this, I used Google Drive and copied the URL of the uploaded file.

Using the following code, I integrated the pictures into the user message:

```
initial_message_content = create_message_with_images(scenario_query_text, include_images)
```

This step combines the text-based scenario description with the visual graph elements from the TFT, creating a multimodal input that allows the LLM to process textual data and visual insights simultaneously.

```
response = openai_client.chat.completions.create(
    model="gpt-4.1-2025-04-14",
    messages=[
        {"role": "system", "content": system_prompt},
        {"role": "user", "content": initial_message_content}  # Contains images
    ],
    max_tokens=4000
)
```

The GPT-4 Vision model processes the combined text and image input, allowing it to analyze the visual patterns in the attention plots and feature importance charts while simultaneously processing the numerical data and contextual information.

4.2.3.1.3.6.2 Forward Simulator with the TFT prediction model

In addition to the TFT plots, I used the model to create a forward simulator:

The GPT-4.1 Vision model processes the combined text and image input, allowing it to analyze the visual patterns in the attention plots and feature importance charts while simultaneously interpreting numerical data and contextual information.

In addition to the visual insights, the model was used to run a forward simulation process consisting of the following steps:

- Load the trained TFT model and historical process data.
- Generate a baseline forecast to identify the timing of potential exceedances in sensor 1.
- Evaluate thousands of reduction combinations in sensor values 2, 6, 10, 14, 18, 22, 26. ranging from 0 to 1000 per train with varying time lags
- For each scenario, modify the historical numbers accordingly and re-run the forecast.
- Identify the scenario with the minimal total reduction that still prevents the exceedance.

The output consists of CSV files containing detailed simulation results and an optimal reduction plan per train, specifying the timing, train-specific rate reductions, and the corresponding forecasted sensor one values. This system performs a series of "what-if" simulations to determine the most efficient operational response to predicted sensor one violations.

The GPT-4.1 Vision model processes the combined text and image input, allowing it to analyze the visual patterns in the attention plots and feature importance charts while simultaneously interpreting numerical data and contextual information.

To run the simulator promptly, I used the feature importance from the TFT to focus only on the trains that are the most impactful in the Sensor 1 prediction.

The need to focus is described in the following calculation:

- 7 trains × 5 reduction options (0, 25, 50, 75, 100) = 5^7 = 78,125 combinations. For EACH of 12 forecast time steps
- Total: 937,500 model predictions
- Time estimate:
    - Each model prediction: ~0.5-2 seconds (on CPU)
    - Total: 130-520 hours (5-22 days!)
    - This is not practical

A “smarter approach” was chosen.

First: we focus only on the trains that matter, by the feature importance output: Train 3 (Sensor 10), Train 6 (Sensor 22), and Train 7 (Sensor 26) flow with reduction options, starting from the lowest [50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600, 650, 700, 750, 800, 850, 900, 1000] and automatically stops when finds effective plan.

The model found the ideal reduction needed in the trains after 4,853 runs (02:34 minutes).

The output:

Table 4-7 - Simulator Results

Apply_At	Total_Reduction	Train_1_Reduction	Train_2_Reduction	Train_3_Reduction	Train_4_Reduction	Train_5_Reduction	Train_6_Reduction	Train_7_Reduction	Prevents_Exceedance_At	Resulting_prediction
19/6/25 12:43	600	0	0	550	0	0	0	50	19/6/25 13:13	5.39

4.2.3.1.3.6.3 The user message:

The following was added:

You are provided with:

```
- Deep learning model insights visualisations (png files)
- Optimal flowrate reduction plan generated by forward simulator (csv file)
```

This is the simulator results plan csv file:

```
{reduction_plan_text if reeduction_plan_text else ""}
```

Explaining the model of the two plots is given:

```
### Mean Attention Plot Analysis
**Description:** This plot shows the attention weights over time used by the Temporal Fusion Transformer (TFT) model. It visualizes how much importance the model assigns to different time steps from the input sequence when making a prediction. The x-axis represents time steps relative to the forecast point (past vs. future), and the y-axis shows the aggregated attention values. This helps identify which moments in the past the model paid more attention to when generating its forecast.

### Feature Importance Analysis
**Description:** This plot displays the variable selection weights used by the TFT model. It shows how the model distributes importance across the different input variables (features). Each bar corresponds to a variable's relative contribution, based on the learned gating mechanisms within TFT. This helps in understanding which features the model found most relevant for sensor 1 prediction.
```

Requesting the model to use the deep learning plots in the root cause analysis:

```
### 2. Root Cause Analysis
- Incorporate insights from the deep learning model insights visualizations.
```

Requesting the model to use the deep learning plots in the mitigation plan:

```
### 3. Corrective Actions
  - Incorporate insights from the deep learning model insights visualizations.
```

4.2.3.1.3.6.4 Run time

Each module (prediction on the last 96 minutes, explainability of the exceedance simulator, generating a report by LLM) was tested for run time separately; their combination should generally express that total run time, from prediction until the LLM has created the report.

Table 4-8 - Prediction metrics on the case study data set

Metric	Value	Description
MQL_P50	0.05130203	Mean Quantile Loss for P50 predictions (q=0.5)
MQL_P90	0.0597105	Mean Quantile Loss for P90 predictions (q=0.9)
MQL_P99	0.01644709	Mean Quantile Loss for P99 predictions (q=0.99)
Mean_MQL	0.04248654	Average of all quantile losses
Total_Predictions	12	Number of predictions with actual values

Table 4-9 - 190 predictions (1-minute strides) simulating performance in online forecasting

Metric	Value	Description
MQL_P50	0.06368012	Mean Quantile Loss for P50 predictions (q=0.5)
MQL_P90	0.02445042	Mean Quantile Loss for P90 predictions (q=0.9)
MQL_P99	0.00713681	Mean Quantile Loss for P99 predictions (q=0.99)
Mean_MQL	0.03175578	Average of all quantile losses
Total_Predictions	190	Number of predictions with actual values

- Predicting the future 12 minutes based on the historical 96 minutes – 1.7 seconds
- Calling Explainer to generate features importance and attention plot – 0 sec
- Forward simulator – 02:34 min.
- LLM generating the report – 02:07 min.
- Total time = 04:43 minutes.

4.2.3.1.3.6.5 SME response

"I think the strong point of this report and the most significant change from previous models is the analysis of the differences between the trains.
While the trains are theoretically identical in most regards, there can be differences in performance. It was interesting that the model shows different elements playing a stronger role in different trains. The recommended reduction is also accordingly by train, which is different from our normal reaction, but could in some cases cause less of a disturbance to production. The idea of the pre-emptive train balancing could be interesting to further explore, though it would require some effort to see how it would work with changing nominations.
For the root cause analysis, this report is quite similar to the last two versions in its insights, aside from the callout of specific trains here. The suggestion regarding additional smart sensors and controls is noteworthy, but I believe it would be beneficial to further investigate the predictions that can be made with the current data before going in that direction. Implementing new sensors and controls would require financial justification, such as evaluating return on investment based on potential savings from a reduction in deferred production."

4.2.4 Generated Code statistics for full model – deep learning and LLM

- Active codebase: 4,699 lines
- "Not using" Folder: 4,351 lines
- Total Project: ~9,050 lines of Python code

5 Discussion

5.1 Interpretation of Findings

5.1.1 The deep learning model

5.1.1.1 Availability of Data

Time series data was available and approachable, with almost no missing results (above 99% data availability). While using only 4.8 % of my available data (due to computational limitation) I was able to get 3.6 million samples, which is 8 times bigger than the biggest data set that was used in the original TFT paper (500K samples), this suggest that at least in the case study company there is enough data to run complex models such as deep learning. While this needs to be proved case by case, I can note that this is not a special case from my experience in the process industry, as sensors (and process historians) have been fundamental instruments in the process industry for many years.

5.1.1.2 Prediction Quality

Results of the training model have shown P50 quantile loss of 0.5724 (on CPU) and 0.1387 (on GPU) –in other words, on average, the median forecast is off by 0.5724 units in the direction of the error (or 0.1387 if we use the GPU model). The mistake in the peaks can be larger, suggesting that for a model aimed at predicting peaks/spikes, a dataset with more peaks/spikes might be beneficial. The dataset had 0.34% of the time results above 5.4 units (Sensor 2). Also, a more comprehensive hyperparameter optimization study could be helpful, as the survey lasts only five epochs.

5.1.1.3 Model Training on CPU vs GPU

The results show that training on a GPU gives better performance than training on a CPU. Even with identical hyperparameters, including float32 precision, fixed seed (42), and batch size, the GPU-trained model produced a lower p50 quantile loss. This likely reflects differences in underlying hardware execution: GPUs benefit from highly parallelized operations, which enable more stable and consistent training dynamics. In contrast, CPUs may process operations sequentially or less efficiently, leading to slightly different convergence paths and potentially suboptimal local minima.

I recommend using GPU-enabled environments for industrial time series forecasting. Modern models like the Temporal Fusion Transformer (TFT) can leverage GPU acceleration, which greatly reduces training and inference time, enabling faster adaptation to process changes. care must be taken to align library versions and dependencies between development and deployment environments, as mismatches (e.g., between GPU-trained and CPU-loaded models) can cause compatibility issues. If possible, a GPU training and execution environment is preferred.

5.1.1.4 Explainability of the Temporal Fusion Transformer

The temporal fusion transformer has built-in explainability that can show the temporal attention (when something important happened) and encoder feature importance (what features were important) relative to the last prediction. The results showed us (fig 14-15) that Sensor31, Sensor 20, and Sensor 10 were the most important in the exceedance prediction during the prediction window of 96 minutes, and the attention was the highest at timestamp ~ (-30) before the last prediction, meaning 30 minutes before time “0” (the present). Surprisingly, the data set had the same top 3 feature importance (in a different order) (Figure 4-11), even though there is a 2-year time difference between the data sets. This might suggest two main things: the first is that the

feature importance changes over time, and the second is that there might be some units that are consistently more important to the prediction than others.

Processes in the process industry are very complex; sometimes it is hard to identify the source of a phenomenon under investigation, due to the non-linear dynamics of the process. The results suggest that a TFT model can list feature importance and critical timestamps that lead to a deviation in a process, a potentially valuable resource to an engineer.

5.1.1.5 Computation Costs

The training consumes the most significant computational resources, lasting 16 hours on a personal computer (MacBook Pro M3, 16 GB RAM) for a data set containing 3,696,192 measurements over 119,232 timestamps. Running the prediction on the test data took 2.7 minutes for a data set with 410,688 measurements over 13,248 timestamps.

It is hard to categorize the TFT training process as “fast”; however, it is at least trainable in a reasonable amount of time (reason is subjective and is use case dependent) using publicly available computation resources.

Assuming a similar TFT is trained and predicts minute by minute in live production, the prediction time is faster, 1.76 seconds. The reason for that is the number of predictions. While in the training, we use prediction on an extensive data set to verify the model prediction along 13,248 timestamps (13,248/96 = 138 prediction windows) to measure performance comprehensively. In an “online” model, the prediction window equals the hyperparameter input_chunk_length, which is 96 minutes.

5.1.1.6 Noteworthy findings in the test dataset

It was interesting to find a visible difference between the absolute error in night vs day. Night times (20:00- 06:00) showed almost 2 times more absolute error than daytime (07:00 – 19:00). We saw that the absolute error was higher at the peaks and valleys. The dataset is a data measurement of summertime, which can have high differences in ambient temperature between night and day, which may affect the process. Also, production rate is related to user needs, which can be changed in different parts of the day.

5.1.2 Hybrid Approach: Multimodal Domain-Specific GPT with Deep Learning

5.1.2.1 The Dataset

The data set used for the hybrid approach is from different dates. The original model was trained and tested on data from June 1st to August 31, 2023. A fresh new data set was chosen for the hybrid approach to test the model's generalization. The data of the data set is June 19th, 2025.

The model was able to predict an exceedance, even within a 2-year difference in the data set, showing the generatability of the model.

The deep model was simulated in a real-world situation where it is running with strides of 1 minute. This means that every 1 minute, it predicts the next 12 minutes and moves 1 minute ahead. This simulates a real-world situation, and it is essential to simulate this because the model will behave differently based on its run time. The prediction started at 10:05, moving 1 minute each time, until 13:02 (and predicts the 12 minutes 13:03-13:14), and discovering an exceedance at exactly 13:40.

It is essential to say that based on the original test results of the model, it was noticed that the P50 tends to underpredict the peaks, while P99 was capturing the peaks, so the P99 results were used for the forecast (and generating 500 predictions instead of 100 in the training set), showing relatively low quantile loss of 0.007, in the specific dataset.

5.1.2.2 SME Response

I created a table summarizing the changes between her responses over each version to discuss the SME response. The blue highlight reflects the main change in the text relative to the previous SME response.

The comparison table suggests that the more information you add to the LLM, the better it gets.

Table 5-1 - SME responses comparison table

Version	Original Response	What Is New Compared to Previous
Naïve	***The report is accurate in identifying that the * generally decreases when the * decreases and that the correlation isn't linear. It is also correct in its analysis that other process conditions contribute to the increased *, and there is no prediction controlling or preventing exceedances. The reports suggest variations between * contributing to the changes, but generally, we don't see any significant differences in * composition between the * that would contribute to this.*** ***The general analysis performed provides information already available, and I don't feel it adds to our understanding of the issues within the system. However, the specific instructions for * reduction could be useful in limiting the production reduction, since today we respond by reducing production in a very basic manner with no variation between possibly different scenarios. If we can reduce the deferred production caused by being out of spec, it could be financially beneficial ."***	Baseline.
Naïve + Process	***"For the second report (Naive + Process Cheat Sheet), it maintained a good logical conclusion process. The model***	***Better at identifying the systems that are related to the problem.***

Version	Original Response	What Is New Compared to Previous
Cheat Sheet	***was better and more specific in identifying systems that can relate to the performance and to the rise in * compared to the first (naive) model. The interpretation of * fouling didn't seem adequately supported, as it also mentions a change in * that is much more directly related to the outlet *.*** ***While the report (and previous report) correctly mentions no prediction controlling or preventing exceedances, it only gives very general suggestions on how to improve and predict the exceedances, which do not necessarily add to our knowledge on the matter."***	
Naïve + Process Cheat Sheet + 5 Whys	***"For the Third report (Naive + Process Cheat Sheet + Five Whys)-*** ***Again, it is consistent with the others in presenting a clear and logical process. The model is better at identifying plausible reasons than the second (Naive + Process Cheat Sheet) report, which is important since there are a number of variables leading to an exceedance, and it is helpful to have different investigation angles called out. Additionally, the report is also more detailed in the corrective action suggestions."***	• Better at identifying the possible reasons that are causing the problem. • More detailed mitigation plan.
Naïve + Process Cheat Sheet + 5 Whys + ND	***"Though I cannot easily judge the accuracy of the prediction, it is helpful that this report gave more explanation as to why the specific * reduction was selected. The recommended reduction itself is also reduced compared to previous reports, which, if it indeed achieves the * spec, is significant in terms of our goal of minimizing disturbances to production.***	• More explanations on the reasoning that led to * reduction calculation. • Improved focus on the process, then on specific unit.

Version	Original Response	What Is New Compared to Previous
	I think the model is improved in its focus on the process more than specific equipment fouling, but I would have found it useful to leave some information on equipment that could play a factor in the exceedance, even if it would not explain the specific event. The report gave similar recommendations for preventative actions to the previous report."	
Naïve + Process Cheat Sheet + 5 Whys + ND + IE	***"As with the previous report, I appreciate the explanation it provides of the * reduction compared to earlier versions and that the numbers are once again reduced.*** ***The model focused on process parameters while also paying attention to instrumentation, which I believe is a notable improvement. Instrumentation could indeed be an important factor that was not adequately addressed in earlier reports.*** ***However, similar to the last report, I would have preferred it to include some information on equipment that could play a factor in the exceedance, even if it would not explain the specific event. The report gave similar recommendations for preventative actions to the previous report."***	Paying more attention to process instrumentation
Naïve + Process Cheat Sheet + 5 Whys + ND + IE + DP	***"I think the strong point of this report and the most significant change from previous models is the analysis of the differences between the trains.*** ***While the trains are theoretically identical in most regards, there can definitely be differences in performance. It was interesting to see that the model shows different elements playing a stronger role in different trains. The recommended reduction is also accordingly by train,***	• Analysis of the relative impact each train donate to the problem. • Recommended * reduction is per-train • Additional suggestions regarding smart sensors and smart controls

Version	Original Response	What Is New Compared to Previous
	which is different from our normal reaction, but could in some cases cause less of a disturbance to production. The idea of the pre-emptive train balancing could be interesting to further explore, though it would require some effort to see how it would work with changing nominations. ***For the root cause analysis, I find this report to be quite similar to the last two versions in its insights, aside from the callout of specific trains here as well. The suggestion regarding additional smart sensors and controls is noteworthy, but I believe it would be beneficial to further investigate the predictions that can be made with the current data before going in that direction. Implementing new sensors and controls would require financial justification, such as evaluating return on investment based on potential savings from a reduction in deferred production."***	

Below is a summary of key insights from the discussion paper:

- Sensor Data in the process industry exists, is accessible, and usable for deep training models.
- The prediction of the CPU model was good enough to use in this thesis, although much better performance is possible with different training approaches, mainly in the peaks and valleys.
- The TFT model has unique interpretability built into its architecture, which gives valuable insights that were not visible even to the SME without it.

- Using 96 past covariates to predict the next 12 minutes takes less than 2 seconds – this allows using the predictor in real-world applications.
- The SME response showed that the more knowledge you add to the domain-specific LLM, the better its answer gets. We can see a pattern of getting more reactions regarding the better insights the SME gets with each version. While the SME reaction for the Naïve version stated that the model gave an accurate report, suggesting that the LLM understands the problem and its role, the SME noted that it adds no new insights. This response was anticipated. By adding the LLM, the process knowledge (process cheat sheet), the SME responds that the model has become more specific in identifying potentially problematic systems that may contribute to the problem, suggesting that the model has gained new knowledge from the process cheat sheet. But still, the SME stated that, as in the previous report, there are no new insights they were unaware of. By adding the knowledge about creating a five whys analysis, the SME responded that the model gave a much more detailed five whys investigation, giving an edge at identifying plausible reasons for the root cause. The SME also stated that the mitigation plan was much more thorough. When adding the fundamentals textbook, the SME responded that the model is better focused on the system. When adding the instrument engineering textbook, the SME answered that she saw notable improvement in the response, as the LLM paid attention to the instrument in the process. The SME responded that while those are good and interesting responses, she would benefit more if the model could identify the specific equipment that could play a factor in the exceedance. When adding the deep learning outputs to the LLM, the SME responds that this report is the most significant change from the other reports, because it now gives a specific list of units impacting the exceedance, due to the interpretability capabilities of the TFT model. This looks like the central insight for the SME from all the other responses. While in all the previous five responds the SME stated positive respond, but no new insight – in the sixth report, that has the TFT interpretability outputs, the SME stated on a new knowledge added and a new insight was given, showing how much each train (and even

parameter within the train) is the most impactful on the exceedance. This new insight was not available to the SMEs without the model. This shows impressive added value to an experienced SME.

To conclude, it seems that it is possible to get new insights about a process from the process sensors alone – using a Temporal Fusion Transformer that learns the essential features of the process and moves them to the LLM that explains those to an engineer in a human-friendly way.

6 Conclusions and Future Research

6.1 Conclusions

This research proposes a novel architecture integrating a deep learning model (Temporal Fusion Transformer) for multivariate time series multi-step forecasting and a domain-specific GPT for interpretability and decision support in complex real-world operations. The system effectively predicted threshold exceedances using sensor data from a real-world process plant and translated these forecasts into actionable insights using prompt engineering and retrieval-augmented generation. Evaluation demonstrated that the TFT model achieved strong predictive performance, especially when trained on GPU hardware. LLM insight quality improved as more domain knowledge was added, especially when insights from a deep learning model are added. Working together, these models form a scalable framework for real-time operational guidance, offering potential to improve proactive decision-making, safety, and compliance in complex industrial environments. There is a learning process of interacting with a deep learning model for time-series and the domain-specific GPT. Both are at the edge of innovation, so it is no surprise that there is not much available guidance literature; it was mainly auto didactive work. The two main learning processes are the training of the TFT and the prompt engineering of the LLM. The temporal fusion transformer's actual value is its explanatory feature, which gives valuable insight into the features that play the key factors in the problem, and the specific point in time at which it happened. This is unique and insightful, as was also stated by the SME. This kind of information is key to understanding process dynamics for the process industry, specifically in a complex environment.

6.2 Limitations

Several limitations should be noted. First, the models were trained and tested on historical data; no live integration with plant systems was attempted. Second, while the dataset represents real-world operations, all identifying metadata was removed and numerical values transformed to

preserve confidentiality, potentially limiting the external generalizability of the specific results. Third, due to hardware limitations, the TFT model was not trained on the full 3.5-year dataset but on a subset of about ~4% of the complete data. Finally, while the domain-specific GPT showed promising results in synthesizing actionable insights, its numerical reasoning remains constrained by known LLM limitations. Creating the TFT and the LLM requires computational resources in terms of time and hardware. The main efforts in building those models are the following:

- Collecting the data from the owner takes a considerable amount of time, due to the coordination that is needed with different people: multiple meetings with the SME to first understand the process and decide which sensors are relevant to use (and what are their tags), and later you should approach and ask for this data. For the sensor data, you would need the help of the IT department because the scale of the information within the files is enormous. I was aiding with the IT as the SME said that extracting this amount of information using their tools is not practical and will take lots of time. The IT has tools that can do it faster. But even then, you must ensure you have all the correct data. I have requested many fixes from the IT department regarding the data they gave me.
- Regarding the information for the LLM, there is a privacy issue. The problem is that companies are suspicious about the LLM learning for their information and might show results to competitors based on the information learnt. So, generally, companies do not accept adding process descriptions and official procedures to the LLM. So, if one is trying to create a domain-specific LLM, one should contact cybersecurity experts to understand what is possible and what is not. After getting the data, you must change the numbers to export it to your computer, as companies do not agree for any process data to go out of their system, so some function should be implemented on the sensory data (or any other data) within 4 weeks.
- Training the model takes considerable time because it takes many hours to get the results, even on a GPU. And you should train many times until you are satisfied with the results.

This is a trial-and-error journey, where the time between each version can be half a day. Train – 8-10 hours – look at results, change hyperparameters – train again. – 2 weeks.

- Prompt engineering the LLM – this is not a science. Prompt engineering is writing instructions to the LLM in the system message and looking at its response. Most of the time, it is not exactly what you wanted, so you update the prompt and do so until you are happy with the results. This can take many versions; for example, making the LLM add references to his professional literature in the rag system when he used them took me 59 versions. -1 week
- Writing the code and debugging – much time was consumed for writing code and debugging. This is a code that has multiple external libraries and many modules. 3 weeks. It is not easy to copy and paste code for your next project. Each project should be given a code based on the case. However, making more models will probably be faster with time.
- Evaluating the model with the SME – 1 week.
- For most cases, this is a waterfall type project as you cannot proceed forward before finishing the previous stage (for example, you can't train the model of you don't have the data, or you can't prompt engineer in you didn't build the whole code environment for this) there is definitely a learning curve for this kind of projects. Still, due to the waterfall and dependency on other people, there is a limit to how fast you can do it.
- Overall, building this model end-to-end would take 3 months of a full-time engineer. It should also be noted that this model also consumes the company time (the SME, IT, and operations point of contact)
- ROI – this depends on the case. In the specific case for the thesis, assuming the model helps with preventing or at least decreasing the time the system is in exceedance will results in swift ROI -, due to the scale of the system and the nature of the particular material, it will probably need at least a year to understand if the in the long term it make contribution, so if it is – we can say that ROI can be 1 year.

- Generally, for ROI in this model. This model should be placed as close to the money-making source as possible. Placing this model in an HR business might give valuable insights, but ROI can be long-term or mayn't be reached at all. Placing this model to prevent threshold exceedances, which results in losing millions of dollars per event, is much more reasonable due to the time it takes to build such a model.

6.3 Future Research

- It was surprising to find that using deep learning techniques on process sensors from the process industry is very limited in the published literature. From a technical point of view, it is a "perfect match". But even so, little to no research has been done. The reasons might be data availability, as this information is sensitive and might have IP restrictions on him. However, I hope that the positive outcomes of this research will drive new studies in the following matters:
    - The suitable architecture of a model deployed in live production (deep learning alone or deep learning with domain-specific GPT). Attention should be given to the training. What training frequency is required? Once a day? Only one time? How will the results be seen? Dashboard? Written report?
    - Searching for different deep learning models that can predict even better than the proposed Temporal Fusion Transformer.
    - The implication of training a model on a multi-year data set (tens of millions of measurements) is that this model used a two-month data set. A whole year of training might let the model learn seasonally related patterns.
    - More study is needed to find techniques to help these models predict better at the peaks.?

- Feature engineering should be explored, as this study did not use it. Feature engineering is a fundamental technique in data sciences, and it will be valuable to see its impact on the model performance.
- Using LLMs is still a challenge regarding data privacy. Industry today is acting causally with LLMs because they might be trained on data that is transferred to them/. Investigation on using a “offline” LLM company LLM can be valuable,

- Potential implication of this model, and where one can use it:
  - This model can be used everywhere there is large amount of sensory data, and that you understand the underlying process you want to explore.
  - Using the TFT prediction even alone, can give insight on a process in ways that might not be available without him. for example, you can train the model to predict your production rate, and using the TFT explainability – you might get more insights about which one of the features effect the most on you production rate?
  - Another way to use the model, is with prediction in the environmental sector. Trying to predict an environmental exceedance in a wastewater, should not be different. Sensors are widely used for monitoring quality of streams, and they can be used, together with process sensors to predict and alert before environmental exceedance happens. In terms of ROI – environmental violation has high revenue implications in forms of multimillion dollar fines, reputation loss, time loss due to investigations and explanations to the ministry of environmental protection.

References

[1] J. F. Cox, “APICS dictionary”.

[2] “Process Industries Division - Institute of Industrial and Systems Engineers.” Accessed: May 28, 2025. [Online]. Available: https://www.iise.org/details.aspx?id=887

[3] “Process automation handbook: a guide to theory and practice,” *Choice Rev. Online*, vol. 45, no. 09, pp. 45-5013-45–5013, May 2008, doi: 10.5860/CHOICE.45-5013.

[4] M. Bortz *et al.*, “AI in Process Industries – Current Status and Future Prospects,” *Chem. Ing. Tech.*, vol. 95, no. 7, pp. 975–988, July 2023, doi: 10.1002/cite.202200247.

[5] S.-L. Jämsä-Jounela, “Future trends in process automation,” *Annu. Rev. Control*, vol. 31, no. 2, pp. 211–220, Jan. 2007, doi: 10.1016/j.arcontrol.2007.08.003.

[6] T. Komulainen, M. Sourander, and S.-L. Jämsä-Jounela, “An online application of dynamic PLS to a dearomatization process,” *Comput. Chem. Eng.*, vol. 28, no. 12, pp. 2611–2619, Nov. 2004, doi: 10.1016/j.compchemeng.2004.07.014.

[7] J. Wang, W. Zhang, Y. Shi, and S. Duan, “Industrial Big Data Analytics: Challenges, Methodologies, and Applications”.

[8] J. Mietkiewicz *et al.*, “Enhancing Control Room Operator Decision Making,” *Processes*, vol. 12, no. 2, p. 328, Feb. 2024, doi: 10.3390/pr12020328.

[9] A. Casolaro, V. Capone, G. Iannuzzo, and F. Camastra, “Deep Learning for Time Series Forecasting: Advances and Open Problems,” *Information*, vol. 14, no. 11, p. 598, Nov. 2023, doi: 10.3390/info14110598.

[10] X. Zhang *et al.*, “Automated Root Causing of Cloud Incidents using In-Context Learning with GPT-4,” Jan. 24, 2024, *arXiv*: arXiv:2401.13810. doi: 10.48550/arXiv.2401.13810.

[11] M. Raza, Z. Jahangir, M. B. Riaz, M. J. Saeed, and M. A. Sattar, “Industrial applications of large language models,” *Sci. Rep.*, vol. 15, no. 1, p. 13755, Apr. 2025, doi: 10.1038/s41598-025-98483-1.

[12] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri, “A review of process fault detection and diagnosis Part I: Quantitative model-based methods”.

[13] “Senseye Predictive Maintenance,” siemens.com Global Website. Accessed: Mar. 31, 2025. [Online]. Available: https://www.siemens.com/global/en/products/services/digital-enterprise-services/analytics-artificial-intelligence-services/senseye-predictive-maintenance.html

[14] B. G. Lipták and H. Eren, Eds., *Instrument engineers’ handbook*, 4th ed. Boca Raton, FL: CRC/Taylor & Francis, 2003.

[15] F. S. Al-Anzi, H. M. S. Lababidi, G. Al-Sharrah, S. A. Al-Radwan, and H. J. Seo, “Plant health index as an anomaly detection tool for oil refinery processes,” *Sci. Rep.*, vol. 12, no. 1, p. 14477, Aug. 2022, doi: 10.1038/s41598-022-18824-2.

[16] B. G. Lipták, Ed., “Instrument Engineers’ Handbook. 3: Process software and digital networks / Béla G. Lipták, editor-in-chief,” 3 ed., Boca Raton, FL: CRC Press, 2002.

[17] “PI System™ | AVEVA.” Accessed: June 22, 2025. [Online]. Available: https://www.aveva.com/en/products/aveva-pi-system/

[18] “mgi_big_data_full_report.”

[19] “the_rise_of_industrial_big_data_wp_gft834.”

[20] X. Kong *et al.*, “Deep Learning for Time Series Forecasting: A Survey,” *Int. J. Mach. Learn. Cybern.*, Feb. 2025, doi: 10.1007/s13042-025-02560-w.

[21] L. Su, X. Zuo, R. Li, X. Wang, H. Zhao, and B. Huang, "A systematic review for transformer-based long-term series forecasting," *Artif. Intell. Rev.*, vol. 58, no. 3, p. 80, Jan. 2025, doi: 10.1007/s10462-024-11044-2.
[22] K. Benidis *et al.*, "Deep Learning for Time Series Forecasting: Tutorial and Literature Survey," *ACM Comput. Surv.*, vol. 55, no. 6, pp. 1–36, July 2023, doi: 10.1145/3533382.
[23] A. Sagheer and M. Kotb, "Time series forecasting of petroleum production using deep LSTM recurrent networks," *Neurocomputing*, vol. 323, pp. 203–213, Jan. 2019, doi: 10.1016/j.neucom.2018.09.082.
[24] Y. Song, S. Gao, Y. Li, L. Jia, Q. Li, and F. Pang, "Distributed Attention-Based Temporal Convolutional Network for Remaining Useful Life Prediction," *IEEE Internet Things J.*, vol. 8, no. 12, pp. 9594–9602, June 2021, doi: 10.1109/JIOT.2020.3004452.
[25] A. Bala, I. Ismail, R. Ibrahim, S. M. Sait, and D. Oliva, "An Improved Grasshopper Optimization Algorithm Based Echo State Network for Predicting Faults in Airplane Engines," *IEEE Access*, vol. 8, pp. 159773–159789, 2020, doi: 10.1109/ACCESS.2020.3020356.
[26] Z. Li, Z. Zheng, and R. Outbib, "Adaptive Prognostic of Fuel Cells by Implementing Ensemble Echo State Networks in Time-Varying Model Space," *IEEE Trans. Ind. Electron.*, vol. 67, no. 1, pp. 379–389, Jan. 2020, doi: 10.1109/TIE.2019.2893827.
[27] J. Macintyre, L. Iliadis, I. Maglogiannis, and C. Jayne, Eds., *Engineering Applications of Neural Networks: 20th International Conference, EANN 2019, Xersonisos, Crete, Greece, May 24-26, 2019, Proceedings*, vol. 1000. in Communications in Computer and Information Science, vol. 1000. Cham: Springer International Publishing, 2019. doi: 10.1007/978-3-030-20257-6.
[28] Ashish Vaswani *et al.*, "Attention is All you Need," *Neural Inf. Process. Syst.*, vol. 30, pp. 5998–6008, June 2017.
[29] J. Kaplan *et al.*, "Scaling Laws for Neural Language Models," Jan. 23, 2020, *arXiv*: arXiv:2001.08361. doi: 10.48550/arXiv.2001.08361.
[30] J. Wei *et al.*, "Emergent Abilities of Large Language Models," Oct. 26, 2022, *arXiv*: arXiv:2206.07682. doi: 10.48550/arXiv.2206.07682.
[31] C. Ling *et al.*, "Domain Specialization as the Key to Make Large Language Models Disruptive: A Comprehensive Survey," Mar. 29, 2024, *arXiv*: arXiv:2305.18703. doi: 10.48550/arXiv.2305.18703.
[32] S. K. Freire, C. Wang, M. Foosherian, S. Wellsandt, S. Ruiz-Arenas, and E. Niforatos, "Knowledge Sharing in Manufacturing using Large Language Models: User Evaluation and Model Benchmarking," Feb. 26, 2024, *arXiv*: arXiv:2401.05200. doi: 10.48550/arXiv.2401.05200.
[33] Y. Xia, M. Shenoy, N. Jazdi, and M. Weyrich, "Towards autonomous system: flexible modular production system enhanced with large language model agents," in *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sinaia, Romania: IEEE, Sept. 2023, pp. 1–8. doi: 10.1109/ETFA54631.2023.10275362.
[34] K. F. ana G. Demartini, "MiningGPT -- A Domain-Specific Large Language Model for the Mining Industry," Dec. 02, 2024, *arXiv*: arXiv:2412.01189. doi: 10.48550/arXiv.2412.01189.
[35] Q. Sun, Y. Li, C. Zhou, and Y.-C. Tian, "Root Cause Analysis for Industrial Process Anomalies through the Integration of Knowledge Graph and Large Language Model," in

2024 43rd Chinese Control Conference (CCC), Kunming, China: IEEE, July 2024, pp. 6855–6860. doi: 10.23919/CCC63176.2024.10662704.
[36] H. Li *et al.*, "Exposing Numeracy Gaps: A Benchmark to Evaluate Fundamental Numerical Abilities in Large Language Models," June 03, 2025, *arXiv*: arXiv:2502.11075. doi: 10.48550/arXiv.2502.11075.
[37] "IEEE Std 754™-2019 (Revision of IEEE Std 754-2008) IEEE Standard for Floating-Point Arithmetic," 2019.
[38] "Time-series forecasting with deep learning_ a survey."
[39] B. Lim, S. O. Arik, N. Loeff, and T. Pfister, "Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting," Sept. 27, 2020, *arXiv*: arXiv:1912.09363. doi: 10.48550/arXiv.1912.09363.
[40] *unit8co/darts*. (July 12, 2025). Python. Unit8 SA. Accessed: July 12, 2025. [Online]. Available: https://github.com/unit8co/darts
[41] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Jan. 30, 2017, *arXiv*: arXiv:1412.6980. doi: 10.48550/arXiv.1412.6980.
[42] "GPT-4.1 Prompting Guide | OpenAI Cookbook." Accessed: July 10, 2025. [Online]. Available: https://cookbook.openai.com/examples/gpt4-1_prompting_guide#delimiters
[43] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large Language Models are Zero-Shot Reasoners," Jan. 29, 2023, *arXiv*: arXiv:2205.11916. doi: 10.48550/arXiv.2205.11916.
[44] J. Wei *et al.*, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," Jan. 10, 2023, *arXiv*: arXiv:2201.11903. doi: 10.48550/arXiv.2201.11903.
[45] "The Five Whys Technique," in *Knowledge Solutions*, Singapore: Springer Singapore, 2017, pp. 307–310. doi: 10.1007/978-981-10-0983-9_32.
[46] "Hyperparameter Optimization in Darts — darts documentation." Accessed: Aug. 01, 2025. [Online]. Available: https://unit8co.github.io/darts/userguide/hyperparameter_optimization.html
[47] "Torch Forecasting Models — darts documentation." Accessed: Aug. 08, 2025. [Online]. Available: https://unit8co.github.io/darts/userguide/torch_forecasting_models.html#trainingsaving-on-gpu-and-loading-on-cpu
[48] "Model - OpenAI API." Accessed: July 31, 2025. [Online]. Available: https://platform.openai.com
[49] "Five whys," *Wikipedia*. July 23, 2025. Accessed: July 31, 2025. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Five_whys&oldid=1302085056
[50] "OpenAI Model Spec." Accessed: July 31, 2025. [Online]. Available: https://model-spec.openai.com/2025-04-11.html#chain_of_command
[51] "Markdown," *Wikipedia*. July 14, 2025. Accessed: Aug. 01, 2025. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Markdown&oldid=1300506913
[52] O. Serrat, "The Five Whys Technique," in *Knowledge Solutions*, Singapore: Springer Singapore, 2017, pp. 307–310. doi: 10.1007/978-981-10-0983-9_32.

Appendix A

Year	Expected Frequency	Expected Timestamps	Actual Timestamps	Missing Timestamps	Duplicate Timestamps
2022	1.00 min	525,600	525,600	0	0
2023	1.00 min	525,600	525,600	0	0
2024	1.00 min	527,040	527,040	0	0
2025(Jan-July 3 23:59)	1.00 min	264,959	264,959	0	0
ALL YEARS	1.00 min	1,843,199	1,843,199	0	0

Year	Total Cells	Time Steps Count	Time Steps %	Column Headers Count	Column Headers %	Non-Zero Numeric Count	Non-Zero Numeric %	Zero Numeric Count	Zero Numeric %	Text Values Count	Text Values %	Empty Cells Count	Empty Cells %	Total %
2022	16,819,200	525,600	3.12%	32	0.00%	15,052,552	89.50%	866,147	5.15%	374,870	2.23%	0	0.00%	100.00%
2023	16,819,200	525,600	3.12%	32	0.00%	14,439,749	85.85%	1,744,487	10.37%	109,333	0.65%	0	0.00%	100.00%
2024	16,865,280	527,040*	3.12%	32	0.00%	15,163,774	89.91%	1,131,903	6.71%	42,532	0.25%	0	0.00%	100.00%
2025	8,476,800	264,959	3.12%	32	0.00%	8,015,922	94.56%	167,021	1.97%	28,926	0.34%	0	0.00%	100.00%
ALL YEARS	58,980,480	1,843,140	3.12%	32	0.00%	52,672,087	89.30%	3,909,561	6.63%	555,661	0.94%	0	0.00%	100.00%

* Note: 2024 is a leap year, so it has 366 days, which equals 527,040 minutes

- Ready to model cells = (Non-Zero Numeric Count Zero Numeric Count) / (Total Cells - Time Steps Count- Column Headers Count - Empty Cells Count)
- Example for 2022: $\frac{15,052,552+866,147}{16,819,200-525,600-32-0} = 0.976 = 97.6\%$
- Ready to model cells: 2022 = 97.7%, 2023 = 99.33%, 2024 = 99.74%, 2025 = 99.65%, All years = 99.03%

Text Value	2022	2023	2024	2025	ALL YEARS
Bad	311	2265	182	29	2787
Comm Fail	193724	80560	20019	0	294303
Configure	311	0	0	0	311
I/O Timeout	59986	3605	5072	28503	97166
Intf Shut	110674	22903	17259	394	151230
Not Connect	9864	0	0	0	9864

Appendix B

Knowledge Solutions

February 2009 | 30

The Five Whys Technique

by Olivier Serrat

When confronted with a problem, have you ever stopped and asked "why" five times? If you do not ask the right question, you will not get the right answer. The Five Whys is a simple question-asking technique that explores the cause-and-effect relationships underlying problems.

Rationale

For every effect there is a cause. But the results chain between the two is fairly long and becomes finer as one moves from inputs to activities, outputs, outcome, and impact.[1] In results-based management,[2] the degree of control one enjoys decreases higher up the chain and the challenge of monitoring and evaluating correspondingly increases.

In due course, when a problem appears, the temptation is strong to blame others or external events. Yet, the root cause of problems often lies closer to home.

The Five Whys Technique

When looking to solve a problem, it helps to begin at the end result, reflect on what caused that, and question the answer five times.[3] This elementary and often effective approach to problem solving promotes deep thinking through questioning, and can be adapted quickly and applied to most problems.[4] Most obviously and directly, the Five Whys technique relates to the principle of systematic problem-solving: without the intent of the principle, the technique can only be a shell of the process. Hence, there are three key elements to effective use of the Five Whys technique: (i) accurate and complete statements of problems,[5] (ii) complete honesty in answering the questions, (iii) the determination to get to the bot-

---

1 Inputs, activities, and outputs are within the direct control of an intervention's management. An outcome is what an intervention can be expected to achieve and be accountable for. An impact is what an intervention is expected to contribute to.

2 Results-based management is a life-cycle management philosophy and approach that emphasizes results in integrated planning, implementing, monitoring, reporting, learning, and changing. Demonstrating results is important for credibility, accountability, and continuous learning, and to inform decision-making and resource allocation.

3 Five is a good rule of thumb. By asking "why" five times, one can usually peel away the layers of symptoms that hide the cause of a problem. But one may also find one needs to ask "why" fewer times, or conversely more.

4 Root cause analysis is the generic name of problem-solving techniques. The basic elements of root causes are materials, equipment, the man-made or natural environment, information, measurement, methods and procedures, people, management, and management systems. Other tools can be used if the Five Whys technique does not intuitively direct attention to one of these. They include barrier analysis, change analysis, causal factor tree analysis, and the Ishikawa (or fishbone) diagram.

5 By repeating "why" five times, the nature of the problem as well as its solution becomes clear.

ADB Asian Development Bank

tom of problems and resolve them. The technique was developed by Sakichi Toyoda for the Toyota Industries Corporation.

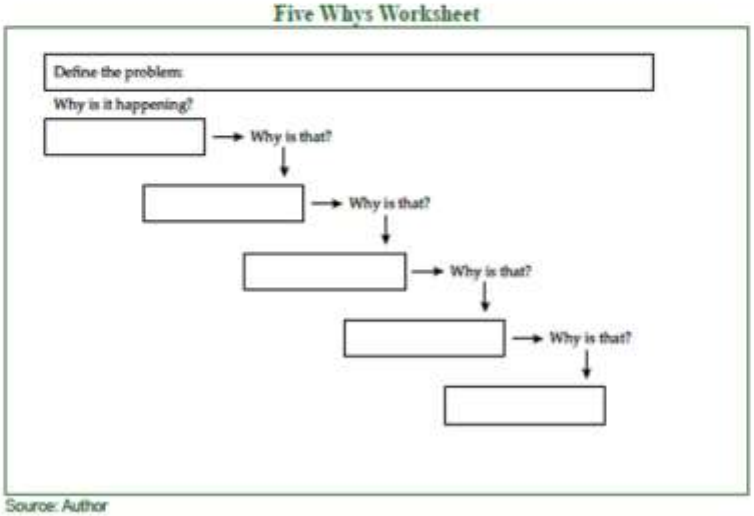
For Want of a Nail
For want of a nail the shoe is lost;
For want of a shoe the horse is lost;
For want of a horse the rider is lost;
For want of a rider the battle is lost;
For want of a battle the kingdom is lost;
And all for the want of a horseshoe nail.
—George Herbert

Process

The Five-Whys exercise is vastly improved when applied by a team and there are five basic steps to conducting it:

- Gather a team and develop the problem statement in agreement. After this is done, decide whether or not additional individuals are needed to resolve the problem.
- Ask the first "why" of the team: why is this or that problem taking place? There will probably be three or four sensible answers: record them all on a flip chart or whiteboard, or use index cards taped to a wall.
- Ask four more successive "whys," repeating the process for every statement on the flip chart, whiteboard, or index cards. Post each answer near its "parent." Follow up on all plausible answers. You will have identified the root cause when asking "why" yields no further useful information. (If necessary, continue to ask questions beyond the arbitrary five layers to get to the root cause.)
- Among the dozen or so answers to the last asked "why" look for systemic causes of the problem. Discuss these and settle on the most likely systemic cause. Follow the team session with a debriefing and show the product to others to confirm that they see logic in the analysis.
- After settling on the most probable root cause of the problem and obtaining confirmation of the logic behind the analysis, develop appropriate corrective actions to remove the root cause from the system. The actions can (as the case demands) be undertaken by others but planning and implementation will benefit from team inputs.

Five Whys Worksheet



Source: Author

Caveat

The Five Whys technique has been criticized as too basic a tool to analyze root causes to the depth required to ensure that the causes are fixed. The reasons for this criticism include:

- The tendency of investigators to stop at symptoms, and not proceed to lower-level root causes.
- The inability of investigators to cast their minds beyond current information and knowledge.
- Lack of facilitation and support to help investigators ask the right questions.
- The low repeat rate of results: different teams using the Five Whys technique have been known to come up with different causes for the same problem.

Jeff Bezos and Root Cause Analysis

[The author explains how while he worked for Amazon.com in 2004 Jeff Bezos did something that the author still carries with him to this day. During a visit the Amazon.com Fulfillment Centers, Jeff Bezos learned of a safety incident during which an associate had damaged his finger. He walked to the whiteboard and began to use the Five Whys technique.]

Why did the associate damage his thumb?

Because his thumb got caught in the conveyor.

Why did his thumb get caught in the conveyor?

Because he was chasing his bag, which was on a running conveyor.

Why did he chase his bag?

Because he had placed his bag on the conveyor, which had then started unexpectedly.

Why was his bag on the conveyor?

Because he was using the conveyor as a table.

And so, the root cause of the associate's damaged thumb is that he simply needed a table. There wasn't one around and he had used the conveyor as a table. To eliminate further safety incidences, Amazon.com needs to provide tables at the appropriate stations and update safety training. It must also look into preventative maintenance standard work.

Source: Adapted from Shmula. 2008. Available: www.shmula.com/

Clearly, the Five Whys technique will suffer if it is applied through deduction only. The process articulated earlier encourages on-the-spot verification of answers to the current "why" question before proceeding to the next, and should help avoid such issues.

Further Reading

ADB. 2007. *Guidelines for Preparing a Design and Monitoring Framework*. Manila. Available: www.adb.org/documents/guidelines/guidelines-preparing-dmf/guidelines-preparing-dmf.pdf

———. 2008a. *Output Accomplishment and the Design and Monitoring Framework*. Manila. Available: www.adb.org/documents/information/knowledge-solutions/output-accomplishment.pdf

———. 2008b. *The Reframing Matrix*. Manila. Available: www.adb.org/documents/information/knowledge-solutions/the-reframing-matrix.pdf

———. 2009a. *Monthly Progress Notes*. Manila. Available: www.adb.org/documents/information/knowledge-solutions/monthly-progress-notes.pdf

———. 2009b. *Assessing the Effectiveness of Assistance in Capacity Development*. Manila. Available: http://www.adb.org/documents/information/knowledge-solutions/assessing-effectiveness-assistance-capacity-development.pdf

Knowledge Solutions

For further information

Contact Olivier Serrat, Head of the Knowledge Management Center, Regional and Sustainable Development Department, Asian Development Bank (oserrat@adb.org).

Asian Development Bank

ADB, based in Manila, is dedicated to reducing poverty in the Asia and Pacific region through inclusive economic growth, environmentally sustainable growth, and regional integration. Established in 1966, it is owned by 67 members—48 from the region. In 2007, it approved $10.1 billion of loans, $673 million of grant projects, and technical assistance amounting to $243 million.

Knowledge Solutions are handy, quick reference guides to tools, methods, and approaches that propel development forward and enhance its effects. They are offered as resources to ADB staff. They may also appeal to the development community and people having interest in knowledge and learning.

Asian Development Bank
6 ADB Avenue, Mandaluyong City
1550 Metro Manila, Philippines
Tel +63 2 632 4444
Fax +63 2 636 2444
knowledge@adb.org
www.adb.org/knowledgesolutions