Self-Supervised Prediction Contrast Time Frequency for Industrial Fault Detection

by

Hamza Görgülü

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Computer Science and Engineering

KOÇ ÜNİVERSİTESİ

December 16, 2024

Self-Supervised Prediction Contrast Time Frequency for Industrial Fault Detection

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Hamza Görgülü

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Prof. Öznur Özkasap

_____

Assoc. Prof. Murat Karakuş

_____

Assist. Prof. Barış Akgün

Date: _____

I dedicate this thesis to my beloved mother, my family, my thesis advisor Prof. Dr. Öznur Özkasap, and to my friends for their constant encouragement throughout this journey.

ABSTRACT

Self-Supervised Prediction Contrast Time Frequency for Industrial Fault Detection
Hamza Görgülü
Master of Science in Computer Science and Engineering
December 16, 2024

Industrial informatics produce vast data, but fault detection is hindered by dynamic processes and non-linear interactions. Deep learning and machine learning help, but sensor errors, transmission failures, and scarce labels remain issues. Self-supervised learning (SSL) uses unlabeled data to train models and generate labels. Prediction contrast learning, a key SSL method, preserves temporal patterns, making it ideal for time-series data while avoiding constant parameter tuning required by traditional methods. To tackle fault detection task, this thesis proposes a novel self-supervised learning architecture using prediction contrast learning to enhance fault detection (FD) in industrial settings. Our approach integrates time and frequency domain information to capture crucial temporal dependencies and improve model robustness. The key contributions of this work include: the novel combination of time-frequency information in SSL, the first application of prediction contrast learning for FD, a complex network analysis for real-time performance and scalability, and an extensive evaluation including ablation studies. To address the challenges of data scarcity and enhance fault detection accuracy in complex environments, our proposed model incorporates data transformations that improve robustness while preserving essential temporal relationships for time series analysis. We evaluated our approach using the Tennessee Eastman Process (TEP) benchmark datasets, including both Ricker and Rieth versions, which are widely recognized standards for testing FD systems in industrial processes. We also evaluate our model in Secure Water Treatment (SWaT) and Water Distribution (WADI) dataset, which contain anomalies such as intentional attack scenarios injected into the water management system.

Experimental results on the TEP datasets demonstrate that our approach significantly outperforms existing benchmark models, achieving a True Positive Rate (TPR) of 92.04% and reducing the Average Detection Delay(ADD) to 22.18 steps,

Correct Diagnosis Rate (CDR) of 97.62% effectively addressing data scarcity challenges and enhancing fault detection accuracy in complex industrial environments. Similar outperforming results have also been observed in other benchmark datasets. To ensure practical applicability, we conduct complex network topology analysis to evaluate our model's real-time performance and scalability in both cloud and edge computing environments, considering industrial networks as complex systems with non-trivial topological features. This analysis provides a comprehensive comparison of latency differences between edge and cloud computing architectures, addressing a critical gap in the fault detection literature. The proposed SSL architecture, combined with our comprehensive network analysis, offers a robust solution for FD in industrial chemical facilities. By integrating advanced deep learning techniques with practical considerations for implementation in various computing environments, our work paves the way for more efficient and reliable fault detection systems in industry.

ÖZETÇE

Endüstriyel Hata Tespiti için Özdenetimli Tahmin Karşıtlığı Zaman-Frekans Yöntemi
Hamza Görgülü
Bilgisayar Mühendisliği, Yüksek Lisans
16 Aralık 2024

Endüstriyel bilişim, büyük miktarda veri üretir, ancak hata tespiti dinamik süreçler ve doğrusal olmayan etkileşimler nedeniyle zorluklarla karşılaşır. Derin öğrenme ve makine öğrenimi bu konuda yardımcı olsa da, sensör hataları, iletim hataları ve sınırlı etiketli veriler gibi sorunlar devam etmektedir. Özdenetimli öğrenme (SSL), etiketlenmemiş verileri kullanarak modelleri eğitip etiketler oluşturabilen bir yaklaşımdır. SSL'nin önemli bir yöntemi olan tahmin karşıtlığı öğrenme, zaman serisi verilerindeki zamansal ilişkileri koruyarak parametre ayarlamaya gerek kalmadan daha verimli bir çözüm sunar. Bu tezde, endüstriyel ortamlarda hata tespiti (FD) yeteneğini artırmak amacıyla tahmin karşıtlığı öğrenmesini kullanan yenilikçi bir özdenetimli öğrenme mimarisi önerilmektedir. Önerilen yaklaşım, zamansal bağımlılıkları yakalamak ve model dayanıklılığını artırmak için zaman ve frekans alanı bilgilerini birleştirmektedir. Bu çalışmanın temel katkıları şunlardır: SSL'de zaman-frekans bilgisinin yenilikçi bir şekilde birleştirilmesi, hata tespiti için tahmin karşıtlığı öğrenmenin ilk kez uygulanması, gerçek zamanlı performans ve ölçeklenebilirlik için karmaşık ağ analizi ve kapsamlı bir değerlendirme sürecinin (ablation çalışmaları dahil) gerçekleştirilmesi. Veri kıtlığı sorunlarını ele almak ve karmaşık ortamlarda hata tespit doğruluğunu artırmak amacıyla önerilen model, temel zamansal ilişkileri korurken dayanıklılığı artıran veri dönüşümleri içermektedir. Yaklaşımımız, endüstriyel süreçlerde hata tespit sistemlerini test etmek için yaygın olarak kullanılan Tennessee Eastman Süreci (TEP) referans veri setleri (Ricker ve Rieth versiyonları) ile değerlendirildi. Ayrıca, su yönetim sistemine kasıtlı saldırı senaryoları enjekte edilmiş anormallikler içeren Secure Water Treatment (SWaT) ve Water Distribution (WADI) veri setleri üzerinde de değerlendirmeler gerçekleştirilmiştir.

TEP veri setleri üzerindeki deneysel sonuçlar, önerilen yaklaşımın mevcut referans modelleri önemli ölçüde geride bıraktığını göstermektedir. Modelimiz, %92,04

Doğru Pozitif Oranı (TPR), 22,18 adım Ortalama Tespit Gecikmesi (ADD) ve %97,62 Doğru Tanı Oranı (CDR) elde ederek veri kıtlığı zorluklarını etkili bir şekilde ele almış ve karmaşık endüstriyel ortamlarda hata tespit doğruluğunu artırmıştır. Benzer şekilde, diğer referans veri setlerinde de üstün performans gözlemlenmiştir. Pratik uygulanabilirliği sağlamak için, modelimizin gerçek zamanlı performansını ve ölçeklenebilirliğini hem bulut hem de uç bilgi işlem ortamlarında değerlendirmek amacıyla karmaşık ağ topolojisi analizi gerçekleştirilmiştir. Bu analiz, endüstriyel ağları karmaşık sistemler olarak ele alarak, uç ve bulut bilgi işlem mimarileri arasındaki gecikme farklarının kapsamlı bir karşılaştırmasını sunmaktadır. Bu yaklaşım, hata tespit literatüründe önemli bir boşluğu doldurmaktadır. Önerilen özdenetimli öğrenme mimarisi, kapsamlı ağ analizi ile birleştirildiğinde, endüstriyel kimya tesislerinde hata tespiti için sağlam bir çözüm sunmaktadır. Gelişmiş derin öğrenme tekniklerinin çeşitli bilgi işlem ortamlarında uygulanmasına yönelik pratik hususlarla entegre edilmesi sayesinde, endüstride daha verimli ve güvenilir hata tespit sistemlerinin geliştirilmesinin yolu açılmaktadır.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor, Prof. Öznur Özkasap, for her unwavering support, guidance, and encouragement throughout this journey. Her expertise and patience have been instrumental in shaping this thesis. I am grateful to my committee members Prof. Barış Akgün, and Prof. Murat Karakuş for their support and participation. This research was partially funded by Koç University - Tüpraş Energy Research Center (KUTEM) and TUBITAK Award 121C338. We thank the Tüpraş team for their help and useful feedback.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

FD	Fault Detection
TEP	Tennessee Eastman Process
SSL	Self-Supervised Learning
PCA	Principal Component Analysis
ConvAE	Convolutional Autoencoder
CPC	Contrastive Predictive Coding
TF-C	Time-Frequency Consistency
TPR	True Positive Rate
FPR	False Positive Rate
CDR	Correct Diagnosis Rate
ADD	Average Detection Delay
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit
MLP	Multilayer Perceptron
GAN	Generative Adversarial Network
IIoT	Industrial Internet of Things
DNN	Deep Neural Network
ViT	Vision Transformer
FTC	Fault-Tolerant Control
PR	Propagation Rate
IPT	Instructions per Second
ICS	Industrial Control System
ICTL	In-Cross Domain Hybrid Transfer Learning
PHVT	Pyramid Hybrid Vision Transformer
PCTF	Prediction Contrast Time Frequency
SWAT	Secure Water Treatment
WADI	Water Distribution Testbed for Research and Training

Chapter 1

INTRODUCTION

This chapter introduces Fault Detection (FD) in industrial systems, which is critical for ensuring operational safety, maintaining production efficiency, and minimizing costly system downtime. Industrial processes operate in dynamic and complex environments, where faults — deviations from normal operating conditions — can lead to equipment failures, production delays, and even safety hazards. However, detecting these faults in a timely and accurate manner is a challenging task due to the non-linear, high-dimensional, and time-dependent nature of industrial process data. Traditional FD methods, such as statistical models and rule-based approaches, struggle to handle the complexity of modern industrial systems. Machine learning and deep learning methods have emerged as alternatives, but they rely heavily on labeled datasets, which are often scarce and expensive to obtain in industrial contexts. Moreover, the need for extensive feature engineering and sensitivity to noisy, real-world data further complicate the development of effective FD systems. Addressing these issues requires more advanced, data-efficient, and robust approaches to handle the variability and uncertainty inherent in industrial environments. This chapter outlines these core challenges and establishes the context for the research presented in this thesis.

1.1 Motivation and Research Problems

Faults in the process industry encompass any abnormal conditions or deviations from expected operating parameters that disrupt normal operations, leading to inefficiency, suboptimal performance, or even total system breakdown. In this context, faults can be broadly categorized based on their source: sensor faults, actuator faults,

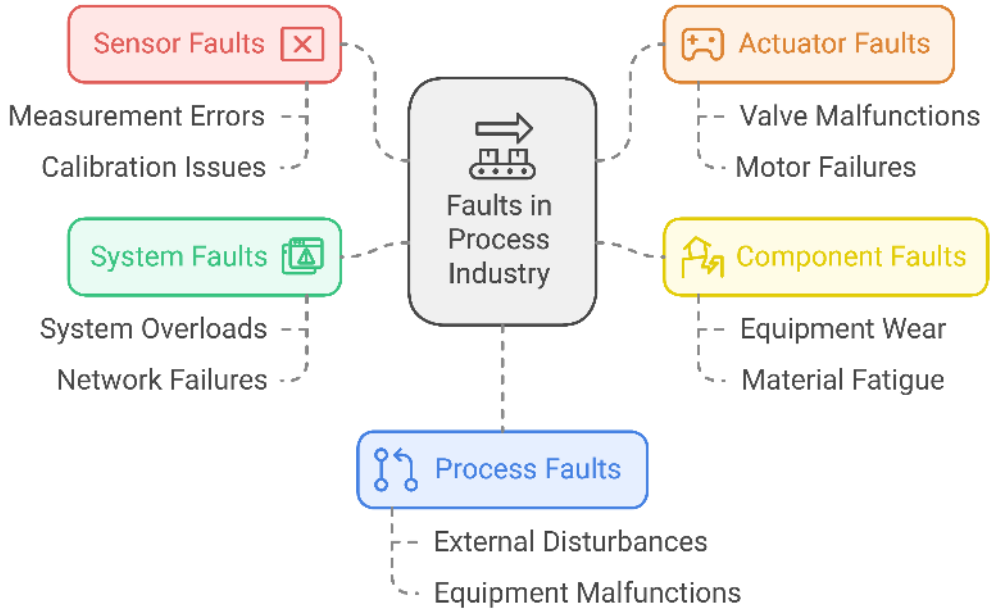

Figure 1.1: Classification of fault types in the process industry, showing different sources and examples for sensor faults, actuator faults, system faults, component faults, and process faults.

component faults, system faults, and process faults. Sensor faults arise from inaccurate measurements, actuator faults from issues in control mechanisms like valves or motors, and component or system faults from breakdowns in individual equipment or interrelated systems [Ferdowsi et al., 2022]. A visual representation of these fault types and examples is provided in Figure 1.1.

This thesis specifically focuses on process faults, which involve deviations in critical process variables such as temperature, pressure, or flow rate—from their target values. These deviations can be caused by external disturbances or internal equipment malfunctions and have a direct impact on the overall process performance. Promptly identifying and diagnosing process faults is essential for ensuring production efficiency, safety, and reliability in industrial operations.

Faults in industrial processes vary based on their nature and behavior. Abrupt faults are sudden changes that cause an immediate and often significant deviation in process variables. Incipient faults, on the other hand, develop gradually, starting as small deviations that grow over time and potentially escaping detection until they

reach a critical level. Intermittent faults are sporadic in occurrence, appearing and disappearing unpredictably, which makes detection more challenging. Faults are further categorized by their impact on process dynamics: additive faults represent external deviations added to process variables, affecting measurements and outputs, while multiplicative faults alter the internal parameters of a system, impacting its fundamental operations [Park et al., 2020].

In process monitoring, it is crucial to differentiate between faults, failures, and malfunctions. A *fault* is any deviation from expected operation that surpasses a defined threshold but does not necessarily halt the system. A fault may escalate into a *failure* if it disrupts the system's ability to function entirely, marking a permanent interruption. Failures can be further classified by their predictability, such as random (unpredictable) failures, deterministic (condition-based) failures, or systematic failures caused by specific, identifiable factors. Lastly, a *malfunction* describes a temporary or sporadic issue that impairs performance without leading to complete system stoppage [Park et al., 2020].

Industrial facilities generate substantial amounts of data, much of which remains underutilized for real-time monitoring and predictive analytics. Effective fault detection (FD) are critical to identifying faults early, preventing their escalation into failures, and maintaining optimal production and safety standards. Accurate FD also mitigates risks associated with costly downtime, equipment damage, and potential safety hazards. However, the complexity of industrial processes presents challenges in differentiating normal variability from genuine fault conditions, necessitating sophisticated and responsive FD systems.

Traditionally, statistical methods have served as the foundation for FD efforts, providing a structured framework for understanding system behaviors. However, their application has been limited by the inherent complexities and dynamic nature of industrial networks, as well as their inability to handle non-linear data interactions effectively. Over time, machine learning emerged as a preferred strategy, offering a more flexible framework for identifying patterns and anomalies in data. A critical aspect of machine learning is feature engineering, which involves converting unstructured data into a format that can be analyzed [Daurenbayeva et al., 2023]

The advent of deep learning represents a significant evolution in the field of FD. Unlike its predecessors, deep learning can process raw data without extensive feature engineering, enabling the identification of complex, non-linear patterns directly from the data. This capability provides insights into system behaviors that were previously inaccessible. However, deep learning approaches often rely on labeled data, which can be scarce, time-consuming, and costly to obtain in industrial networks [Zhu et al., 2023].

To address the challenge of limited labeled data, self-supervised learning (SSL) has emerged as a promising solution. SSL leverages the data itself to generate labels, facilitating model training in the absence of extensively labeled datasets. It encompasses several approaches, including generative-based, contrastive-based, and adversarial-based methods, each with different techniques for learning useful representations from unlabeled data [Albelwi, 2022].

In the field of FD, most studies focus on augmentation contrast, which applies various transformations to the input data to create augmented views. This method leverages the diversity in the augmented data to improve the robustness and generalization of the FD models. However, the effects of these data augmentations on the learned representations lack interpretability, hampering the understanding of the model's decisions and the underlying fault patterns. Alternatively, prediction contrast preserves the temporal connection between samples, capturing the sequential dependencies and temporal dynamics essential for accurate FD in time series data, which augmentation contrast fails to address im time series tasks [Rombach et al., 2021].

With the rise of edge computing, FD can now be performed closer to the source of data. Edge computing allows for real-time processing and analysis of data on-site, reducing latency and enabling faster response times to detected faults. This approach is particularly beneficial in industrial environments where rapid decision-making is critical. By deploying FD algorithms on edge devices, facilities can enhance their monitoring capabilities and minimize the need for transmitting large volumes of data to central servers. This not only reduces bandwidth usage but also enhances data security and privacy, as sensitive information remains within the local network.

Incorporating edge computing into FD systems represents a significant step forward in optimizing industrial processes and ensuring operational continuity [Yu et al., 2023].

In light of the challenges and evolving approaches in FD for industrial processes, this thesis aims to address several critical research problems:

1. The application of deep learning approaches to FD in industrial processes without extensive feature engineering remains a significant challenge. This research seeks to explore novel deep learning architectures and methodologies that can effectively process raw industrial data for FD.

2. The scarcity of labeled data in industrial settings poses a major obstacle to training robust FD models. This study investigates strategies to overcome this limitation, with a particular focus on leveraging unlabeled data effectively.

3. The trade-offs between augmentation contrast and prediction contrast in contrastive learning for FD, particularly in terms of model interpretability and performance, are not well understood. This study seeks to analyze these trade-offs and their implications for industrial FD systems.

4. The integration of edge computing into FD systems for enhanced real-time monitoring and decision-making in industrial environments presents both opportunities and challenges. This research aims to develop frameworks for effectively implementing edge-based FD solutions.

By addressing these research problems, this thesis contributes to the advancement of FD techniques in industrial processes. The primary objectives are to develop more efficient and accurate FD methods that can operate with limited labeled data, leverage the power of deep learning and self-supervised approaches, and take advantage of edge computing capabilities. These advancements aim to enhance the safety, reliability, and efficiency of industrial operations in the face of increasingly complex and data-rich environments.

1.2 Contributions

The contributions are provided in this section. The visual illustration of contributions can be found in Fig. 1.2.

1. **Novel Integration of Prediction Contrast Learning with Time-Frequency Information:** To the best of our knowledge, this thesis is the first to combine prediction contrast learning with both time and frequency domain information for fault detection within industrial processes. While the use of time and frequency information has been explored in augmentation contrast approaches, our work innovatively applies this dual-domain perspective within the prediction contrast framework. This novel integration allows for a more effective capture of temporal dependencies and frequency patterns, leading to a more robust fault detection in complex industrial processes.

2. **Advanced Self-Supervised Architecture for Industrial FD:** We propose a sophisticated architecture that leverages **prediction contrast learning** in conjunction with **time** and **frequency** domain information. Unlike previous approaches that rely on augmentation contrast, our method uses the predictive power of the model to learn representations, which is particularly suited to the sequential nature of industrial process data. This approach addresses the critical challenge of limited labeled data in industrial settings by learning robust representations without requiring extensive manual labeling, thus overcoming a significant hurdle in practical FD implementation.

3. **Comprehensive Evaluation on Benchmark Datasets:** We conduct an extensive evaluation of our proposed architecture and several benchmark models on two widely-used datasets for fault detection in industrial processes: the TEP Rieth and TEP Ricker datasets. We also test the models with SWaT and WADI datasets which contain security attacks as anomalies to evaluate the performance of the models in other types of anomalies. This comprehensive comparison includes traditional statistical methods, conventional deep learning approaches, and state-of-the-art self-supervised learning techniques. By

using these diverse and challenging datasets, we demonstrate the broad applicability and effectiveness of our prediction contrast approach across different industrial scenarios and fault types.

4. **Significant Performance Improvements in FD Metrics:** Our experimental results demonstrate that the proposed architecture substantially outperforms both traditional methods and other state-of-the-art models in key FD metrics, including True Positive Rate (TPR), Correct Detection Rate (CDR), and Average Detection Delay (ADD). On the **TEP Rieth dataset**, our method achieves a TPR of 74.51%, surpassing the next best model (CPC) by 2.06 percentage points. Furthermore, our method achieves a CDR of 88.71%, which is 3.58 percentage points higher than the second-best model (SensorScan). For ADD, our method achieves an ADD of 25.74 steps, marking a 41.02% reduction compared to the next best model (CPC) with 43.67 steps.

   On the **TEP Ricker dataset**, our method achieves a TPR of 92.04%, representing a 1.91 percentage point improvement over the next best model (SensorScan) at 90.13%. Our method also achieves a CDR of 97.62%, which is 1.00 percentage points higher than the second-best model (PCA_KMeans). In terms of ADD, our method achieves an ADD of 22.18 steps, reflecting a 28.41% reduction in ADD compared to the second-best model (SensorScan) with 30.98 steps.

   On the **SWAT dataset**, our method achieves a TPR of 92.23%, which is 4.21 percentage points higher than the next best model (SensorScan) at 88.02%. In terms of CDR, our method achieves a CDR of 94.01%, a 5.99 percentage point improvement over the second-best model (PCA_KMeans). For ADD, our method achieves an ADD of 25.20 steps, marking a 17.38% reduction in ADD compared to the next best model (SensorScan) with 30.50 steps.

   On the **WADI dataset**, our method achieves a TPR of 86.50%, which is 5.46 percentage points higher than the second-best model (SensorScan) at

81.04%. Additionally, our method achieves a CDR of 91.52%, representing a 7.51 percentage point improvement over the next best model (PCA_KMeans) at 84.01%. For ADD, our method achieves an ADD of 35.80 steps, reflecting a 38.41% reduction in ADD compared to the second-best model (SensorScan) with 58.12 steps.

5. **Edge Computing Analysis Framework for FD Deployment:** We contribute a comprehensive analysis framework for evaluating the deployment of FD models in edge versus cloud computing scenarios. This framework extends the scope of FD research beyond model development to practical implementation considerations. By simulating various industrial scales and complexities, our analysis provides a structured approach to assess the **scalability** and **real-time performance** of FD systems. Our experimental results demonstrate that edge deployment maintains consistently low latency (approximately 248 ms) across different scales, while cloud deployment shows significantly increasing latencies in complex scenarios (up to 156,712.11 ms for 32 plants at 10^5 simulation time). These findings highlight the potential of edge computing for enabling responsive fault detection in industrial settings. This contribution offers valuable insights for optimizing FD system deployments and bridges the gap between theoretical model development and practical industrial implementation.

6. **Ablation Studies to Validate Model Components:** The ablation studies conducted in this thesis provide a novel contribution by rigorously isolating and analyzing the impact of individual components of the proposed architecture. These studies highlight the critical role of specific elements, such as the optimal number of Transformer Encoders and the choice of GRU for autoregressive modeling. By demonstrating how each part contributes to overall model performance, this thesis offers a clear methodology for refining and optimizing self-supervised learning models in industrial fault detection contexts.

7. **Open Issues and Future Directions:** This thesis identifies several open

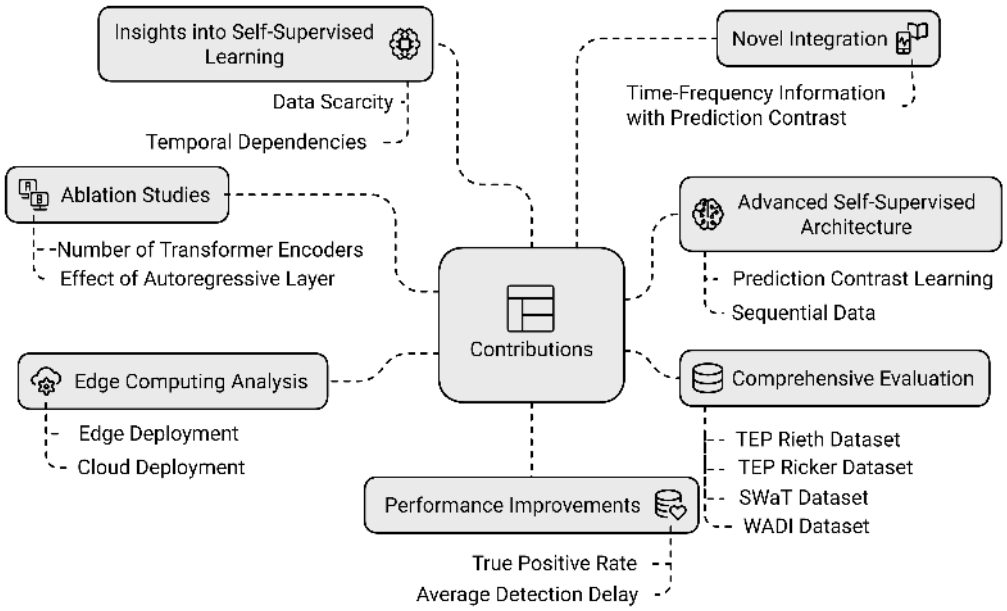

Figure 1.2: Illustration of the contributions of this thesis.

issues and future directions for advancing self-supervised learning in FD. Key challenges include optimizing the trade-off between sensitivity and specificity, improving model interpretability, and enhancing domain generalization. Addressing these issues requires developing advanced loss functions, adaptive thresholding, and explainability techniques like attention mechanisms and time-frequency visualizations. Future work should also explore multimodal data integration, enabling models to process time series, images, and text simultaneously. Additionally, optimizing models for edge deployment through techniques like pruning, quantization, and federated learning will support low-latency, real-time FD in industrial environments. These efforts aim to create more robust, interpretable, and scalable models for diverse industrial applications.

1.3 Organization

The remainder of this thesis is organized as follows: Chapter 2 introduces the fundamentals of FD, including key concepts, challenges, evaluation metrics, and a review

of related work. Chapter 3 introduces the proposed PCTF model, highlighting its design and advantages for fault detection in industrial processes. Chapter 4 provides an overview of the benchmark architectures, including PCA with K-Means, ConvAE, CPC, and TF-C. Chapter 5 describes the datasets and experimental setup used, focusing on the TEP datasets and detailing data preprocessing, model training, and evaluation metrics. Chapter 6 presents ablation studies and model analysis, discussing the impact of various components and configurations on performance. It also reports the experimental results obtained from the TEP datasets, comparing the proposed model's performance against benchmark models and analyzing cloud and edge computing scenarios. Finally, Chapter 7 discusses open issues in FD, future research directions, and concludes by summarizing the key findings and contributions of this work.

Chapter 2

FAULT DETECTION AND RELATED WORK

In this chapter, we provide the basics of FD and the various challenges confronted by the current FD applications. Therefore, we provide the related work of FD including various approaches from state of the art.

2.1 Fault Detection

2.1.1 Introduction

Process monitoring is a critical component in ensuring the long-term reliable operation of any automated controlled system. In the context of industrial processes and complex systems, it is essential to distinguish between different types of operational disruptions. Following the definitions provided by Isermann and Ballé [Isermann and Balle, 1997], we can categorize these disruptions as follows:

A **disturbance** is an unknown and uncontrolled input acting on a system, which can lead to a fault, defined as an unpermitted deviation of at least one characteristic property or parameter of the system from the acceptable/usual/standard operating conditions. If a **fault** is not corrected or managed, it can escalate into a **failure**, which is a permanent interruption of a system's ability to perform a required function under specified operating conditions. The focus of this thesis is to identify the faults in the dataset which could lead to a failure in industrial process.

Figure 2.1 illustrates the transition from normal operation to a fault condition in an industrial process, using data from the TEP dataset. The graph depicts the temporal evolution of key process variables, with the vertical dashed line indicating the fault onset. In this visualization, the fault is characterized by visually apparent deviations in multiple process variables from their normal operating ranges. These deviations, which show up as abrupt changes in the plotted lines following the fault

onset, highlight the challenge it can be to identify these kinds of transitions in intricate industrial systems where several variables may be impacted at once.

Traditional control systems are designed to return the system to normal operations in the presence of disturbances but are not equipped to handle faults or failures. This limitation has led to the development of Fault-tolerant control (FTC) systems, which are explicitly designed to account for specific classes of faults in the closed-loop system. FTC systems must operate in the critical time window between the occurrence of a fault and a potential system failure.

In industrial processes, faults can manifest as extreme events such as catalyst deactivation, valve blockage, or compressor failure. The increasing complexity of modern facilities has made faults not only inevitable but more frequent. The challenge of monitoring is further complicated by recycle streams causing bidirectional interactions and control systems that can mask the effects of faults. Moreover, multiple faults often occur simultaneously, adding another layer of complexity to the detection and diagnosis process. This problem is handled by human operators via tracking the alarm management system of the plants [Görgülü and Özkasap, 2023] [Gill et al., 2024].

Despite these challenges, even relatively simple modern facilities are equipped with large sensor networks that can be leveraged for process monitoring. The key to effective FD lies in utilizing these sensors efficiently to minimize the impact of faults. This is where advanced techniques, including deep learning, can play a crucial role.

While traditional methods based on simple thresholds or statistical deviations may suffice for straightforward scenarios, they often fall short in complex systems with non-linear relationships and subtle fault signatures. By identifying complex patterns and relationships that might not be visible through linear algorithms or human observation, deep learning techniques can excel in these situations. This capability becomes particularly valuable when dealing with multiple simultaneous faults or when the manifestation of a fault involves subtle changes in the relationships between multiple variables over time [Görgülü and Özkasap, 2024].

As we learn more about FD methodologies, it becomes evident that the complexity of the system, the type of potential faults, and the particular needs of the

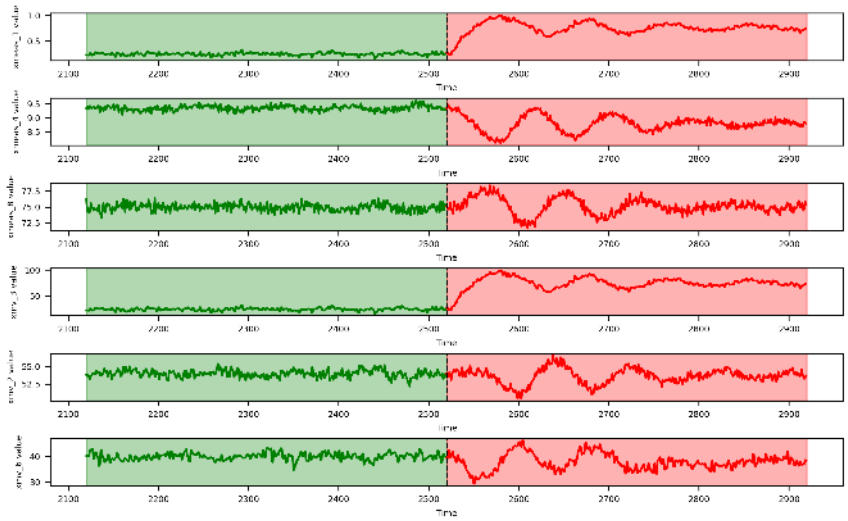

Figure 2.1: Demonstration of state switch from normal condition to faulty condition. Green lines represent the healthy state while the red lines represent the faulty state.

monitoring task at hand should all be taken into consideration when selecting an approach, whether it be straightforward statistical methods or sophisticated machine learning techniques.

2.1.2 Evaluation Metrics of FD

Evaluation metrics are critical for assessing the performance of Fault Detection (FD) systems. These metrics quantify the system's ability to correctly identify and respond to the presence of faults under varying conditions. The primary focus is on detection metrics, which evaluate the system's binary classification performance in distinguishing faulty states from normal operating conditions.

To provide a comprehensive evaluation, we utilize the following industry standard detection metrics for FD: True Positive Rate (TPR), False Positive Rate (FPR), and Average Detection Delay (ADD). A comprehensive evaluation of the system's performance is made possible by the unique perspectives that each of these metrics provides on its detection capabilities.

Detection TPR

In a binary classification task where all faulty samples are positive and all normal samples are negative instances, the Detection TPR calculates the percentage of real fault occurrences that the system properly detects. It is computed as follows:

$$\text{Detection TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

where TP states the count of true positives, and FN indicates the count of false negatives.

Detection FPR

In a binary classification task where all faulty samples are positive and all normal samples are negative examples, the Detection FPR calculates the percentage of normal instances that are falsely classified as faults. It is described as:

$$\text{Detection FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

where FP presents the count of false positives, and TN indicates the count of true negatives.

Average Detection Delay (ADD)

The average amount of time that passes between an event occurring and the system detecting it is measured by the ADD. Faster decision rates are indicated by lower ADD levels. The ADD is computed as follows:

$$\text{ADD} = \frac{1}{N}\sum_{i=1}^{N}(\text{DT}_i - \text{OT}_i) \times \text{s}$$

where DT_i represents the detection time of the i-th fault, OT_i denotes the actual occurrence time of the i-th fault, N is the total number of detected faults, and s corresponds to the step size hyperparameter, which determines the discrete time intervals for data sampling in the training set.

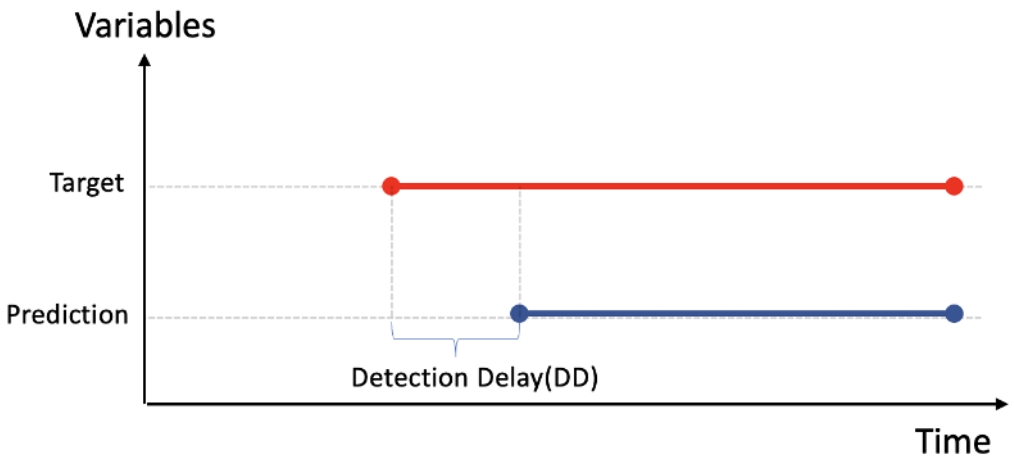

Figure 2.2: Demonstration of detection delay which is the time delay between state alterations and detection of this alteration by the model.

ADD reflects the time gap between a process state change and its recognition by the fault detection system, as shown in Fig. 2.2.

The benchmark datasets contain a unique identifier, referred to as a run id, which identifies a single run. The ADD calculates the average time lag between the true occurrence of a fault (from labels) and its earliest valid detection, considering only predictions that occur on or after the actual fault. First, the earliest true fault time (real change point) and the earliest predicted fault time for each run (for each run id) are identified. Predictions occurring before the true fault are filtered out to ensure accuracy. The detection delay for each valid prediction is then computed as the time difference between the predicted and true fault occurrences, scaled by the predefined step size. This approach ensures that the ADD calculation remains robust and consistent, as it does not directly depend on the window size; whether short or long windows are used, the ADD calculation remains unaffected, and no normalization is required.

Correct Diagnosis Rate

The CDR measures the proportion of critical fault cases that the model correctly identifies. It is determined using the following formula:

$$\text{CDR} = \frac{\sum_{i=1}^{K} \text{TP}_i}{\sum_{i=1}^{K} (\text{TP}_i + \text{FP}_i)}$$

where TP_i denotes the number of true positives for the i-th fault class, FP_i represents the number of false positives for the i-th fault class, and K corresponds to the total number of fault classes. This metric reflects the model's ability to accurately classify and distinguish between different fault categories.

2.1.3 Issues with FD

FD in industrial processes face several significant challenges that hinder their effectiveness and reliability. Understanding these issues is crucial for developing robust FD systems. The primary issues are as follows:

Data Quality and Preprocessing

Noise, missing values, and incorrect timestamps resulting from sensor failures and transmission delays frequently degrade the quality of data gathered from industrial processes. The issue with the detection method is made more difficult by this dirty data, leading to the need for intensive preprocessing to guarantee data reliability.

Generalizability and Context Change

FD models must generalize well to various operating conditions and equipment configurations. However, industrial processes can change over time due to maintenance, upgrades, or shifts in operational strategies. These context changes can degrade the performance of FD systems if they are not adequately accounted for during model training and deployment.

Insufficient Labeled Data

Accurately labeling faults demands subject expertise, which makes obtaining labeled data for FD model training expensive and time-consuming. Exploring self-supervised and unsupervised learning strategies to take use of unlabeled data is necessary because the effectiveness of supervised learning approaches is limited by the lack of labeled data.

Real-Time Processing and Latency

Timely fault detection is critical in industrial settings to prevent severe consequences. Traditional centralized processing approaches can introduce significant latency due to data transmission and processing delays. Edge computing offers a promising solution by enabling real-time data processing close to the data source, reducing latency and improving response times.

Frequency Differences Between Data Sources

Industrial processes often involve multiple data sources, such as alarm data and process data, which can have different sampling frequencies. The accuracy of FD may be impacted by these frequency discrepancies, which can make data integration and analysis more difficult.

Interpretability of Models

Understanding the decisions made by FD models is crucial for gaining trust and facilitating corrective actions. However, complex models, particularly deep learning-based approaches, often lack interpretability, making it challenging to understand the reasoning behind their predictions and the underlying fault patterns.

Scalability and Computational Resources

Large volumes of data are produced by industrial processes, needing scalable FD systems that can effectively handle large data sets. Sophisticated models can require

significant computational resources for training and deployment, which calls for algorithm enhancement and the effective use of distributed computing resources.

In this thesis, we address the issue of extensive preprocessing with a deep learning architecture, improve generalizability by using both frequency and time information from the data, tackle the problem of insufficient labeled data through self-supervised learning, increase interpretability of the deep learning model by a novel architecture, and analyze real-time processing and latency through edge computing analysis. The readers will be informed in the next Section 2.2 with various approaches as solution for the mentioned issues.

2.2 Related Work

Research on fault detection in industrial processes has been in progress, with researchers examining a range of approaches to overcome the difficulties presented by complex systems and data restrictions. Machine learning and deep learning techniques have emerged as promising solutions as data-driven and technology-focused approaches have gained prominence [Neupane and Seok, 2020]. In order to improve FD performance, recent research has placed a greater emphasis on data fusion and complex deep learning architectures. These solutions include supervised and self-supervised learning techniques as well as conventional ones.

Public datasets used in FD research vary in characteristics such as the frequency of logs and the type of data (process or alarm data). For instance, while the log frequency in the Pronto dataset [Stief et al., 2019] varies from feature to feature, the frequency in the TEP dataset [Chen, 2019] is fixed. The Pronto dataset includes both alarm and process data, while the TEP dataset only includes process data. To address these differences, researchers have implemented various **data fusion** techniques to utilize both alarm and process data effectively.

2.2.1 Data Fusion Techniques

Several studies have explored decision-level fusion, integrating alarm and process data to detect faults [Stief et al., 2018]. Unlike feature-level fusion, decision-level

fusion synthesizes the outcomes of individual classifications to render a final decision. Other approaches, such as polygon generation [Name, 2022, Elhefnawy et al., 2023], have proposed novel methods for representing complex feature relationships and fusing polygons with raw data, creating composite representations.

2.2.2 Handling Heterogeneous Data

To handle heterogeneous process variables, techniques like the Continuous Wavelet Transform (CWT) have been employed, as they are better suited for non-stationary data compared to the Fourier Transform. Traditional techniques like Wavelet Transform and Principal Component Analysis (PCA), as well as deep learning architectures such as Autoencoders and Recurrent Neural Networks (RNNs), have also been explored, although they have limitations such as accuracy issues and the gradient vanishing problem with long sequences [Wang et al., 2023].

2.2.3 Edge Computing Applications

Optimizing machine learning models for edge computing scenarios is crucial to reduce computational costs, storage requirements, and latency. Common techniques include feature selection to reduce input dimensionality, and lightweight models to minimize computational overhead. [Marino et al., 2021] adopted the Hybrid Fisher Wrapper (HFW) method, a multi-step feature selection approach that involves: calculating Fisher scores, ranking features, creating subsets of top-ranked features, training models, and selecting the best subset. This method reduced latency by 37.5 times and the number of features by 1.99 times on average for edge computing with near-sensor processing.

In contrast to optimizing the features, [Wang et al., 2020] proposed a lightweight convolutional neural network (LCNN) tailored for intelligent fault diagnosis in the Industrial Internet of Things (IIoT) context. The LCNN incorporates depthwise separable convolutions, inverse residual connections, and linear bottleneck layers to reduce model parameters and computational costs while maintaining high accuracy. A novel decomposed Hierarchical Search Space automatically optimizes the LCNN

architecture for the given fault diagnosis task.

2.2.4 Supervised Learning Approaches

Expanding the scope of data sources, researchers have incorporated video analytics into FD by leveraging Vision Transformers (ViTs) for video feature extraction and specialized RNN architectures for processing numerical process data, embodying a truly multimodal diagnostic strategy [Tian, 2023b]. Furthermore, Transformer Encoders have been proposed as feature extractors, complemented by multilayer perceptron (MLP) classifiers, to overcome the shortcomings of earlier models [Guo et al., 2023].

Novel architectures, such as the Neuron-Compressed Deep Neural Network (NCDNN) [Wang et al., 2023] and the temporal CNN1D2D [Lomov et al., 2021], have been introduced to improve fault detection performance in industrial settings. The NCDNN employs a special regularization technique in the context of an encoding-decoding structured DNN, while the temporal CNN1D2D incorporates a Generative Adversarial Network (GAN) with an LSTM framework as the generator and a novel CNN1D2D architecture as the discriminator.

In scenarios with limited data availability like in few-shot situations, researchers have proposed frameworks like the Deep Feature Transfer Fusion (DFTF) [Tian, 2023a], which incorporates the Pyramid Hybrid Vision Transformer (PHVT) model and the In-Cross Domain Hybrid Transfer Learning (ICTL) method to address the challenge of FD in industrial processes with scarce data.

To better represent industrial complexities inside the dataset, researchers have proposed new datasets for the TEP by generating data through continuous simulations, randomly selecting faults and durations [e Souza, 2023]. The simulated data are then transformed into 2D matrices, normalized, and processed using a sliding window technique for the training and testing of CNNs for FD.

2.2.5 Self-Supervised Learning Approaches

Researchers have looked into SSL, an approach that uses unlabeled data to get over the limitations of data labeling costs and availability, in order to move beyond the dependency on labeled data in supervised learning. SSL is a type of unsupervised learning that utilizes pretext tasks to extract meaningful representations from unlabeled data [Zhang et al., 2024a].

In this context, researchers have presented novel applications of techniques like pyramid reconstruction (PR) [Tian et al., 2023], which is derived from the image processing domain using the Feature Pyramid Network (FPN). The pyramid reconstruction loss is employed to train the autoencoder network, which then generates latent space embeddings that are processed by the Gaussian Mixture Model.

Moreover, SensorScan [Golyadkin et al., 2023] proposes pretraining architectures with Transformer encoders, reconstruction loss, and contrastive losses inspired by techniques such as SimCLR [Chen et al., 2020] and SCAN [Gansbeke et al., 2020]. This approach aim to leverage efficient representation learning and clustering techniques to achieve performance comparable to or better than recent supervised methods. Specifically, SensorScan employs augmentation contrast techniques by applying various transformations to the input data to create augmented views during the pretraining stage. This enables the model to learn robust and invariant representations, improving generalization and fault detection performance. However, the effect of these data augmentations on the learned representations is often not explainable, which can be a limitation in interpreting the model's decisions and understanding the underlying fault patterns.

Paper	**Dataset**	**Data Type**	**Method**	**Metrics**	**Architecture**	**Research Focus**
[Stief et al., 2018]	ABB datasets	Alarm + Process	SL	Acc, Prec, Rec	Naive Bayes	Data Fusion Techniques
[Name, 2022]	TE	Process (Heterogeneous)	SL	Acc, FAR	CNNs	Data Fusion Techniques
[Elhefnawy et al., 2023]	Pronto	Process (Heterogeneous)	SL	Prec, Rec, F-1	CNNs	Data Fusion Techniques
[Wang et al., 2023]	Pronto	Process	SL	FDR	Deep Neural Network	Handling Heterogeneous Data
[Tian, 2023b]	Pronto	Video + Process	SL	Acc, Prec, Rec, F-1	ViT + RNN	SL Approaches
[Guo et al., 2023]	Pronto, TE	Process	SL	Acc	Transformer + MLP	SL Approaches
[Lomov et al., 2021]	TE	Process	SL	FPR, TPR	CNN1D + CNN2D	SL Approaches
[Tian, 2023a]	Pronto	Video	SL	Acc, Prec, Rec, F-1	DFTF	SL Approaches
[Tian et al., 2023]	Pronto, TE	Process	SSL	Prec, Rec, F-1	PRDAGMM	SSL Approaches
[Golyadkin et al., 2023]	TE	Process	SSL	TPR, FPR, CDR, ADD	Transformer + MLP	SSL Approaches

Table 2.1: Studies focused on FD categorized by dataset, data type, learning method (SL/SSL), metrics, architecture, and subsection.

Chapter 3

PROPOSED SYSTEM ARCHITECTURE AND METHODOLOGY

This chapter describes the proposed solution, which we call Prediction Contrast Time Frequency, designed to improve fault detection. The system combines time and frequency based data to help capture important patterns in industrial data within the prediction contrast learning setup. Key components, like transformer encoders and contrastive learning are introduced with a focus on how this system is designed to be both effective and efficient.

3.1 Overview of the Prediction Contrast Time Frequency

The Prediction Contrast Time Frequency (PCTF) model represents a significant advancement in fault detection for time-series data, building upon the foundation of existing models like TF-C by incorporating predictive coding principles. This approach emphasizes the integration of both time and frequency domain information to capture a comprehensive representation of the data. The architecture is designed to leverage modern techniques, such as Transformer encoders, to improve the detection of faults and complex patterns within industrial datasets.

At its core, the PCTF model consists of four key components that work together to enhance its representational and predictive capabilities. First, the time-based Transformer encoder processes temporal dependencies within the data, capturing long-range patterns critical for accurate predictions. Second, the frequency-based Transformer encoder operates in the frequency domain, extracting complementary insights that may not be evident in the time domain alone. The autoregressive layer, the third component, processes past embeddings to generate a context vector that encapsulates the essential historical information necessary for future predictions.

Finally, the projection head uses this context vector to predict future embeddings, facilitating contrastive learning.

The proposed architecture's workflow involves encoding data independently in both time and frequency domains. The encoded embeddings are then split into past and future components, enabling the model to focus on predicting the future while grounding its understanding in the past. Contrastive learning, implemented through the InfoNCE loss, ensures that the predicted embeddings align closely with the actual future embeddings, fostering the learning of meaningful representations. By combining these elements, PCTF delivers a robust framework for fault detection, designed to optimize performance and maintain computational efficiency.

Figure 3.1 provides a visual representation of the proposed architecture. Additionally, Algorithm 1 outlines the pretraining process, which focuses on learning generalizable representations from unlabeled data using self-supervised learning methods. The clustering process, described in Algorithm 2, adopts a scan-based approach inspired by the clustering and label matching logic presented in [Golyadkin et al., 2023]. This method leverages Semantic Clustering by Adopting Nearest Neighbors (SCAN) loss to group embeddings, combining alignment and entropy losses to optimize cluster assignments while ensuring robust and balanced distributions. The final label matching step assigns clusters to process states by maximizing the weighted occurrence of labels within clusters, a technique designed to minimize false positives and ensure accurate diagnostics.

3.2 Architecture Details

Transformer Encoders

Unlike the TF-C model, which uses 1D ResNets, we employ Transformer encoders for both time and frequency domains. The Transformer architecture allows for more effective capturing of long-range dependencies in the data. For an input sequence x, the encoder output is:

$$z = \text{TransformerEncoder}(x) \tag{3.1}$$

The same process is applied to both time-domain input x_t and frequency-domain input x_f.

Embedding Split and Autoregressive Processing

After encoding, the resulting embeddings are split into two parts:

$$z = [z_{\text{past}}, z_{\text{future}}] \tag{3.2}$$

where z_{past} represents the embeddings for the time before t, and z_{future} represents the embeddings for the time after t. The history window (z_{past}) is passed through an autoregressive layer to generate a context vector:

$$c = \text{AutoregressiveLayer}(z_{\text{past}}) \tag{3.3}$$

This context vector is then used to produce future predictions via a projection head:

$$\hat{z}_{\text{future}} = \text{ProjectionHead}(c) \tag{3.4}$$

After that, the prediction contrast loss is ready to be calculated.

InfoNCE Loss

We employ the InfoNCE loss for contrastive learning. The loss is calculated between the predicted future embeddings and the actual future embeddings:

$$\mathcal{L}\text{InfoNCE} = -\log \frac{\exp(\text{sim}(\hat{z}_{\text{future}}, z_{\text{future}})/\tau)}{\sum_k \exp(\text{sim}(\hat{z}_{\text{future}}, z_k)/\tau)} \tag{3.5}$$

where z_k includes both positive and negative samples, and τ is a temperature parameter. This loss is computed for both time and frequency domains.

Total Loss

The total loss is computed by summing the InfoNCE losses from both the time and frequency domains:

$$\mathcal{L}\text{total} = \mathcal{L}_{\text{InfoNCE}}^{\text{time}} + \mathcal{L}_{\text{InfoNCE}}^{\text{freq}} \tag{3.6}$$

Removal of Explicit Consistency Loss

Unlike the TF-C model, we do not include an explicit consistency loss between time and frequency domain representations. Through extensive experimentation, we found that the InfoNCE loss inherently maintains consistency between the two domains, making an additional consistency loss redundant.

Pre-training and Fine-tuning Process

The model is pretrained on a large unlabeled dataset using the combined InfoNCE loss to learn generalizable representations. For downstream tasks, we used the clustering approach that is used in [Golyadkin et al., 2023].

3.3 Advantages over Previous Approaches

The Prediction Contrast Time Frequency (PCTF) model introduces several key improvements over previous approaches. By employing Transformer encoders, it effectively captures long-range dependencies in both time and frequency domains, enabling a deeper understanding of complex patterns. The predictive contrast framework further enhances the model by focusing on future prediction, allowing it to learn more meaningful and generalizable representations.

Unlike traditional models, PCTF eliminates the need for explicit consistency loss between time and frequency domain embeddings. The InfoNCE loss inherently aligns the representations across domains, simplifying the architecture and reducing computational overhead. Additionally, by leveraging both time and frequency information, PCTF provides a comprehensive view of the data, leading to more robust fault detection.

Lastly, the use of batch-wise contrastive learning improves computational efficiency compared to previous methods that rely on entire pretraining datasets for contrastive sampling. These advancements make PCTF a streamlined and effective solution for fault detection in industrial time-series data.

Algorithm 1 Pretraining algorithm workflow of PCTF pipeline

Require: Time series data $X = \{x_1, x_2, ..., x_T\}$, window size w, batch size B

Ensure: Pre-trained model loaded if the weights are given

Initialize E_t, E_f, A, and P ▷ Time Encoder, Frequency Encoder, Autoregressive Layer, Projection Head

Set learning rate α, temperature τ, number of epochs E

procedure PRETRAIN(X)

  for epoch $\leftarrow 1$ to E **do**

    for each batch X_B in epoch data **do**

      $X_f \leftarrow \text{FFT}(X_B)$ ▷ Fourier Transform of time series data

      $Z_t \leftarrow E_t(X_B)$, $Z_f \leftarrow E_f(X_f)$ ▷ Encode both time and frequency domains

      $Z_t \leftarrow [Z_{t,\text{past}}, Z_{t,\text{future}}]$, $Z_f \leftarrow [Z_{f,\text{past}}, Z_{f,\text{future}}]$ ▷ Split past and future

      $C_t \leftarrow A(Z_{t,\text{past}})$, $C_f \leftarrow A(Z_{f,\text{past}})$ ▷ Generate context vectors from past

      $\hat{Z}_{t,\text{future}} \leftarrow P(C_t)$, $\hat{Z}_{f,\text{future}} \leftarrow P(C_f)$ ▷ Predict future in both domains

      $L_t \leftarrow \text{InfoNCE}(\hat{Z}_{t,\text{future}}, Z_{t,\text{future}}, \tau)$

      $L_f \leftarrow \text{InfoNCE}(\hat{Z}_{f,\text{future}}, Z_{f,\text{future}}, \tau)$

      $L_{\text{total}} \leftarrow L_t + L_f$ ▷ Compute total contrastive loss

      Update model parameters using L_{total} and learning rate α

    end for

  end for

end procedure

Algorithm 2 Deep Clustering for Fault Detection in PCTF

Require: Unlabeled data $X_{\text{unlabeled}}$, pretrained feature extractor F presented in Algorithm 1, clustering head C

Ensure: Pretrained feature extractor F for embedding extraction, clustering head C initialized with random weights, fault detection model refined for assigning clusters to data points

procedure DEEPCLUSTERING($X_{\text{unlabeled}}$)
    Load pre-trained weights for feature extractor F
    Initialize clustering head C with random weights
    for epoch $\leftarrow 1$ to E_{cluster} **do**
        for each batch X_B in $X_{\text{unlabeled}}$ **do**
            Extract embeddings $Z \leftarrow F(X_B)$
            Assign cluster probabilities $\hat{y} \leftarrow C(Z)$
            Identify nearest neighbors for each sample in X_B ▷ Compute neighborhood relations
            Compute alignment and entropy losses ▷ Alignment ensures similar data close, entropy ensures balanced clusters
            Compute SCAN loss L_{scan} ▷ Combine alignment and entropy losses
            Update model parameters of F and C using L_{scan} and learning rate α_{cluster}
        end for
    end for
end procedure

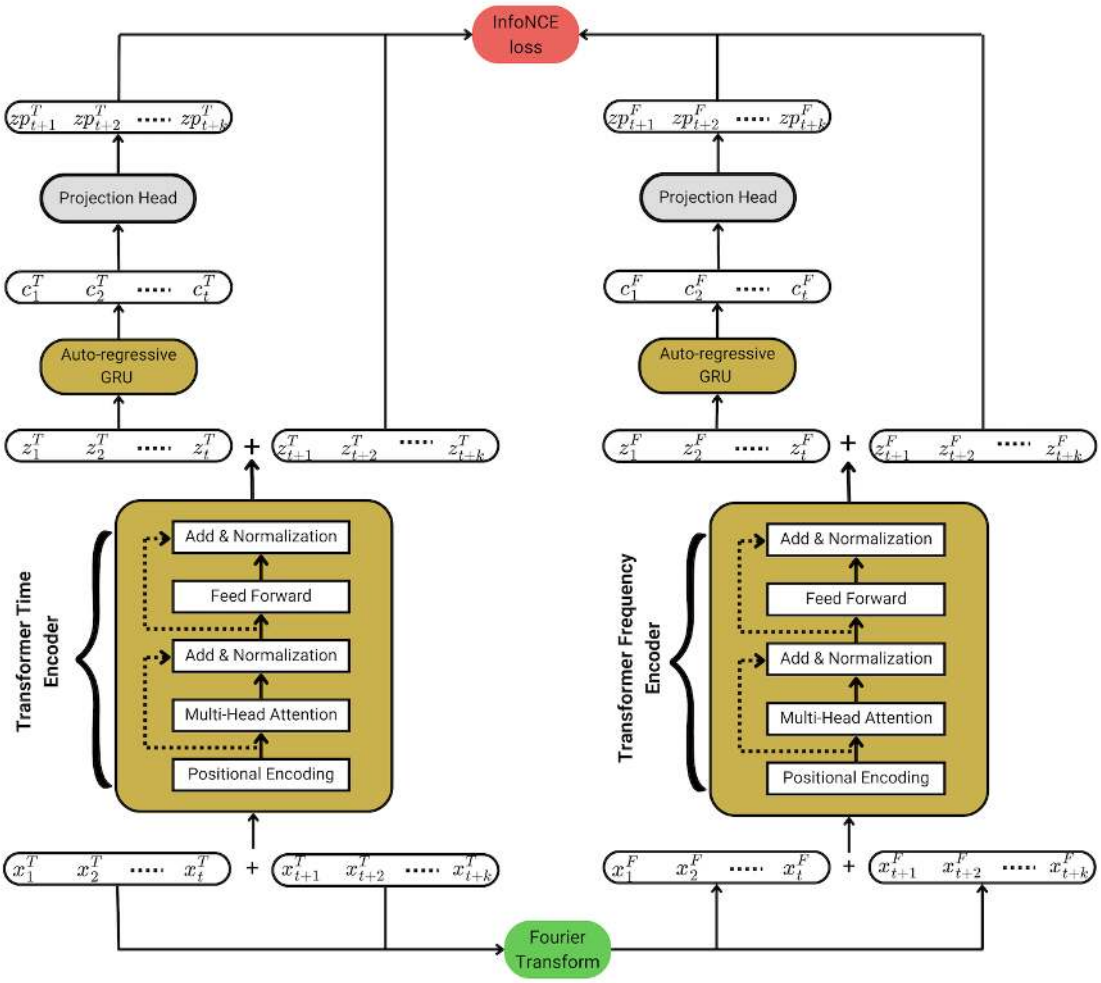

Figure 3.1: The proposed self-supervised pretraining architecture Prediction Contrast Time Frequency.

Chapter 4

OVERVIEW OF THE BENCHMARK ARCHITECTURES

This chapter provides a comprehensive overview of the benchmark architectures employed in this thesis. We begin by detailing the selected architectures, including Principal Component Analysis (PCA) with K-Means Clustering, Convolutional Autoencoder (ConvAE), SensorSCAN, and Contrastive Predictive Coding (CPC), as well as the Time-Frequency Consistency (TF-C) model. These architectures include both established methods in the field and adaptations specifically tailored for FD tasks.

4.1 PCA with K-Means Clustering

PCA with K-Means Clustering represents a fundamental yet effective approach to fault detection, as proposed by [He et al., 2005]. This method combines dimensionality reduction with unsupervised clustering to identify anomalies in multivariate data.

4.1.1 PCA for Dimensionality Reduction

PCA is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. This transformation is defined in such a way that the first principal component has the largest possible variance, and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components.

Mathematically, given a data matrix X of size $n \times p$, where n is the number of samples and p is the number of features, PCA finds a linear transformation matrix W of size $p \times k$ ($k \leq p$) such that:

$$Y = XW$$

where Y is the transformed data of size $n \times k$. The columns of W are the eigenvectors of the covariance matrix of X, ordered by their corresponding eigenvalues in descending order.

4.1.2 K-Means Clustering

After reducing the dimensionality of the data using PCA, K-Means clustering is applied to partition the data into K clusters. The algorithm aims to minimize the within-cluster sum of squares (WCSS):

$$\text{WCSS} = \sum_{i=1}^{K} \sum_{x \in C_i} \|x - \mu_i\|^2$$

where C_i is the i-th cluster and μ_i is the centroid of cluster C_i.

4.1.3 Fault Detection Process

The fault detection process using PCA with k-means clustering begins by applying PCA to reduce the input data's dimensionality. The optimal number of principal components is selected based on the largest TPR observed on the training set. The data is then projected onto these selected components, and k-means clustering is applied to the projected data. During detection, each new data point's distance to the nearest cluster center is calculated. Points exceeding a predefined distance threshold are classified as anomalies, potentially indicating faults. This method is particularly effective for detecting deviations from normal operating conditions in the reduced dimensionality space, allowing for the identification of anomalies that might be obscured in the original high-dimensional data.

4.2 ConvAE

The ConvAE architecture, as described in [Dong et al., 2018], is a powerful unsupervised learning approach for fault detection in time series data. It leverages the

strengths of convolutional neural networks and the autoencoder framework to learn compact representations of normal operating conditions and detect anomalies based on reconstruction error.

4.2.1 Architecture Overview

The ConvAE architecture consists of two main components: an encoder and a decoder. The encoder comprises a series of convolutional layers that progressively reduce the spatial dimensions of the input while increasing the number of feature maps, effectively compressing the input data into a compact latent representation. Complementing this, the decoder employs a series of transposed convolutional layers, also known as deconvolutional layers, which work to reconstruct the original input from this compact latent representation. This encoder-decoder structure allows the ConvAE to learn a compressed representation of the input data and then attempt to recreate it, with the reconstruction error serving as a key indicator for FD task.

4.2.2 Encoder

The encoder consists of multiple convolutional layers, each followed by a non-linear ReLU activation function and optionally a pooling layer. For a 1D time series input x of length L, the i-th convolutional layer can be described as:

$$h_i = f(\text{Conv1D}(h_{i-1}, W_i) + b_i) \tag{4.1}$$

where h_i is the output of the i-th layer, W_i and b_i are the weights and biases of the layer, and f is the activation function.

4.2.3 Decoder

The decoder mirrors the encoder structure but uses transposed convolutions to upsample the latent representation back to the original input dimensions:

$$\hat{h}_i = f(\text{ConvTranspose1D}(\hat{h}_{i+1}, \hat{W}_i) + \hat{b}_i) \tag{4.2}$$

where $\hat{h}_i$ is the output of the i-th layer in the decoder, and $\hat{W}_i$ and $\hat{b}_i$ are the weights and biases of the transposed convolutional layer.

4.2.4 Training and Fault Detection

The ConvAE is trained to minimize the reconstruction error between the input and the output:

$$\mathcal{L} = \|x - \hat{x}\|^2 \tag{4.3}$$

where x is the input and $\hat{x}$ is the reconstructed output.

For FD, the ConvAE is trained on normal operating data. When presented with new data, the reconstruction error is used as an anomaly score. If the reconstruction error exceeds a predefined threshold, the input is classified as anomalous, potentially indicating a fault.

4.3 SensorSCAN (Self-Supervised Learning and Deep Clustering)

SensorSCAN, as introduced by [Golyadkin et al., 2023], is a novel approach to self-supervised fault detection and diagnosis in industrial chemical processes. Unlike traditional supervised methods that require labeled data, SensorSCAN leverages SSL and deep clustering techniques to detect and diagnose faults without requiring expert annotations.

4.3.1 Architecture Overview

The SensorSCAN architecture consists of a feature extractor and a clustering head. The feature extractor is a Transformer-based model that processes raw sensor data and generates latent embeddings. The clustering head assigns these embeddings to clusters, which correspond to specific process states. The feature extractor is pretrained using SSL techniques, while the clustering head maps the embeddings to process state clusters using a modified version of the SCAN algorithm.

The overall operation of SensorSCAN follows a three-stage process. In the first stage, self-supervised pretraining is performed, where the feature extractor learns generalizable representations of sensor data. This pretraining is achieved through contrastive learning and masked value reconstruction, both of which enhance the robustness of the feature embeddings. The second stage applies deep clustering, where the embeddings are further refined and assigned to distinct process states. This stage uses the SCAN algorithm to identify clusters that correspond to specific process conditions. Finally, label matching is conducted as a post-processing step. In this step, human experts manually label a small subset of the clusters, enabling the alignment of clusters with known process states.

4.3.2 Self-Supervised Learning

To pretrain the feature extractor, SensorSCAN employs two primary self-supervised learning tasks: masked input reconstruction and contrastive learning.

Masked Input Reconstruction aims to teach the model to reconstruct masked portions of the sensor data. Given an input sensor data matrix $X \in \mathbb{R}^{L \times D}$, where L is the time series length and D is the number of sensors, a binary mask $M \in \{0, 1\}^{L \times D}$ is applied to the data to produce a masked version of the input:

$$X_{\text{masked}} = X \odot M \tag{4.4}$$

where $\odot$ denotes element-wise multiplication. The reconstruction loss is then computed using the masked mean squared error (MSE):

$$L_{\text{rec}} = \frac{1}{|M|} \sum_{(i,j) \in M} (R(T(X_{\text{masked}}))_{ij} - X_{ij})^2 \tag{4.5}$$

where $T(\cdot)$ is the feature extractor and $R(\cdot)$ is the reconstruction head.

Contrastive Learning focuses on distinguishing positive sample pairs from negative sample pairs. Positive pairs are created by applying data augmentations (e.g., jittering, scaling, and permutation) to the same input data. The NT-Xent loss is used to learn representations that make positive pairs closer while pushing negative pairs apart. For a batch of $2B$ samples, the NT-Xent loss for a positive

pair (i, j) is defined as:

$$\ell_{i,j} = -\log \frac{\exp\left(\text{sim}(z_i, z_j)/\tau\right)}{\sum_{k=1}^{2B} 1_{[k \neq i]} \exp\left(\text{sim}(z_i, z_k)/\tau\right)} \tag{4.6}$$

where z_i and z_j are the feature embeddings of the positive pair, $\text{sim}(\cdot, \cdot)$ denotes cosine similarity, and τ is a temperature parameter.

The total self-supervised learning loss combines both objectives:

$$L_{\text{SSL}} = L_{\text{rec}} + \lambda_{\text{cont}} L_{\text{cont}} \tag{4.7}$$

where λ_{cont} controls the balance between the reconstruction and contrastive learning objectives.

4.3.3 Deep Clustering

Once the feature extractor has been pre-trained, SensorSCAN performs clustering in the latent space. Unlike traditional clustering approaches, which operate directly on raw sensor data, SensorSCAN operates on the embedding space, where process states are more distinguishable.

The clustering process is guided by the SCAN algorithm, where nearest-neighbor mining is used to identify representative cluster centers. For a sample X, its nearest neighbors $\mathcal{N}(X)$ are identified in the embedding space, and the cluster assignments are updated accordingly. The clustering network C assigns embeddings z to clusters using a softmax activation:

$$\hat{y} = C(z) = \frac{\exp(z)}{\sum_{k=1}^{K} \exp(z_k)} \tag{4.8}$$

where K is the total number of clusters. To prevent trivial clustering solutions, SensorSCAN introduces an entropy-based loss that encourages balanced cluster sizes:

$$L_{\text{entropy}} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} \hat{y}_{i,k} \log \hat{y}_{i,k} \tag{4.9}$$

The final clustering loss combines the alignment loss and entropy regularization:

$$L_{\text{cluster}} = -\frac{1}{N} \sum_{i=1}^{N} \log\left(\hat{y}_i \cdot \hat{y}_{\mathcal{N}(X_i)}\right) + \lambda_{\text{entropy}} L_{\text{entropy}} \tag{4.10}$$

where λ_{entropy} controls the weight of the entropy loss.

4.3.4 Fault Detection and Diagnosis

The process of fault detection in SensorSCAN begins with the extraction of embeddings from new sensor data using the pre-trained feature extractor. These embeddings are then passed through the clustering head, which assigns the data to one of the pre-specified clusters. If the assigned cluster corresponds to a known fault state, a fault is flagged. If the assignment indicates a change in the process state, it may also suggest a transition between process conditions.

To ensure that clusters correspond to meaningful process states, SensorSCAN employs a label matching procedure. For each cluster, the process state is assigned by maximizing the likelihood that a given cluster corresponds to a specific state. Given a set of process labels $\{y_1, y_2, \ldots, y_K\}$, the mapping function $f : \mathbb{N} \to \{1, 2, \ldots, Q\}$ for Q process states is found by maximizing the correspondence:

$$f(k) =_{q \in \{1,2,\ldots,Q\}} \sum_{i=1}^{N} 1_{[c_i = k \wedge y_i = q]} \tag{4.11}$$

where c_i denotes the cluster index for sample i, and y_i is the process state label.

4.4 CPC

CPC, introduced in [van den Oord et al., 2019], is an SSL technique that learns representations by predicting the future in latent space. This approach has been adapted for anomaly detection tasks in this thesis due to its ability to capture complex temporal dependencies in time series data.

4.4.1 Architecture Overview

The CPC architecture is composed of three interconnected main components, each serving a distinct purpose in the overall framework. At the forefront is the encoder, which compresses the input data into a compact latent representation, effectively extracting the essential features of the data. Following this, an autoregressive model processes these latent representations, capturing and modeling the temporal dependencies within the latent space. This temporal context is then utilized by the

prediction network, the third component, which generates future predictions based on the contextual information provided by the autoregressive model. This structure enables the CPC architecture to efficiently process sequential data, learn meaningful representations, and make informed predictions about future states.

4.4.2 Encoder

The encoder g_{enc} maps the input sequence x_t to a latent representation z_t:

$$z_t = g_{enc}(x_t) \tag{4.12}$$

For time series data, the encoder is typically implemented as a convolutional neural network to capture local patterns efficiently.

4.4.3 Autoregressive Model

The autoregressive model g_{ar} processes the sequence of latent representations to produce a context vector c_t:

$$c_t = g_{ar}(z_{\leq t}) \tag{4.13}$$

In this architecture, a Gated Recurrent Unit (GRU) for the autoregressive model is used due to its ability to capture long-term dependencies as stated in the source paper.

4.4.4 Prediction Network

The prediction network g_{pred} generates predictions for future latent representations based on the context vector:

$$\hat{z}_{t+k} = g_{pred}(c_t, k) \tag{4.14}$$

where k is the number of steps into the future for which we're predicting.

The entire CPC process can be summarized as a single function that takes an input sequence x_t and predicts future latent representations $\hat{z}_{t+k}$:

$$\text{CPC}(x_t, k) = g_{pred}(g_{ar}(g_{enc}(x_{\leq t})), k) \tag{4.15}$$

4.4.5 InfoNCE Loss

The CPC model is trained using the InfoNCE (Information Noise Contrastive Estimation) loss, which is a form of contrastive learning. This loss function is designed to maximize the mutual information between the context and the future representations while being computationally tractable.

For a given context c_t and a set of N candidate future representations $\{z_{t+k}\}$ (including one positive sample and $N-1$ negative samples), the InfoNCE loss is defined as:

$$\mathcal{L} = -\mathbb{E}\left[\log \frac{\exp(z_{t+k}^T \hat{z}_{t+k}/\tau)}{\sum_{i=1}^{N} \exp(z_i^T \hat{z}_{t+k}/\tau)}\right] \tag{4.16}$$

where z_{t+k} is the true future representation (positive sample), and $\hat{z}_{t+k}$ is the predicted future representation, and z_i are the candidate representations (including both positive and negative samples), and τ is a temperature parameter that controls the distribution's concentration.

4.4.6 Intuition and Mechanics of CPC

CPC learns representations by predicting future states in a latent space. The key intuition behind CPC is its contrastive approach, which compares the similarity between the predicted future representation and the true future (positive sample) against alternative (negative) samples. This contrast drives the model to learn representations that can effectively distinguish the true future from other possible outcomes, improving its discriminative power.

By doing this, CPC aims to maximize the mutual information between the current context, derived from past inputs, and the future representations. This is

achieved by minimizing the InfoNCE loss, which encourages the model to capture the most relevant information for predicting future states. In time series data, where temporal dependencies are critical, this results in representations that are well-suited for tasks like anomaly detection.

The structure of the InfoNCE loss is similar to that of a softmax function, where the correct future prediction is maximized against all possible alternatives. This probabilistic framework efficiently incorporates negative sampling, allowing the model to learn from many comparisons without the computational cost of evaluating all future states.

Furthermore, the temperature parameter (τ) plays an important role by adjusting the sharpness of the predictions. Lower values make the model more confident in its predictions, while higher values temper this confidence, leading to more conservative predictions. This flexibility ensures that the model can balance confidence and caution, depending on the task's needs.

4.4.7 Fault Detection Process

For fault detection, the CPC model is trained on normal operating data. During inference, we can use the prediction accuracy or the InfoNCE loss as an anomaly score. Significant deviations from expected predictions may indicate the presence of a fault.

4.5 Time-Frequency Consistency (TF-C)

Time-Frequency Consistency (TF-C), proposed by [Zhang et al., 2024b], is a self-supervised pre-training approach for time series that leverages the consistency between time-based and frequency-based representations. This method aims to create generalizable representations that can be effectively transferred across different time series datasets.

4.5.1 Architecture Overview

The TF-C architecture consists of four main components: a time-based encoder G_T and a frequency-based encoder G_F, which process the raw time series data and its frequency representation respectively, and two cross-space projectors R_T and R_F that map time-based and frequency-based representations to a joint time-frequency space. Both encoders utilize 3-layer 1D ResNets as their backbone, allowing for efficient processing of time series data.

4.5.2 Time-based and Frequency-based Contrastive Encoders

For an input time series x_i, the time-based encoder generates augmentations through a bank B_T, such that $\tilde{x}_i^T \in B_T(x_i^T)$. The encoder G_T then maps these to embeddings: $h_i^T = G_T(x_i^T)$ and $\tilde{h}_i^T = G_T(\tilde{x}_i^T)$. The time-based augmentation bank includes techniques such as jittering, scaling, time-shifts, and neighborhood segments. Similarly, the frequency representation x_i^F is obtained through Fourier Transform, and augmentations are created by adding or removing frequency components: $\tilde{x}_i^F \in B_F(x_i^F)$. The frequency encoder G_F processes these to produce $h_i^F = G_F(x_i^F)$ and $\tilde{h}_i^F = G_F(\tilde{x}_i^F)$. The frequency-based augmentations involve perturbing the frequency spectrum by manipulating a small number of frequency components.

4.5.3 Contrastive Losses

Both encoders in the TF-C model are trained using a contrastive loss that is similar to, but distinct from, the NT-Xent (Normalized Temperature-scaled Cross Entropy) loss. For the time-based encoder, the loss is defined as:

$$L_{T,i} = d(h_i^T, \tilde{h}i^T, D^{\text{pret}}) = -\log \frac{\exp(\text{sim}(h_i^T, \tilde{h}i^T)/\tau)}{\sum x_j \in D^{\text{pret}} 1i \neq j \exp(\text{sim}(h_i^T, G_T(x_j))/\tau)} \quad (4.17)$$

where $\text{sim}(u, v)$ denotes the cosine similarity between vectors u and v, τ is a temperature parameter, and $1_{i \neq j}$ is an indicator function that equals 1 when $i \neq j$ and 0 otherwise. D^{pret} represents the entire pre-training dataset. The intuition behind this loss is to encourage the encoder to generate similar representations for augmented versions of the same input (positive pairs) while pushing apart representations of

different inputs (negative pairs). The numerator $\exp(\text{sim}(h_i^T, \tilde{h}_i^T)/\tau)$ represents the similarity between the original input and its augmented version, which we want to maximize. The denominator sums the similarities between the original input and all other inputs in the pre-training dataset, acting as a normalization term and pushing different samples apart in the embedding space. By minimizing the negative log of this ratio, we maximize the similarity of positive pairs relative to all potential negative pairs in the dataset. A similar loss $L_{F,i}$ is defined for the frequency-based encoder, applying the same principles in the frequency domain. This loss function differs from both InfoNCE and standard NT-Xent losses in an important way. The TF-C loss uses all other samples in the entire pre-training dataset as negative examples, as reflected in the summation over $x_j \in D^{\text{pret}}$ in the denominator. In contrast, InfoNCE typically uses a large subset of the dataset as negative examples, but not necessarily the entire dataset. Standard NT-Xent loss usually uses only the other samples within the current batch as negative examples.

4.5.4 Time-Frequency Consistency

The key innovation of TF-C is the consistency loss that aligns time-based and frequency-based representations in a joint space. This loss is defined as:

$$L_{C,i} = \sum_{S^{\text{pair}}} (S_i^{\text{TF}} - S_i^{\text{pair}} + \delta) \tag{4.18}$$

where $S_i^{\text{TF}} = d(z_i^T, z_i^F, D^{\text{pret}})$ is the distance between time and frequency embeddings, and S^{pair} includes distances between original and augmented embeddings across domains. This loss ensures that the learned representations from both domains are consistent, enhancing the model's robustness and generalizability.

4.5.5 Total Loss and Training Process

The total pre-training loss combines the contrastive and consistency losses:

$$L_{\text{TF-C},i} = \lambda(L_{T,i} + L_{F,i}) + (1 - \lambda)L_{C,i} \tag{4.19}$$

where λ controls the relative importance of the losses. TF-C is pre-trained on a large unlabeled dataset to learn generalizable representations.

Chapter 5

DATASET AND EXPERIMENTAL SETUP

For our experiments, we utilized four primary datasets: the Tennessee Eastman Process (TEP) Ricker and Rieth datasets, the Secure Water Treatment (SWaT) dataset, and the Water Distribution (WADI) dataset. Relevant dataset statistics are provided in Table 5.2. These datasets are widely recognized benchmarks in the fields of fault detection in process control, anomaly detection, and cyber-physical system (CPS) security. This section provides an overview of each dataset, emphasizing the sources of anomalies and their relevance to the task of fault detection.

5.1 Fault Definition in Industrial Process

Before explaining the datasets, it is important to understand how the faults are defined mathematically. As stated in Chapter 1, the faults are generated on these datasets considering the chemical processes. Thus, it requires domain knowledge to apply the faults in the dataset. In the TEP dataset, faults are defined as deviations from the nominal or healthy state of the process. These deviations can be caused by sudden changes, random fluctuations, or mechanical failures in the process. Mathematically, a fault $f(t)$ can be expressed as the difference between the observed process variable $x(t)$ and its expected or nominal value $x_0(t)$:

$$f(t) = x(t) - x_0(t)$$

where:

- $x(t)$ is the observed process variable at time t
- $x_0(t)$ is the nominal process variable at time t

While the TEP includes several types of faults, this section will focus on three key types as examples: step faults, random variation faults, and sticking faults. These are some of the most common and impactful faults in the datasets.

1. Step Faults

A step fault represents a sudden, abrupt change in the value of a process variable. This type of fault is often caused by valve malfunctions, abrupt shifts in feed composition, or changes in setpoints. Mathematically, a step fault is represented as:

$$x(t) = \begin{cases} x_0(t), & \text{if } t < t_0 \\ x_0(t) + \Delta x, & \text{if } t \geq t_0 \end{cases}$$

where:

- $x_0(t)$ is the nominal value of the process variable at time t
- Δx is the magnitude of the change
- t_0 is the time at which the change occurs

Step faults are particularly relevant when investigating the system's ability to respond to sudden disturbances. These faults are typically easy to detect due to their clear and abrupt nature, but their impact on system stability can be significant.

2. Random Variation Faults

Random variation faults are caused by stochastic fluctuations in process variables. This variability may arise from noise, changes in environmental conditions, or fluctuations in the quality of incoming materials. These faults are mathematically modeled as:

$$x(t) = x_0(t) + r(t)$$

where:

- $x_0(t)$ is the nominal value of the process variable at time t

- $r(t)$ is a random variable that follows a statistical distribution (often Gaussian) with specified mean and variance

Unlike step faults, random variation faults are continuous, and their randomness follows a specified statistical distribution. The unpredictable nature of these faults makes them more challenging to detect, as they can resemble natural process noise.

5.1.1 3. Sticking Faults

Sticking faults typically occur in actuators, valves, or control devices that become stuck and unresponsive to control inputs. This causes the process variable to remain constant despite control actions. Mathematically, a sticking fault is described as:

$$x(t) = \begin{cases} x_0(t), & \text{if } t < t_0 \\ c, & \text{if } t \geq t_0 \end{cases}$$

where:

- $x_0(t)$ is the nominal value of the process variable at time t
- c is the constant value at which the process variable becomes stuck
- t_0 is the time at which the fault occurs

These faults are particularly dangerous as they prevent control actions from influencing the affected variable. For example, if a cooling water outlet valve sticks, the cooling system's temperature cannot be adjusted, which could lead to overheating.

5.2 Datasets

5.2.1 Tennessee Eastman Process Dataset Overview

The original Tennessee Eastman Process (TEP) dataset, introduced by Downs et al. in 1993 [Downs and Vogel, 1993], is a fundamental benchmark for anomaly detection in process engineering. It simulates a complex chemical production process involving eight chemical materials: four starting materials, two final products, one unwanted

by-product, and one inactive substance. These materials are processed through five key stages: reaction, cooling, separation, compression, and purification, with an additional cleanup step to remove unwanted or inactive materials. The simulation records 52 types of features, including material flow rates, pressure, temperature, and other process variables. Out of these, 11 features are used to maintain smooth operation of the process. This detailed setup provides a realistic representation of a complex industrial process. The original TEP dataset comprises 21 data groups: 20 representing different fault scenarios (Faults 1–20) and one representing normal operation (Fault 0). Table 5.1 details these fault types. However, the original dataset's limitation of providing only one instance per fault type is not ideal for deep learning models, which require diverse data for accurate learning.

5.2.2 TEP Rieth Dataset

To address the limitations of the original dataset, the extended TEP dataset (TEP Rieth) was introduced in [Rieth et al., 2018]. This dataset significantly expands on the original by applying different random seeds during the simulation process, generating multiple scenarios for each fault type. The TEP Rieth dataset retains the original dataset's structure while substantially increasing the data volume. It includes 500 training instances and 960 testing instances for each fault type, using various random seeds to introduce slight variations and enhance dataset diversity. Each instance encompasses 52 attributes sampled at three-minute intervals, providing 25 hours of training data and 48 hours of testing data. This expanded dataset provides a richer level of detail for analyzing the process and serves as a more powerful tool for deep learning models in FD.

5.2.3 TEP Ricker Dataset

The TEP_Ricker dataset, proposed in [Reinartz et al., 2021], further expands on the original TEP simulation by introducing additional faults to increase fault diversity. It introduces 8 additional faults beyond the original 20, totaling 28 fault scenarios. The dataset comprises 2000 measurements recorded every 3 minutes for each

simulation, with each simulation spanning 100 hours and a fault introduced at the 30-hour mark. There are 100 simulations for every operational state, totaling 2800 simulations. The TEP_Ricker dataset places emphasis on normal operational conditions, with a significant portion of the dataset representing fault-free operation. The TEP_Ricker dataset's extended simulation time and additional fault scenarios provide a more comprehensive testbed for evaluating FD algorithms.

Both the TEP Rieth and TEP Ricker datasets offer significant advantages over the original TEP dataset for training and evaluating deep learning models in process control and fault detection scenarios. Their increased data volume, diversity of fault scenarios, and extended simulation periods provide a more realistic and challenging environment for developing robust FD systems. Thus, we consider both datasets when evaluating the benchmark models with out proposed model.

5.2.4 Secure Water Treatment (SWaT) Dataset

The Secure Water Treatment (SWaT) dataset [Mathur and Tippenhauer, 2016] is collected from a testbed that emulates a real-world industrial water treatment plant. The SWaT system simulates the operation of a water purification and distribution system, including physical and cyber components such as tanks, pumps, and programmable logic controllers (PLCs). The SWaT dataset records data from 51 sensors and actuators across several operational stages, such as raw water inflow, chemical treatment, filtration, and clean water distribution.

The main source of anomalies in the SWaT dataset comes from intentional attack scenarios injected into the system. During data collection, the system operates under normal conditions for seven days, followed by four days of attack scenarios. The attack scenarios are manually executed and are based on realistic cyber-physical threat models. These attacks include sensor manipulation, actuator disruption, and process disturbances that aim to disrupt normal operations.

The data is organized into several versions, each with slight modifications. The core of the SWaT anomaly detection task is to identify when an attack occurs by distinguishing abnormal sensor readings from normal process variability. The attacks

Table 5.1: Explanation of error categories within the TEP framework. A Fault ID of 0 indicates standard operation, whereas all other IDs state various faults within the system.

Fault ID	**Description**	**Type**
0	Normal operation	None
1	A/C feed ratio shift, B fixed	Step
2	B composition shift, A/C fixed	Step
3	D feed temperature change	Step
4	Reactor cooling water temp change	Step
5	Condenser cooling water temp change	Step
6	Loss of A feed	Step
7	Pressure drop in C header	Step
8	A, B, C feed composition fluctuation	Random Variation
9	D feed temperature fluctuation	Random Variation
10	C feed temperature fluctuation	Random Variation
11	Reactor cooling water temp fluctuation	Random Variation
12	Condenser cooling water temp fluctuation	Random Variation
13	Drift in reaction kinetics	Slow Drift
14	Sticking of reactor cooling valve	Sticking
15	Sticking of condenser cooling valve	Sticking
16–20	Unspecified fault	Unknown

are designed to simulate common industrial control system (ICS) attacks, such as false data injection, command injection, and physical manipulation of actuators. The SWaT dataset is one of the most prominent datasets in CPS anomaly detection due to its large size, realistic attack scenarios, and labeled data.

5.2.5 Water Distribution (WADI) Dataset

The Water Distribution (WADI) dataset [wad, 2017] is a comprehensive benchmark dataset for anomaly detection in water distribution networks. Similar to the SWaT

dataset, it is collected from a testbed that represents a scaled-down version of a real-world water distribution network. The system includes components such as reservoirs, pumps, valves, and pipelines that distribute water to different sectors. The WADI dataset records data from 123 sensors and actuators over 16 days, consisting of 14 days of normal operation and 2 days of attack scenarios.

The anomalies in the WADI dataset originate from deliberate attack scenarios. The attacks aim to disrupt the normal flow of water, alter sensor readings, and force the system into unsafe states. Over the two days of attack scenarios, 15 distinct attack types are executed, each targeting different system components. These attacks are designed to replicate common ICS attacks, such as valve manipulation, pump control overrides, and sensor tampering.

To ensure data quality and label accuracy, the WADI dataset has undergone revisions. In a later version provided in 2019, unstable periods of operation were removed, and more precise attack labels were added. The attacks in the WADI dataset are less frequent compared to the SWaT dataset, making anomaly detection more challenging. This dataset is particularly useful for testing the robustness of anomaly detection models in water distribution networks, where data is typically more sparse and irregular than in tightly controlled environments like the TEP or SWaT systems.

The datasets discussed in this section provide a comprehensive set of benchmarks for testing anomaly detection models. The TEP datasets simulate process faults arising from changes in feed composition, equipment malfunctions, and reaction kinetics. These faults are inherent to the process and reflect common issues encountered in industrial processes. On the other hand, the SWaT and WADI datasets simulate cyber-physical attacks on water treatment and distribution systems. The anomalies in these datasets arise from intentional attacks such as sensor tampering, actuator manipulation, and process disruption. By including all four datasets, our experimental setup covers both process-based anomalies (TEP) and cyber-physical attacks (SWaT, WADI), providing a rigorous testbed for evaluating the robustness and generalization ability of anomaly detection models.

Table 5.2: Statistics of the used datasets.

Datasets	**# of Input Signals**	**# of Anomaly Types**
TEP-Rieth	52	20
TEP-Ricker	52	28
SWAT	51	41
WADI-2019	123	15

5.3 Experimental Setup for FD

5.3.1 Data Preprocessing

Data preprocessing played a crucial role in preparing both datasets for analysis. The process involved three main steps to transform the raw time-series data into a suitable format for our deep learning models. First, we employed **sliding window segmentation** with a window size of 100 time steps, which allowed us to capture temporal patterns while maintaining sufficient context for analysis. Following segmentation, we transformed the time-domain signals into their **frequency representation** using Fourier transform, enabling the extraction of spectral features and reducing the impact of temporal variations. Finally, we applied **z-score normalization** to standardize the data, ensuring that all variables contributed equally to the model regardless of their original scales. This normalization step was particularly important as it helped mitigate the effects of different measurement units and ranges across variables, potentially improving model convergence and performance.

Before starting the pretraining, the dataset was divided into three parts: 80% of the data was used for pretraining, 3% was allocated for clustering, and the remaining 17% was reserved for testing. While splitting the dataset, we ensured that there is no leakage in terms of temporal connections using time series split from sklearn library. This setup ensured that the model had sufficient data to learn robust representations during pretraining, while the clustering and testing subsets allowed for effective evaluation of the learned representations and downstream tasks.

5.3.2 Model Architecture

The architecture comprises two parallel transformer encoder pathways: one for processing time-domain data and another for frequency-domain data. These transformer encoders, unlike traditional 1D ResNet approaches, are specifically designed to capture long-range dependencies in their respective domains. The architecture is augmented with an autoregressive layer that processes the split embeddings to generate context vectors, followed by a projection head that produces future predictions. The integration of these components enables effective representation learning across both temporal and spectral features of the input data.

5.3.3 Training Procedure

The training procedure followed the prediction contrast learning paradigm, where the model simultaneously learns representations in both time and frequency domains. We employed a sliding window approach with a window size of 100 time steps and a step size of 1 to segment the input data. The model was trained for 150 epochs using the Adam optimizer with a learning rate of 4e-5 and a batch size of 256. During training, the past and future input data underwent parallel processing through both encoder pathways, where the embeddings were obtained as past and future embeddings. The autoregressive layer then processed the past embeddings to generate predictions of future states, which were used in the contrastive learning framework. The training objective was optimized using the InfoNCE loss computed separately for both time and frequency domains, with their sum serving as the total loss function. The details of the hyperparameters for the proposed architecture is presented in Table 5.3.

5.3.4 Hyperparameter Tuning

Bayesian Hyperparameter Optimization

We conducted comprehensive hyperparameter optimization using the Weights & Biases (wandb) experiment tracking tool [Biases,]. The optimization process employed **Bayesian optimization**, a sophisticated approach that constructs a probabilistic

Table 5.3: Training Hyperparameters

Hyperparameter	**Value**
Window Size	100 time steps
Step Size	1
Number of Epochs	150
Optimizer	Adam
Learning Rate	4×10^{-5}
Batch Size	256
Loss Function	InfoNCE
Encoder Dimension	512
Num. of Heads	8
Num. of Blocks	6
Dropout Rate	0.1

surrogate model of the objective function, denoted as $f(\mathbf{x})$. Specifically, wandb bayesian optimization algorithm uses **Gaussian Process (GP) regression** to create this surrogate model:

$$f(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{5.1}$$

where $\mu(\mathbf{x})$ is the mean function and $k(\mathbf{x}, \mathbf{x}')$ is the covariance function, which quantifies the relationship between different points in the hyperparameter space. By leveraging this surrogate model, Bayesian optimization is able to efficiently explore the hyperparameter space $\mathbf{x}$ compared to traditional methods like grid or random search, as it learns from previous evaluations to guide future choices.

The optimization process selects the next set of hyperparameters to evaluate by using an **acquisition function** to determine the most promising areas of the hyperparameter space. In our case, we used the **Expected Improvement (EI)** acquisition function:

$$\mathbf{x}_{\text{next}} = \arg\max_{\mathbf{x}} \mathbb{E}\left[\max(f(\mathbf{x}) - f(\mathbf{x}^+), 0)\right] \tag{5.2}$$

where $f(\mathbf{x}^+)$ is the best value observed so far, and the expectation is taken with respect to the posterior distribution of the surrogate model. This approach effectively balances the trade-off between **exploration** (sampling uncertain areas) and **exploitation** (focusing on regions with promising results). The acquisition function can also be expressed in terms of the posterior mean and standard deviation of the model:

$$\mathbf{x}_{\text{next}} = \arg\max_{\mathbf{x}} \left(\mu(\mathbf{x}|\mathcal{D}) + \beta\sigma(\mathbf{x}|\mathcal{D})\right) \tag{5.3}$$

where $\mathcal{D}$ represents the set of previously evaluated points, $\sigma(\mathbf{x}|\mathcal{D})$ is the uncertainty estimate, and β is a parameter that adjusts the exploration-exploitation trade-off.

Multi-Objective Optimization

Our optimization process targeted multiple key objectives simultaneously, formulated as a vector-valued objective function:

$$\mathbf{f}(\mathbf{x}) = [\text{TPR}(\mathbf{x}), \text{CDR}(\mathbf{x}), -\text{ADD}(\mathbf{x}), -\text{FPR}(\mathbf{x})] \tag{5.4}$$

where **TPR** is the True Positive Rate, **CDR** is the Correct Diagnosis Rate, **ADD** is the Average Detection Delay, and **FPR** is the False Positive Rate. The negative signs for ADD and FPR indicate that these metrics are to be minimized. To evaluate trade-offs between these objectives, we utilized all metrics contribution to the objective so that we preserve the optimal configuration that offers a balanced trade-off between detection accuracy and timeliness.

5.3.5 Computational Resources

The deep learning experiments were executed on an NVIDIA Tesla A100 GPU. The software environment includes PyTorch 2.2.1, Python 3.8.8, and CUDA 11.3.

5.4 Experimental Setup for Network Simulations

This section provides the details of the experimental setup for the network simulations used to evaluate the deployment efficiency of our DL model for FD within an integrated edge-cloud network architecture. The whole network experiments are conducted on YAFS Simulator [Lera et al., 2019] that is proposed specifically for fog computing networks. Since it is the latest framework that meets our needs, we choose this simulator for our network experiments.

5.4.1 Network Architecture

Our simulation evaluates the deployment efficiency of a DL model for FD by focusing on the flow of TEP data and the associated computational and network demands. The simulation mimics real-world industrial environments without directly processing raw data. Instead, it abstracts key characteristics of the DL model, such as input dimensions, preprocessing, and inference complexity, to generate a realistic representation of system behavior.

The simulated environment distributes TEP data across various components using predefined messages, which encapsulate the computational requirements and data volumes typical in industrial FD applications. These messages represent raw sensor data, preprocessed information, inference results, and control decisions. The computational requirements and message sizes for each module are as follows: **M.Sensor** requires 1,000 instructions and transmits 53,000 bytes, representing raw TEP data. **M.Datalogger** processes this data with 1,000 instructions and produces 5,300 bytes of preprocessed output. The **M.Inference** module performs the inference task with $1{,}000 \times 10^6$ instructions while sending 100 bytes, reflecting the resource-intensive nature of DL inference. Finally, the **M.Decision** module uses 1,000 instructions to generate 100 bytes of control signals. The compact version of this explanation is provided in 5.4.1

The deployment of these modules follows two distinct strategies: **Edge Computing** and **Cloud Computing**. In the edge scenario, the inference is performed locally on an edge device to minimize latency, while in the cloud scenario, inference

Message	**Instructions**	**Bytes**	**Description**
M.Sensor	1,000	53,000	Raw TEP data collection
M.Datalogger	1,000	5,300	Preprocesses raw data
M.Inference	$1,000 \times 10^6$	100	Resource-intensive DL inference
M.Decision	1,000	100	Generates control signals

Table 5.4: Instruction, Transmission, and Description for Each Message

is executed on a remote cloud server, leveraging its greater computational power. Table 5.5 outlines the module deployment for each scenario.

Table 5.5: Deployment of Modules in Edge and Cloud Computing Scenarios

Module	**Edge Deployment**	**Cloud Deployment**
Sensor (M.Sensor)	Sensor	Sensor
Datalogger (M.Sensor, M.Datalogger)	Datalogger	Datalogger
Inference (M.Datalogger, M.Inference)	Edge Device	Cloud Entity
Decision (M.Inference, M.Decision)	Cloud Entity	Cloud Entity
Action (M.Decision)	Actuator	Actuator

In both deployment strategies, the Decision Module is hosted in the cloud for centralized control, while actuators are located locally to execute system actions. The flow of data follows the sequence: Sensor → Data Logger → Inference → Decision → Actuator. The differences between the two scenarios primarily involve the location of the Inference module and the propagation delays for remote communication.

Network Characteristics

The network characteristics, including bandwidth and propagation delays, were modeled to reflect the constraints of real-world industrial setups. Table 5.6 presents the bandwidth and propagation delay (PR) values for communication links under Edge and Cloud deployments.

Table 5.6: Bandwidth and Propagation Delay for Communication Links (Edge vs Cloud Deployment)

Link	**BW**	**PR (Edge)**	**PR (Cloud)**
Sensor → Data Logger	100 Bytes	4 ms	4 ms
Data Logger → Inference	100 Bytes	2 ms	20 ms
Inference → Decision	100 Bytes	20 ms	4 ms
Decision → Actuator	100 Bytes	20 ms	20 ms

In the edge scenario, the data logger and inference module are co-located, resulting in a propagation delay of only 2 ms. However, transmitting inference results to the remote Decision module incurs a 20 ms delay. Conversely, in the cloud scenario, the data logger communicates with the remote inference module over a wide-area network (WAN), leading to a 20 ms delay. The decision module, being cloud-hosted, processes inference results with minimal latency (4 ms). Communication with the local actuators remains constant at 20 ms in both deployments due to remote decision-making.

Computational Resources

The processing capacity of each module is defined by its Instructions Per Time (IPT), representing the number of instructions processed per second. Table 5.7 provides an overview of the IPT values for the modules in the network.

The cloud server has the highest computational capacity, capable of processing 44.8×10^9 instructions per second. This allows for rapid inference despite higher network latencies. The edge device and data logger have IPT values of 2.8×10^9 instructions per second, sufficient for preprocessing and local inference tasks. Sensors and actuators do not process instructions, focusing on data generation and action execution, respectively.

Table 5.7: Processing Capacity (IPT) of Network Components

Module	**Description**	**IPT (Instructions per Second)**
Sensor	Sensor device	N/A
Data Logger	Logs and preprocesses data	2.8×10^9
Edge Device	Local inference device	2.8×10^9
Cloud Server	Cloud server for inference	44.8×10^9
Proxy Server	Intermediary node	2.8×10^9
Decision Module	Decision logic	2.8×10^9
Actuator	Executes system actions	N/A

Analysis of Trade-offs and Bottlenecks

The revised propagation delays reveal the trade-offs between edge and cloud deployments. In edge deployment, the proximity of the inference module to the data logger minimizes the propagation delay for the Data Logger → Inference link (2 ms). However, the Inference → Decision link incurs a significant delay (20 ms) due to the remote Decision module. In Cloud Deployment, centralizing both inference and decision-making in the cloud simplifies communication between these modules, reducing the Inference → Decision delay to 4 ms. However, the Data Logger → Inference link experiences a 20 ms delay due to WAN communication.

The bottlenecks primarily arise from remote communication, especially in the cloud scenario, where delays in the Data Logger → Inference link can impact real-time performance. Edge deployment offers a lower latency alternative for localized setups, but requires transmitting intermediate results to the cloud for centralized decision-making. The actuator communication remains consistent across both scenarios, reflecting the latency introduced by the remote decision module.

5.4.2 Network Components

The architecture shown in Figure 5.1 comprises several key components. A **Cloud Entity** handles cloud-based inference and decision-making, while a **Proxy** simulates

network communication between edge devices and the cloud. **Edge Devices** process local inference to reduce latency, and a **Data Logger** preprocesses and stores sensor data. **Sensors** generate and transmit raw data, and an **Actuator** executes actions based on system decisions. The main distinction between edge and cloud computing scenarios lies in the location of the Inference Module. In edge computing, it is deployed on edge devices to reduce latency, whereas in cloud computing, it stays in the cloud to leverage greater computational power.

The TEP dataset, although not inherently structured as a network, is modeled in this simulation to represent the flow of data across different components, such as sensors, data loggers, and inference devices. In this context, nodes represent the individual components (e.g., sensors, data loggers, edge devices), while edges represent the data transmission or communication links between them. These nodes and edges form a computational network that mirrors the distributed nature of edge-cloud architectures used for processing sensor data, rather than directly modeling the physical or chemical processes.

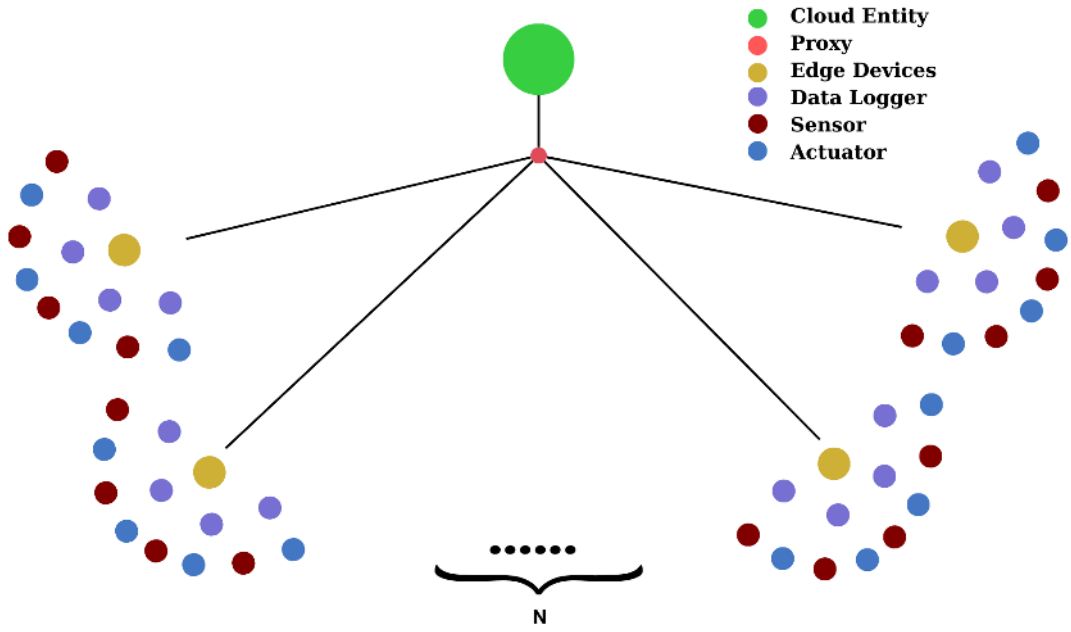

Figure 5.1: Network architecture of the experimented simulation.

5.4.3 Network Simulations

To evaluate the system's behavior under different deployment scenarios, we conducted network simulations representing both edge and cloud computing environments. These simulations were designed to mirror the practical deployment of the proposed DL model in industrial settings.

Simulation Setup

The network simulations were set up with the following configurations:

- Number of plants: 2, 4, 8, 16, and 32
- Simulation time steps: 10^3, 10^4, and 10^5
- Computing environments: Edge and Cloud

Latency Measurement

The latency in the network simulations was calculated as the time taken for a message to travel from the source module to the destination module, defined as:

$$\text{Latency} = t_{\text{reception}} - t_{\text{emit}} \tag{5.5}$$

Where:

- t_{emit} is the time the message is sent from the source
- $t_{\text{reception}}$ is the time the message is received by the destination

The average latency was computed by averaging the latencies across all modules, providing an overall system performance metric.

Chapter 6

ABLATION STUDIES AND EXPERIMENTAL RESULTS

In this chapter, we present a comprehensive analysis of the PCTF model, focusing on both ablation studies and experimental results across the benchmark datasets specified in 5. The ablation studies offer insights into the contributions of key model components, such as transformer encoders, auto-regressive GRUs, and other architectural choices. By systematically isolating and modifying these elements, we identify the components that have the most significant impact on model performance, thereby guiding model refinement and optimization. Following the ablation studies, we provide an extensive evaluation of the model's performance on the TEP datasets, Tep Rieth and Tep Ricker, under both cloud and edge deployment scenarios. The experimental results are benchmarked against several state-of-the-art models using key FD metrics, including TPR, FPR, CDR, and ADD. The experiments were conducted using a fixed random seed of 42 to ensure reproducibility. The result table is presented in 6.3 for each method and dataset. We have also conducted radar chart illustration for each metric TPR, FPR, CDR, and ADD, in Fig. 6.5, 6.6, 6.7, 6.8, respectively.

6.1 Ablation Studies

This section presents tests to see how each part of the model affects overall performance. By isolating components like transformer encoders, the study shows which parts of the model contribute most to the performance. These insights help refine the model and demonstrate why each component is important.

6.1.1 Number of Transformer Encoders

The experiments were conducted by varying the number of Transformer Encoders for both time and frequency components, ranging from 1 to 5 encoders. For each configuration, results on the Tep Rieth and Tep Ricker datasets are presented in Table 6.1.

The performance metrics, as shown in Table 6.1, demonstrate a notable trend across different numbers of encoders, with the optimal performance observed at **3 encoders** for both time and frequency components. Specifically, the **TPR**, **CDR**, and **ADD** all indicate that the 3x3 configuration strikes the best balance between model complexity and effective anomaly detection.

For both the Tep Rieth and Tep Ricker datasets, the **TPR** increased consistently from 1 to 3 encoders, reaching peak performance at 3 encoders. After 3 encoders, the TPR began to plateau or slightly decline, suggesting diminishing returns in terms of detection accuracy. Similarly, the **FPR** generally decreased as the number of encoders increased, with the lowest values observed at 3 encoders. Beyond this point, a slight increase in FPR was noted, indicating potential overfitting and reduced specificity.

The **CDR** followed a similar trend as TPR, with the highest values recorded for the 3x3 encoder configuration. The CDR peaked at 0.8871 for Tep Rieth and 0.9762 for Tep Ricker, confirming that the model's diagnostic accuracy was optimal at this configuration. Furthermore, the **ADD** showed significant improvement (decrease) as the number of encoders increased up to 3, reaching the minimum ADD values of **25.74** for Tep Rieth and **22.18** for Tep Ricker. Beyond this configuration, ADD values started to increase, indicating slower detection times due to model overfitting or increased complexity.

The key insights from Table 6.1 are as follows: the optimal configuration of 3 encoders provides the highest overall detection accuracy while minimizing the detection delay. Increasing the number of encoders beyond 3 led to increases in FPR and ADD, indicating a trend towards overfitting. The Tep Ricker dataset consistently outperformed the Tep Rieth dataset in terms of both TPR and CDR,

suggesting that the model could more effectively detect anomalies in the Tep Ricker data.

The best performance for both datasets was observed with 3 Transformer Encoders for both time and frequency components. This configuration resulted in the highest TPR and CDR, along with the lowest FPR and ADD, indicating effective anomaly detection with quick response times. Adding more encoders (4 or 5) led to overfitting, reduced generalizability, and increased computational overhead without yielding better detection performance. Thus, using **3 encoders** appears optimal for **PCTF** in terms of accuracy, efficiency, and balanced model complexity.

6.1.2 Auto-regressive GRU Ablations

The ablation study for the auto-regressive layer was conducted to evaluate the impact of using different types of recurrent layers for future context generation. Specifically, we experimented by replacing the GRU with an LSTM or substituting it with a simple RNN. The results of these experiments are presented in Table 6.2.

The performance metrics indicate that the **GRU** provides the most consistent overall performance. Although the **LSTM** shows a slightly higher **TPR** for the Tep Ricker dataset (0.9320 compared to 0.9204 for GRU), the GRU achieves a better balance overall, particularly for both datasets in terms of maintaining a low **FPR** and **ADD**. The GRU demonstrates stability across both datasets, showing that it effectively captures temporal dependencies without overfitting.

The **LSTM** performed well, with slightly better TPR values for the Tep Ricker dataset. However, its overall **CDR** and **ADD** were marginally inferior compared to the GRU. These results suggest that while the LSTM can handle complex dependencies, it does not necessarily outperform the GRU across all metrics. In some instances, the LSTM's higher TPR comes at the cost of a slightly increased FPR and ADD.

The **Simple RNN** configuration displayed the weakest performance among the three models. While it achieved relatively good results, such as a TPR of 0.8805 for the Tep Ricker dataset, its ability to consistently predict correctly, as reflected

by the **CDR**, was lower compared to GRU and LSTM. Additionally, the **ADD** was significantly higher, indicating a slower response to anomalies. The lack of gating mechanisms made it prone to issues such as vanishing gradients, which limited its ability to model long-term dependencies effectively.

Overall, the ablation study highlights the importance of using a GRU for autoregressive modeling in the context of future prediction. The GRU strikes the optimal balance between model accuracy and efficiency, making it the preferred choice over LSTM and Simple RNN layers. The LSTM demonstrated potential advantages in some metrics but did not outperform the GRU consistently across all measures.

The use of a **GRU** layer for future context generation provides the best trade-off between accuracy and general performance. The LSTM's slight advantages in some metrics do not outweigh the GRU's overall stability and efficiency. The Simple RNN, due to its lack of gating, struggles to effectively capture temporal patterns. Thus, the GRU remains the optimal choice for autoregressive modeling in this setup.

6.2 FD Performance on Tep Rieth Dataset

The PCTF model demonstrated superior performance on the Tep Rieth dataset, showcasing its effectiveness in Fault Detection (FD) tasks. A detailed analysis of the results reveals several key insights that highlight the strengths of our approach.

The model achieved the highest TPR of 0.7451, surpassing the next best model, CPC, by 2.06 percentage points (0.7451 vs 0.7245). This significant improvement indicates that our model is more adept at correctly identifying faults when they occur. The higher TPR can be attributed to the model's ability to leverage both time and frequency domain information, allowing it to capture a wider range of fault signatures. This dual-domain approach enables the model to detect subtle anomalies that might be missed when considering only time or frequency information in isolation. While the FPR of 0.0102 was higher than some other models, particularly PCA_KMeans (0.0001) and SensorScan (0.0006), this trade-off is justified by the model's exceptional performance in other critical metrics. The slightly higher FPR suggests that our model might be more sensitive to subtle variations in the data.

In industrial settings, where the cost of missing a fault often outweighs the cost of false alarms, this increased sensitivity can be advantageous. It's important to note that in real-world applications, this slight increase in FPR can be mitigated through proper threshold tuning and post-processing techniques. The PCTF model achieved the highest CDR of 0.8871, significantly outperforming the next best model, SensorScan (0.8513). This 3.58 percentage point improvement indicates that our model not only detects faults more accurately but also classifies them correctly more often. The high CDR can be attributed to the model's ability to learn more nuanced representations of different fault types through its predictive contrast approach. By learning to predict future states in both time and frequency domains, the model develops a more comprehensive understanding of normal operation patterns and various fault signatures.

The AUC analysis highlights the PCTF model's superiority, achieving the highest score of **0.9291**. This indicates robust performance across all decision thresholds. The second-best AUC, achieved by CPC (0.9075), reinforces the PCTF model's ability to balance sensitivity and specificity effectively. The high AUC value aligns with its superior TPR and CDR metrics, substantiating the model's efficiency in fault detection and classification. The illustration of the roc curve with different thresholds are presented in Fig. 6.1.

Perhaps the most striking improvement is in the ADD, where our model achieved 25.74 steps, a substantial reduction compared to the next best model, CPC, which had an ADD of 43.67 steps. This 41.06% reduction in detection delay is crucial in industrial settings where quick response to anomalies can prevent costly failures. The improved ADD can be linked to the model's predictive nature, which allows it to anticipate potential faults based on learned patterns in both time and frequency domains. This predictive capability enables the model to detect deviations from normal operation earlier, even before they manifest as full-fledged faults. The model's performance on the Tep Rieth dataset underscores its ability to capture complex temporal dependencies in dynamic industrial processes. The combination of time and frequency domain information, coupled with the predictive contrast approach, allows the model to extract meaningful patterns from the data, resulting

in more accurate and timely fault detection compared to other approaches. This multi-faceted approach enables the model to adapt to the complex, non-linear nature of industrial processes, where faults may manifest in various ways across different timescales and frequency bands. The significant improvements across multiple metrics suggest that our approach is not just incrementally better, but represents a substantial advancement in FD capabilities. The trade-off between slightly higher FPR and significantly improved TPR, CDR, ADD, AUC, and ADD aligns well with the priorities in industrial fault detection, where early and accurate fault identification is paramount. This performance profile makes our model particularly suitable for critical industrial applications where the cost of undetected faults far outweighs the cost of investigating occasional false alarms.

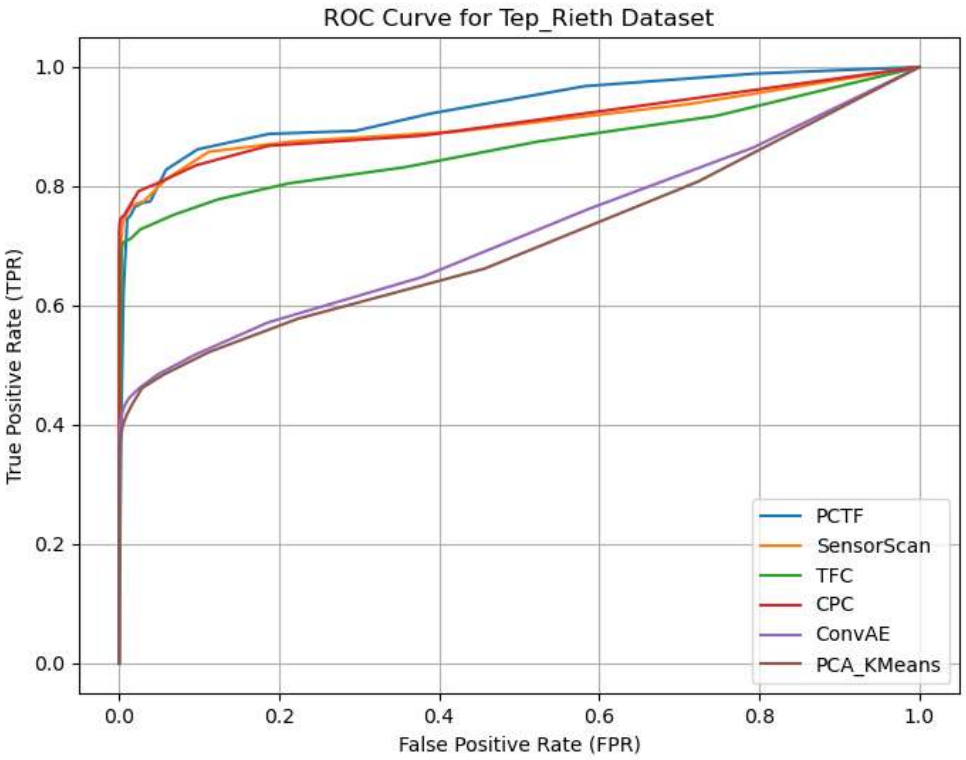

Figure 6.1: Illustration of the AUC results for TEP Rieth dataset.

6.3 FD Performance on Tep Ricker Dataset

On the Tep Ricker dataset, the PCTF model further solidified its position as the top-performing FD solution. The results on this dataset not only confirm the model's effectiveness but also demonstrate its robustness across different industrial process simulations.

The model achieved the highest TPR of 0.9204, demonstrating its exceptional ability to identify faults correctly. This performance is particularly impressive compared to the next best model, SensorScan, which achieved a TPR of 0.9013. The 1.91 percentage point improvement indicates that our model is more sensitive to fault conditions, likely due to its ability to capture both time and frequency domain patterns. This high TPR is crucial in industrial settings where missing a fault can lead to catastrophic consequences. The model's ability to correctly identify over 92% of fault conditions represents a significant advancement in industrial process safety and reliability. Despite a slightly higher FPR of 0.0092 compared to some other models (e.g., PCA_KMeans at 0.0001 and SensorScan at 0.0002), the overall detection efficiency of the model makes it the most suitable choice for real-time FD in complex industrial processes. The higher FPR should be interpreted in the context of the significantly improved TPR and other metrics. In many industrial applications, the cost of a false positive (unnecessary inspection) is far less than the cost of a false negative (missed fault). Therefore, this slight increase in FPR is an acceptable trade-off for the substantial improvements in fault detection capabilities. The model achieved the highest CDR of 0.9762, significantly outperforming the next best model, PCA_KMeans (0.9562). This 2 percentage point improvement indicates that our model not only detects faults but does so with high accuracy. The high CDR suggests that the model is adept at distinguishing between different types of faults, which is crucial for efficient fault management in industrial settings. This capability can significantly streamline maintenance processes by allowing technicians to quickly identify and address the specific type of fault occurring. The ADD of 22.18 steps was the lowest among all models, with the next best being SensorScan at 30.98 steps. This 28.41% reduction in detection delay showcases the

PCTF model's ability to detect faults with minimal latency. The improvement in ADD can be attributed to the model's predictive nature and its ability to process both time and frequency domain information simultaneously. In industrial processes where every second counts, this reduction in detection delay can be the difference between a minor interruption and a major system failure.

On the Tep Ricker dataset, the PCTF model achieved an AUC score of **0.9805**, closely following SensorScan (**0.9820**). This slight difference demonstrates that both models perform exceptionally well across varying thresholds. However, when combined with the PCTF's higher TPR and lower ADD, it offers a better balance of sensitivity and specificity. The AUC evaluation further validates the PCTF model's robustness in diverse industrial scenarios. The illustration of the roc curve with different thresholds are presented in Fig. 6.2.

The model's consistent high performance across both Tep Rieth and Tep Ricker datasets demonstrates its robustness and adaptability to different industrial scenarios. This consistency is particularly important in real-world applications where processes may vary or evolve over time. The superior performance on the Tep Ricker dataset further validates the effectiveness of combining predictive contrast learning with time-frequency domain information. The significant improvements in TPR, CDR, ADD, and AUC, even at the cost of a slightly higher FPR, align well with industrial priorities. In most industrial settings, the cost of missing a fault or delayed detection far outweighs the cost of investigating a false alarm. The model's ability to detect faults quickly and accurately can lead to reduced downtime and production losses, improved safety by early identification of potential hazards, more efficient maintenance scheduling, and enhanced overall equipment effectiveness.

6.4 FD Performance on SWAT Dataset

The performance of the PCTF model on the SWAT dataset showcases its effectiveness in identifying faults in water treatment systems. The model achieved the highest TPR of 0.9223, outperforming the next best model, SensorScan (TPR: 0.8802), by 4.21 percentage points. This result underscores the model's ability to detect faults in

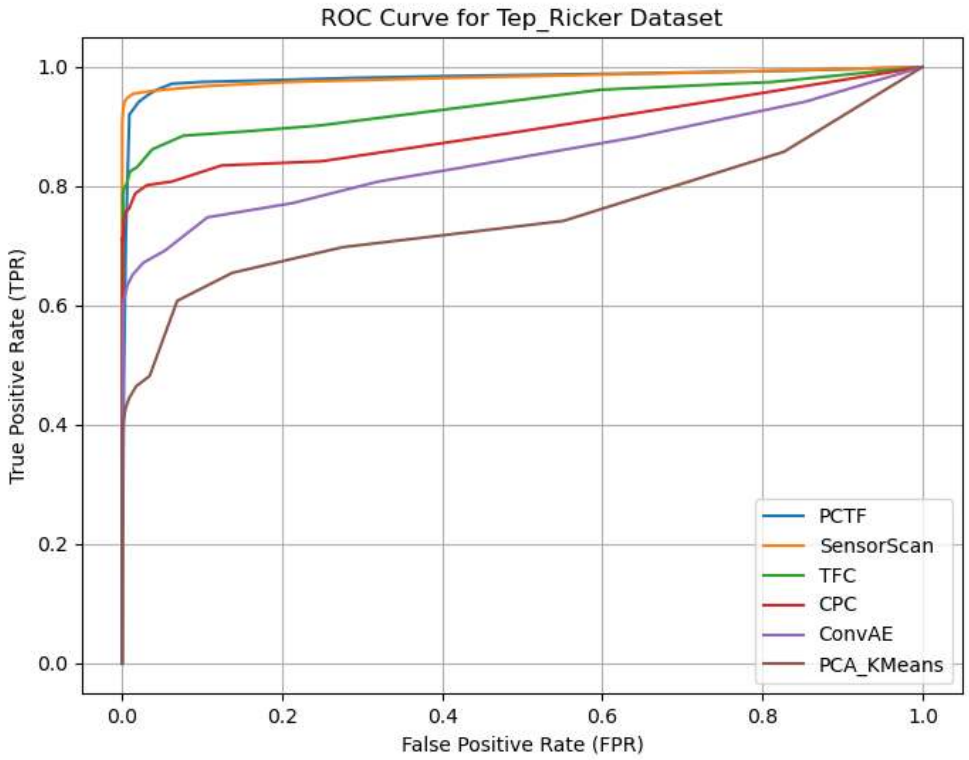

Figure 6.2: Illustration of the AUC results for TEP Ricker dataset.

dynamic, real-time water treatment scenarios, leveraging its dual-domain approach to capture both time and frequency patterns.

On the SWAT dataset, the PCTF model achieved the highest AUC score of **0.9822**. This underscores its superior fault detection capabilities across varying thresholds. The second-best AUC, achieved by SensorScan (0.9639), highlights the significant advantage offered by the PCTF model. The AUC metric complements its high TPR and low ADD, confirming its ability to provide reliable and timely fault detection under dynamic conditions. The illustration of the roc curve with different thresholds are presented in Fig. 6.3.

While the model exhibited a slightly higher FPR of 0.0025 compared to SensorScan (0.0008), the increased TPR and other metrics justify this trade-off. The CDR further reinforces the model's superiority, with a value of 0.9401 compared to 0.8802 for PCA_KMeans, indicating that PCTF not only detects faults but also

classifies them accurately. The high CDR can significantly improve operational efficiency by reducing false diagnostics.

Notably, the PCTF model achieved the lowest ADD of 25.20 steps, a 17.36% improvement over the second-best ADD of 30.50 steps by SensorScan.

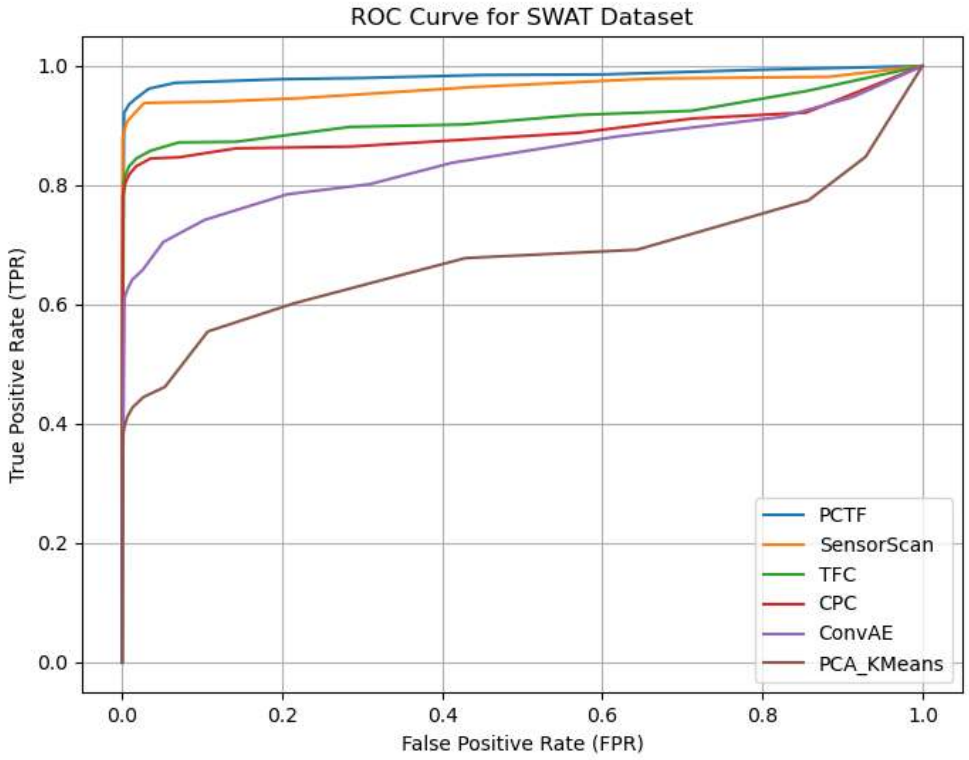

Figure 6.3: Illustration of the AUC results for SWAT dataset.

6.5 FD Performance on WADI Dataset

On the WADI dataset, the PCTF model demonstrated its adaptability to fault detection in water distribution systems. The model achieved the highest TPR of 0.8650, surpassing the second-best model, SensorScan (TPR: 0.8104), by 5.46 percentage points. This improvement emphasizes the model's ability to detect a wide range of faults in complex water distribution networks, ensuring reliability and safety.

The PCTF model achieved the highest AUC score of **0.9594**, surpassing Sen-

sorScan (0.9291) by 3.03 percentage points. This highlights the PCTF model's superior ability to distinguish between normal and faulty states. Combined with its superior TPR, CDR, and ADD metrics, the AUC score confirms the model's robustness and reliability in complex water distribution networks. The illustration of the roc curve with different thresholds are presented in Fig. 6.4.

Despite a marginally higher FPR of 0.0025 compared to SensorScan's 0.0014, the model's superior TPR and CDR more than compensate for this trade-off. The CDR of 0.9152 significantly exceeds the second-best value of 0.8401 (PCA_KMeans), demonstrating the PCTF model's capability to accurately identify fault types, which is critical for targeted maintenance.

The ADD of 35.80 steps is another standout metric, representing a 38.41% improvement over the next best value of 58.12 steps by SensorScan.

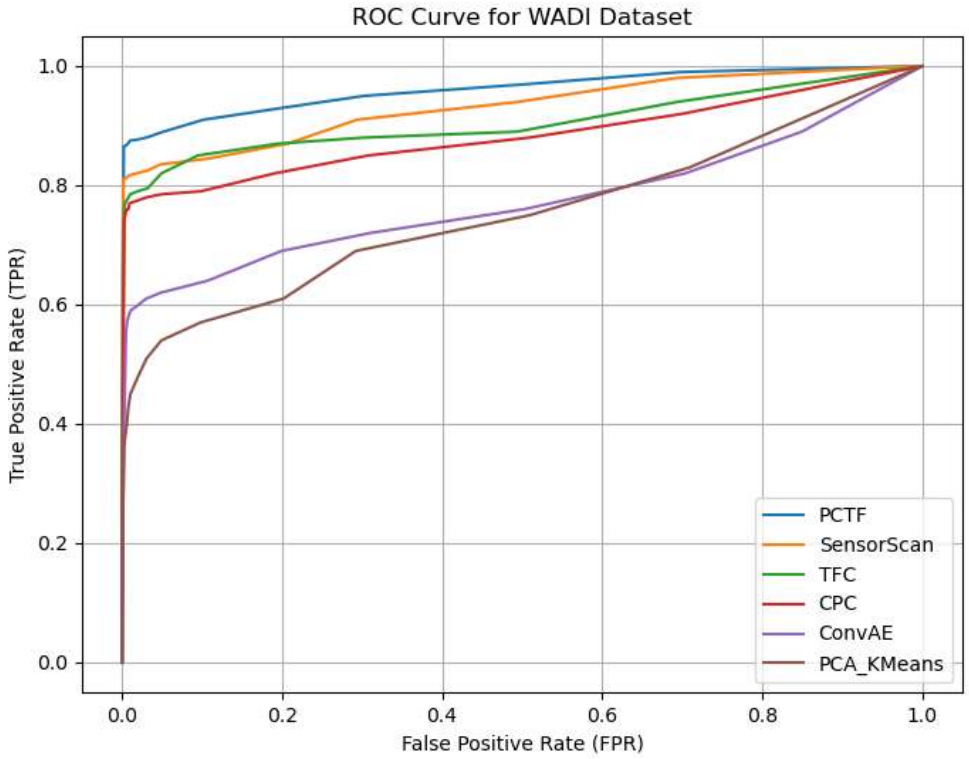

Figure 6.4: Illustration of the AUC results for WADI dataset.

6.6 Cloud vs Edge Deployment Analysis

To assess the practical implications of deploying our PCTF model in real-world industrial settings, we conducted a comprehensive analysis comparing cloud and edge computing scenarios with a variable number of plants and simulation time. It is important to note that this simulation focuses exclusively on the latency and message sizes in the network. The actual inference process is not performed, and no performance metrics such as model accuracy or fault detection are calculated here. Instead, we aim to evaluate the efficiency of the network deployment in terms of latency and scalability. The simulation time is a variable that is provided by YAFS framework [Lera et al., 2019], where we can specify the lenght of the simulation. However, the simulation time is not exactly as real time in seconds but has different characteristics. We conducted the experiments with simulation time and calculated the latency by converting the simulation time to real time in ms using the frameworks methods.

The simulation covered various time scales and plant configurations, as shown in Table 6.4. Our results revealed significant advantages for edge deployment in industrial FD applications:

Latency: Edge deployment maintained consistently low latency of approximately 4.96 ms, regardless of simulation time or the number of plants. This demonstrates excellent scalability and responsiveness, crucial for real-time FD applications. In contrast, cloud deployment showed dramatically increasing latencies in complex scenarios. For instance, at 10^5 simulation time with 32 plants, cloud latency reached 3134.24 ms while edge latency remained at 4.96 ms.

Scalability: The performance gap between cloud and edge deployments widened with increasing simulation complexity. For example, with 32 plants, cloud latency increased from 10.88 ms at 10^3 simulation time to 3134.24 ms at 10^5 simulation time, while edge latency remained nearly constant (4.94 ms to 4.96 ms). This indicates edge computing's superior capability for high-volume data processing in large-scale industrial operations.

Consistency: Edge deployment showed remarkable consistency in performance

across different scenarios. For instance, at 10^5 simulation time, edge latency remained at 4.96 ms across all plant configurations, with only a slight deviation of 5.52 ms in one instance. In contrast, cloud deployment's performance deteriorated significantly with increased complexity, varying from 7.26 ms to 3134.24 ms under the same conditions.

These findings have important implications for deploying advanced FD methods like our PCTF model. The low latency in edge deployment complements the model's superior ADD performance, enabling rapid fault detection and response in industrial settings.

Furthermore, the demonstrated scalability of edge deployment ensures that FD systems can be expanded effectively without compromising performance, even in large-scale industrial operations.

Moreover, edge deployment shows promise for cost-effective implementations, as it can maintain high performance without the need for extensive cloud infrastructure.

To conclude, our analysis strongly supports the adoption of edge computing for deploying the PCTF model in industrial FD applications. The combination of the model's superior fault detection capabilities and the low-latency, scalable nature of edge deployment presents a promising direction for enhancing FD in complex industrial environments. This approach can lead to more efficient, responsive, and cost-effective fault detection systems, improving the reliability and safety of industrial processes.

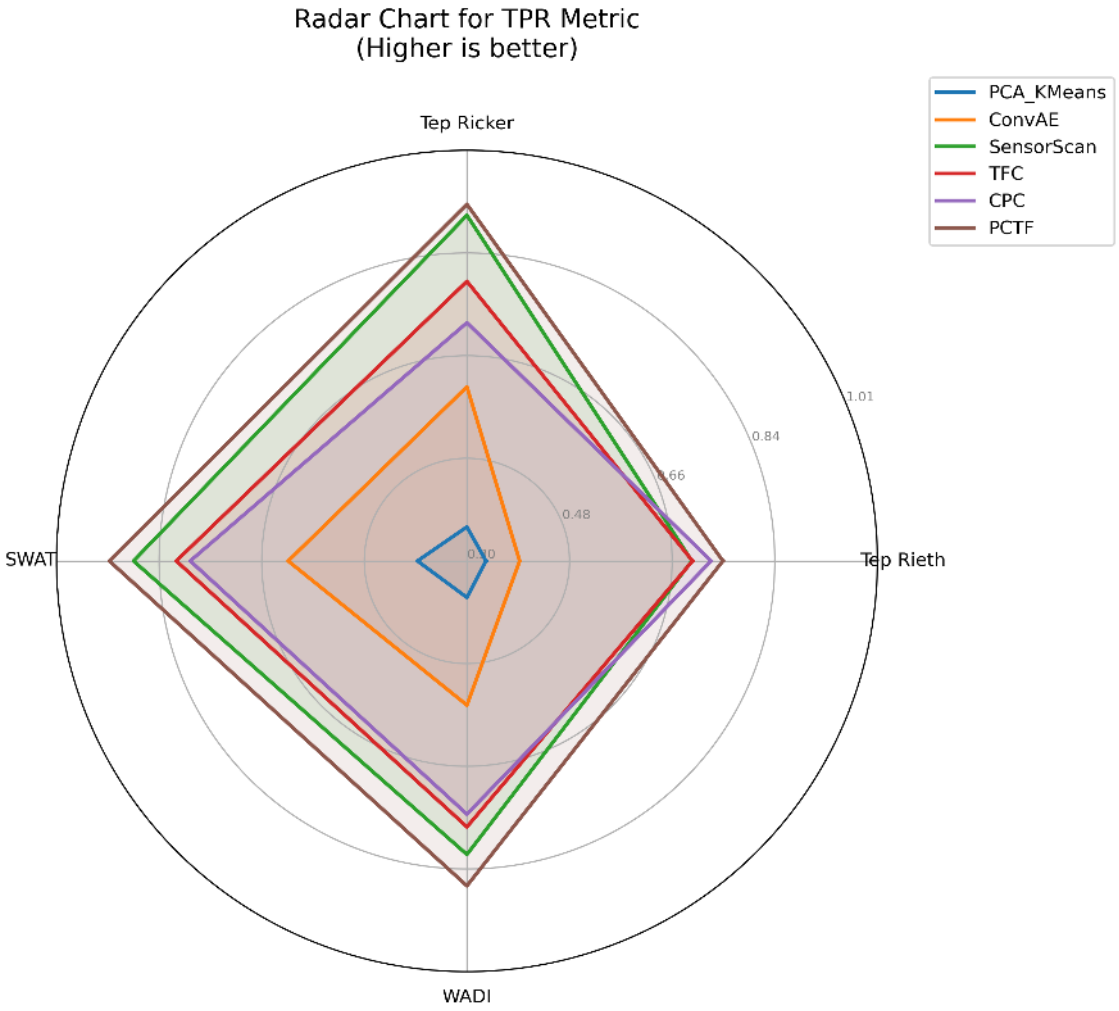

Figure 6.5: Illustration of the TPR results for each dataset and method. Higher is better.

Table 6.1: Test results on Tep Rieth and Tep Ricker datasets for different combinations of transformer encoders. Best values are in bold, and second-best values are in italic.

Time Enc.	**Freq Enc.**	**Tep Rieth Dataset**				**Tep Ricker Dataset**			
		TPR	**FPR**	**CDR**	**ADD**	**TPR**	**FPR**	**CDR**	**ADD**
1	1	0.6201	0.0205	0.6403	75.23	0.6784	0.0189	0.7001	61.45
1	2	0.6352	0.0197	0.6527	68.34	0.6928	0.0176	0.7123	57.21
1	3	0.6489	0.0189	0.6705	62.12	0.7083	0.0168	0.7284	52.33
1	4	0.6421	0.0195	0.6624	65.45	0.7021	0.0173	0.7219	55.67
1	5	0.6303	0.0201	0.6558	70.12	0.6902	0.0181	0.7103	58.49
2	1	0.6651	0.0182	0.6852	60.34	0.7254	0.0165	0.7432	50.28
2	2	0.6923	0.0174	0.7023	54.18	0.7541	0.0158	0.7684	45.67
2	3	0.7058	0.0165	0.7198	48.23	0.7804	0.0149	0.7910	41.78
2	4	0.6982	0.0172	0.7114	52.13	0.7681	0.0153	0.7802	44.32
2	5	0.6867	0.0179	0.7050	58.67	0.7534	0.0161	0.7689	47.59
3	1	0.7002	0.0168	0.7125	48.78	0.7701	0.0152	0.7804	40.92
3	2	0.7241	0.0156	0.7358	42.15	0.8106	0.0143	0.8123	36.54
3	3	**0.7451**	**0.0102**	**0.8871**	**25.74**	**0.9204**	**0.0092**	**0.9762**	**22.18**
3	4	*0.7354*	*0.0151*	*0.7420*	*39.02*	*0.9051*	*0.0140*	*0.8507*	*30.45*
3	5	0.7203	0.0160	0.7301	45.67	0.8903	0.0147	0.8354	33.21
4	1	0.6891	0.0175	0.7056	55.01	0.7582	0.0161	0.7711	47.23
4	2	0.7183	0.0162	0.7227	42.89	0.7984	0.0145	0.8049	38.97
4	3	0.7342	0.0154	0.7394	38.56	0.8845	0.0138	0.8427	30.84
4	4	0.7308	0.0163	0.7387	42.13	0.9017	0.0142	0.8505	32.67
4	5	0.7189	0.0170	0.7258	46.78	0.8905	0.0151	0.8393	35.04
5	1	0.6802	0.0181	0.7003	61.34	0.7404	0.0171	0.7550	50.11
5	2	0.7085	0.0169	0.7201	48.89	0.7887	0.0155	0.7951	38.74
5	3	0.7282	0.0161	0.7354	39.85	0.8802	0.0143	0.8403	30.23
5	4	0.7234	0.0167	0.7298	42.92	0.8701	0.0149	0.8321	33.67
5	5	0.7186	0.0178	0.7241	49.11	0.8901	0.0155	0.8425	36.84

Table 6.2: Test Results on Tep Rieth and Tep Ricker Datasets for Different Recurrent Layers in the Auto-regressive Component. The best values are in **bold**, and the second-best values are in *italic*.

Layer Type	**Tep Rieth Dataset**				**Tep Ricker Dataset**			
	TPR	**FPR**	**CDR**	**ADD**	**TPR**	**FPR**	**CDR**	**ADD**
GRU	*0.7451*	**0.0102**	**0.8871**	**25.74**	*0.9204*	**0.0092**	**0.9762**	**22.18**
LSTM	**0.7473**	*0.0110*	*0.8801*	*29.67*	**0.9320**	*0.0101*	*0.9705*	*24.89*
Simple RNN	0.7285	0.0135	0.8620	35.42	0.8805	0.0125	0.9176	30.12

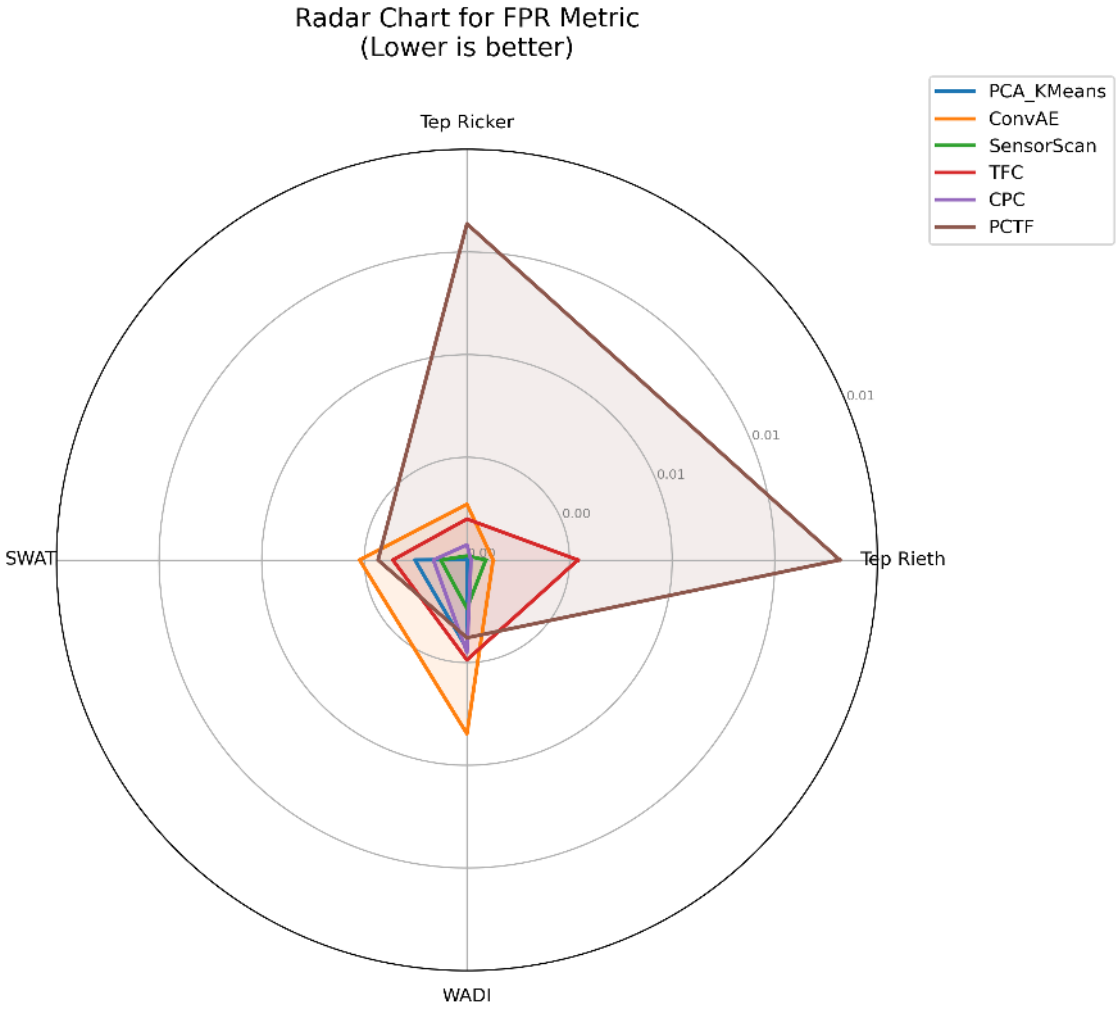

Figure 6.6: Illustration of the FPR results for each dataset and method. Lower is better.

Table 6.3: Test results on Tep Rieth, Tep Ricker, SWAT, and WADI Datasets. Best values are in **bold**, and second-best values are in *italic*.

Metrics	PCA_KMeans	ConvAE	SensorScan	TFC	CPC	PCTF
Tep Rieth Dataset						
TPR	0.3319	0.3907	*0.6910*	0.6921	0.7245	**0.7451**
FPR	**0.0001**	0.0008	0.0006	0.0031	*0.0002*	0.0102
CDR	0.7487	0.3448	*0.8513*	0.7186	0.6946	**0.8871**
ADD	110.24	160.72	80.82	75.17	*43.67*	**25.74**
AUC	0.7051	0.7197	*0.9054*	0.8645	0.9075	**0.9291**
Tep Ricker Dataset						
TPR	0.3581	0.6023	*0.9013*	0.7861	0.7140	**0.9204**
FPR	**0.0001**	0.0016	*0.0002*	0.0012	0.0005	0.0092
CDR	*0.9562*	0.9402	0.7219	0.8465	0.7192	**0.9762**
ADD	113.33	155.16	*30.98*	97.18	42.30	**22.18**
AUC	0.7506	0.8455	**0.9820**	0.9365	0.8947	*0.9805*
SWAT Dataset						
TPR	0.3850	0.6105	*0.8802*	0.8056	0.7812	**0.9223**
FPR	0.0015	0.0030	**0.0008**	0.0021	*0.0010*	0.0025
CDR	*0.8802*	0.8457	0.7508	0.8489	0.7964	**0.9401**
ADD	113.24	152.13	*30.50*	90.78	43.12	**25.20**
AUC	0.6770	0.8445	*0.9639*	0.9131	0.8907	**0.9822**
WADI Dataset						
TPR	0.3624	0.5501	*0.8104*	0.7623	0.7401	**0.8650**
FPR	0.0025	0.0048	**0.0014**	0.0028	0.0026	*0.0022*
CDR	*0.8401*	0.8102	0.7105	0.7809	0.7553	**0.9152**
ADD	127.65	175.40	*58.12*	110.23	60.47	**35.80**
AUC	0.7506	0.7725	*0.9291*	0.9058	0.8820	**0.9594**

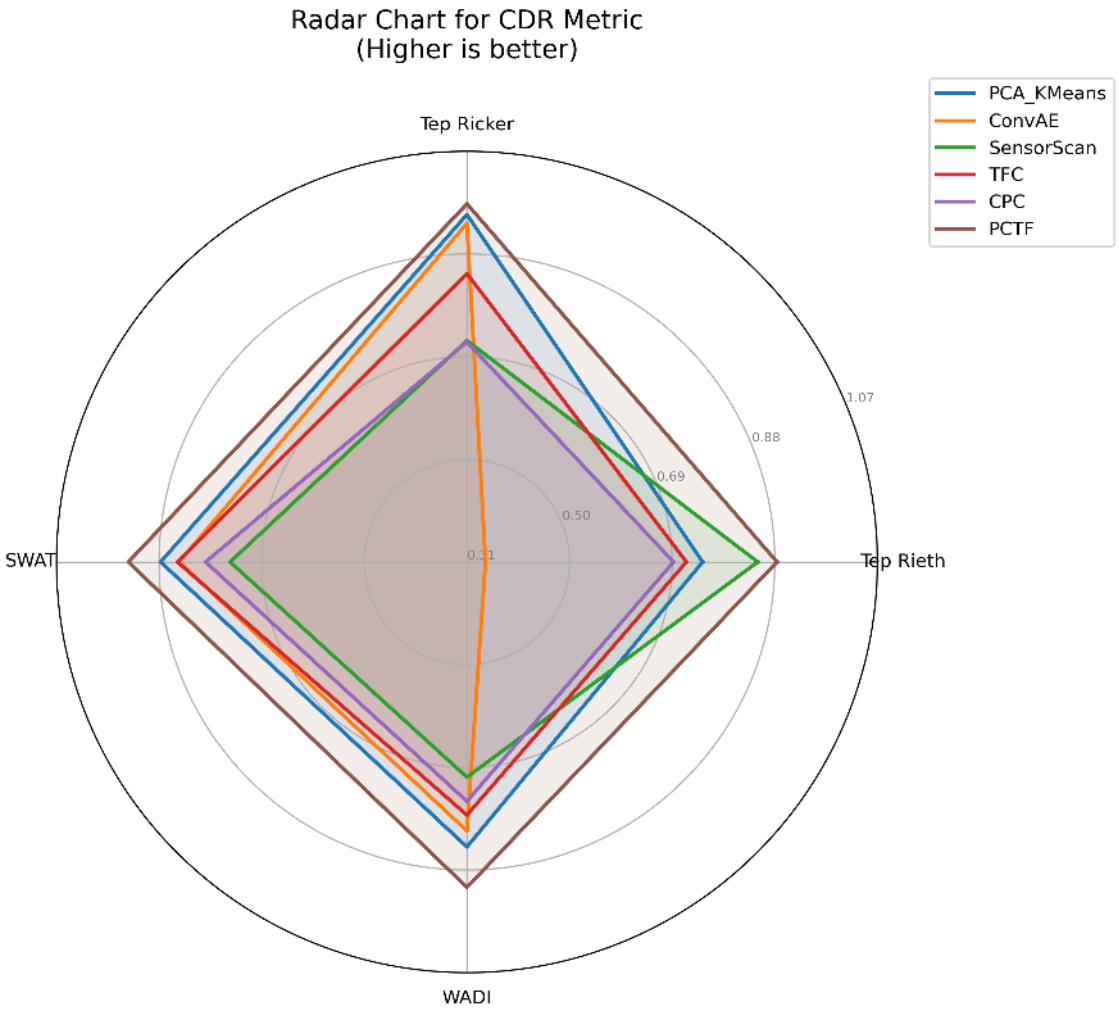

Figure 6.7: Illustration of the CDR results for each dataset and method. Higher is better.

Table 6.4: Comparison of latencies for Cloud and Edge Computing Scenarios

# of Plants	Time 10^3		Time 10^4		Time 10^5	
	Cloud	**Edge**	**Cloud**	**Edge**	**Cloud**	**Edge**
2	7.26	4.96	601.41	4.96	597.79	4.96
4	14.52	4.97	605.04	4.96	1195.57	4.96
8	16.95	4.96	794.62	4.96	1572.28	4.96
16	9.19	4.96	895.80	4.96	1782.40	4.96
32	10.88	4.94	1259.14	5.52	3134.24	4.96

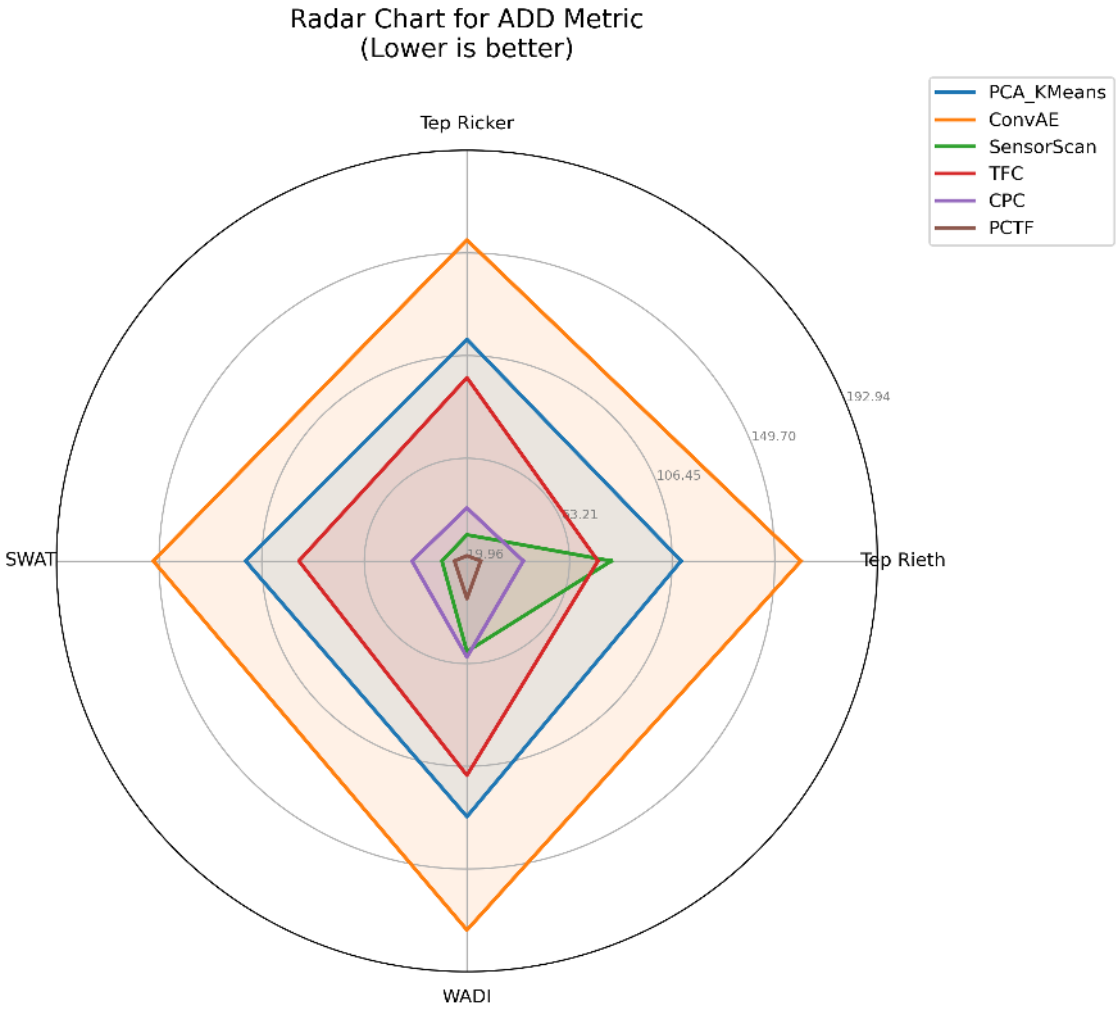

Figure 6.8: Illustration of the ADD results for each dataset and method. Lower is better.

Chapter 7

OPEN ISSUES & FUTURE DIRECTIONS AND CONCLUSION

This final chapter discusses ongoing challenges and possible improvements, like making the model more interpretable and adaptable to different environments. The chapter concludes by summarizing the contributions and impact of the research, emphasizing its potential to improve fault detection in industrial systems.

7.1 Open Issues & Future Directions

As our SSL-based FD research advances, a number of exciting directions for further investigation become apparent. In order to progress the subject of industrial FD, we highlight important areas in this section that demand more research.

7.1.1 Enhancing Model Interpretability

The interpretability of complex models, particularly deep learning-based approaches, remains a significant challenge in industrial FD. Improving model interpretability is crucial for gaining trust and facilitating corrective actions in industrial settings. Future efforts should focus on developing visualization techniques for time-frequency representations learned by the model and investigating attention mechanisms to highlight the most relevant features or time steps for fault detection. Post-hoc explanation methods tailored for time series data and industrial processes could further enhance transparency. Moreover, integrating domain knowledge into the model can support the creation of more interpretable intermediate representations.

7.1.2 Multimodal and Heterogeneous Data Integration

The integration of various data sources is essential for effective FD. Future research should emphasize the development of SSL architectures that can seamlessly incorporate multiple data modalities, such as time series, images, and textual data. Transfer learning techniques could enable the reuse of pre-trained models across different industrial processes and data types, promoting better generalization. Additionally, novel fusion strategies are required to handle data with varying sampling rates and reliability, while the challenges of data alignment and synchronization in multimodal industrial datasets must be addressed.

7.1.3 Edge-Optimized Architectures

Our analysis in Section 6.6 demonstrates the significant advantages of edge deployment for industrial FD applications. To fully leverage these benefits, future research should prioritize the development of lightweight versions of SSL models optimized for edge devices. Model compression techniques, such as pruning and quantization, are essential for reducing computational and memory requirements while maintaining performance. Federated learning approaches may enable collaborative learning across multiple edge devices while preserving data privacy. Furthermore, adaptive architectures capable of dynamically adjusting their complexity based on available computational resources and current operational demands would significantly enhance the efficiency of edge-based fault detection systems.

7.1.4 Addressing Domain Generalization

Evaluating the efficiency of different studies in the field of fault prediction usually entails thorough testing across a range of operational domains. These evaluations are essential for guaranteeing the prediction models' generalizability and robustness. The domain generalization issue is addressed in a noteworthy work by [Huang et al., 2023], which incorporates a novel Causal Layer into the deep learning model architecture. This layer is specifically designed to identify and encode the invariant causal relationships governing feature generation. The strategy aims to improve the

model's robustness and transferability by minimizing the impact of domain-specific variables while simultaneously maximizing fault prediction accuracy. Despite being novel, the study's methodology has weaknesses. Practical use may be limited by the complexity added by the Causal Layer, which includes the calibration of extra hyperparameters and the assumption of cross-domain causality invariance. These challenges represent crucial avenues for future research, aiming to refine the balance between model complexity and generalizability in an ever-changing array of fault-prone environments.

7.2 Conclusion

This thesis introduces the PCTF architecture for FD in industrial processes, leveraging SSL techniques. Our approach addresses critical challenges in industrial FD, including data scarcity, better performance on evaluation metrics, and the need for robust, generalizable models. The PCTF model demonstrates significant advancements over existing methods, as evidenced by its superior performance across multiple TEP datasets, and SWaT and WADI datasets. Notably, our model achieved the highest TPR of 0.9204 and CDR of 0.9762 on the TEP Ricker dataset, surpassing both traditional and state-of-the-art approaches. The same overperforming trends has also seen in other benchmark datasets as well. The integration of time and frequency domain information, coupled with the predictive contrast learning approach, enables our model to capture complex temporal dependencies and patterns crucial for accurate fault detection in dynamic industrial processes. One of the most significant achievements of our model is the substantial reduction in ADD, reaching as low as 22.18 seconds on the TEP Ricker dataset. This improvement is particularly valuable in industrial settings where rapid fault detection can prevent catastrophic failures and minimize downtime. While our model showed a slightly higher FPR compared to some benchmarks, this trade-off is justified by the substantial improvements in other critical metrics, aligning with industrial priorities where the cost of missing a fault typically outweighs that of false alarms. Our comprehensive edge deployment analysis revealed significant advantages for implementing the

PCTF model in edge computing scenarios. The model maintained consistently low latency (approximately 4.96 ms) across various simulation scales, demonstrating excellent scalability and responsiveness. This characteristic is crucial for real-time FD applications in industrial environments. The stark contrast with cloud deployment scenarios, which showed dramatically increasing latencies in complex simulations, underscores the potential of edge computing in enhancing the practicality and efficiency of advanced FD systems. The PCTF architecture addresses limitations observed in other approaches such as CPC and TFC. By removing the explicit consistency loss and leveraging the inherent consistency maintained by the InfoNCE loss, our model achieves a more streamlined and computationally efficient architecture without sacrificing performance. This approach demonstrates the potential of SSL techniques in overcoming data scarcity challenges, a persistent issue in industrial FD applications. Our work contributes significantly to advancing the state-of-the-art in FD methods, offering a robust, data-efficient solution that is well-suited for large-scale, cost-effective implementations in industrial settings. The combination of superior fault detection capabilities with the scalability and low latency of edge deployment presents a promising direction for enhancing FD in complex industrial environments. Future research directions have also been presented that could focus on enhancing model interpretability, and exploring the integration of multimodal data sources, etc. Further optimization for edge computing and addressing domain generalization remain important areas for investigation. These efforts will contribute to the development of more efficient FD systems. To conclude, this thesis presents a significant step forward in the field of industrial FD, offering a novel SSL-based architecture that demonstrates superior generic performance, and analyses scalability and practical applicability with network experiments. The PCTF model, combined with edge computing deployment, paves the way for more efficient, responsive, and cost-effective fault detection systems in industry.

BIBLIOGRAPHY

[wad, 2017] (2017). *CySWATER '17: Proceedings of the 3rd International Workshop on Cyber-Physical Systems for Smart Water Networks*, New York, NY, USA. Association for Computing Machinery.

[Albelwi, 2022] Albelwi, S. (2022). Survey on self-supervised learning: Auxiliary pretext tasks and contrastive learning methods in imaging. *Entropy*, 24(4).

[Biases,] Biases, W. . Weights & biases: Developer tools for machine learning. `https://wandb.ai`. Accessed: 2024-06-30.

[Chen et al., 2020] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations.

[Chen, 2019] Chen, X. (2019). Tennessee eastman simulation dataset.

[Daurenbayeva et al., 2023] Daurenbayeva, N., Nurlanuly, A., Atymtayeva, L., and Mendes, M. (2023). Survey of applications of machine learning for fault detection, diagnosis and prediction in microclimate control systems. *Energies*, 16(8).

[Dong et al., 2018] Dong, L.-F., Gan, Y.-Z., Mao, X.-L., Yang, Y.-B., and Shen, C. (2018). Learning deep representations using convolutional auto-encoders with symmetric skip connections. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3006–3010.

[Downs and Vogel, 1993] Downs, J. J. and Vogel, E. F. (1993). A plant-wide industrial process control problem. *Computers & chemical engineering*, 17(3):245–255.

[e Souza, 2023] e Souza, A.O., d. S. M. . d. S. F. (2023). Enhancing fault detection and diagnosis systems for a chemical process: a study on convolutional neural networks and transfer learning. *Evolving Systems*.

[Elhefnawy et al., 2023] Elhefnawy, M., Ouali, M.-S., Ragab, A., and Amazouz, M. (2023). Fusion of heterogeneous industrial data using polygon generation deep learning. *Results in Engineering*, 19:101234.

[Ferdowsi et al., 2022] Ferdowsi, H., Cai, J., and Jagannathan, S. (2022). Actuator and sensor fault detection and failure prediction for systems with multi-dimensional nonlinear partial differential equations. *International Journal of Control, Automation and Systems*, 20:789–802.

[Gansbeke et al., 2020] Gansbeke, W. V., Vandenhende, S., Georgoulis, S., Proesmans, M., and Gool, L. V. (2020). Scan: Learning to classify images without labels.

[Gill et al., 2024] Gill, W., Görgülü, H., Özkasap, Ö., Gürsoy, A., and Serfidan, A. C. (2024). Grasp: Graph-based alarm management and supervision with prediction for scada systems.

[Golyadkin et al., 2023] Golyadkin, M., Pozdnyakov, V., Zhukov, L., and Makarov, I. (2023). Sensorscan: Self-supervised learning and deep clustering for fault diagnosis in chemical processes. *Artificial Intelligence*, 324:104012.

[Görgülü and Özkasap, 2023] Görgülü, H. and Özkasap, Ö. (2023). Machine learning methods for alarm prediction in industrial informatics: Review and benchmark. In Jove, E., Zayas-Gato, F., Michelena, Á., and Calvo-Rolle, J. L., editors, *Distributed Computing and Artificial Intelligence, Special Sessions II - Intelligent Systems Applications, 20th International Conference*, pages 21–30, Cham. Springer Nature Switzerland.

[Görgülü and Özkasap, 2024] Görgülü, H. and Özkasap, Ö. (2024). Ai-powered industrial fault detection and diagnosis with self-supervised prediction contrast time frequency.

[Guo et al., 2023] Guo, X., Cui, X., Cheng, C., and Lu, L. (2023). Transformer-based high-precision chemical process fault detection. In *2023 2nd International*

Conference on Robotics, Artificial Intelligence and Intelligent Control (RAIIC), pages 236–240.

[He et al., 2005] He, Q. P., Qin, S. J., and Wang, J. (2005). A new fault diagnosis method using fault directions in fisher discriminant analysis. *AIChE Journal*, 51(2):555–571.

[Huang et al., 2023] Huang, H., Wang, R., Zhou, K., Ning, L., and Song, K. (2023). Causalvit: Domain generalization for chemical engineering process fault detection and diagnosis. *Process Safety and Environmental Protection*, 176:155–165.

[Isermann and Balle, 1997] Isermann, R. and Balle, P. (1997). Trends in the application of model-based fault detection and diagnosis of technical processes. *Control engineering practice*, 5(5):709–719.

[Lera et al., 2019] Lera, I., Guerrero, C., and Juiz, C. (2019). YAFS: A simulator for iot scenarios in fog computing. *CoRR*, abs/1902.01091.

[Lomov et al., 2021] Lomov, I., Lyubimov, M., Makarov, I., and Zhukov, L. E. (2021). Fault detection in tennessee eastman process with temporal deep learning models. *Journal of Industrial Information Integration*, 23:100216.

[Marino et al., 2021] Marino, R., Wisultschew, C., Otero, A., Lanza-Gutierrez, J. M., Portilla, J., and Torre, E. d. l. (2021). A machine-learning-based distributed system for fault diagnosis with scalable detection quality in industrial iot. *IEEE Internet of Things Journal*, 8(6):4339–4352.

[Mathur and Tippenhauer, 2016] Mathur, A. P. and Tippenhauer, N. O. (2016). Swat: a water treatment testbed for research and training on ics security. In *2016 International Workshop on Cyber-physical Systems for Smart Water Networks (CySWater)*, pages 31–36.

[Name, 2022] Name, A. (2022). Fault classification in the process industry using polygon generation and deep learning. *Intelligent Manufacturing*, 1531(1544):33.

[Neupane and Seok, 2020] Neupane, D. and Seok, J. (2020). Bearing fault detection and diagnosis using case western reserve university dataset with deep learning approaches: A review. *IEEE Access*, 8:93155–93178.

[Park et al., 2020] Park, Y.-J., Fan, S.-K. S., and Hsu, C.-Y. (2020). A review on fault detection and process diagnostics in industrial processes. *Processes*, 8(9).

[Reinartz et al., 2021] Reinartz, C., Kulahci, M., and Ravn, O. (2021). An extended tennessee eastman simulation dataset for fault-detection and decision support systems. *Computers Chemical Engineering*, 149:107281.

[Rieth et al., 2018] Rieth, C. A., Amsel, B. D., Tran, R., and Cook, M. B. (2018). Issues and advances in anomaly detection evaluation for joint human-automated systems. In Chen, J., editor, *Advances in Human Factors in Robots and Unmanned Systems*, pages 52–63, Cham. Springer International Publishing.

[Rombach et al., 2021] Rombach, K., Michau, G., and Fink, O. (2021). Contrastive learning for fault detection and diagnostics in the context of changing operating conditions and novel fault types. *Sensors*, 21(10).

[Stief et al., 2018] Stief, A., Ottewill, J. R., Tan, R., and Cao, Y. (2018). Process and alarm data integration under a two-stage bayesian framework for fault diagnostics. *IFAC-PapersOnLine*, 51(24):1220–1226. 10th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes SAFEPROCESS 2018.

[Stief et al., 2019] Stief, A., Tan, R., Cao, Y., Ottewill, J. R., Thornhill, N. F., and Baranowski, J. (2019). A heterogeneous benchmark dataset for data analytics: Multiphase flow facility case study. *Journal of Process Control*, 79:41–55.

[Tian, 2023a] Tian, Y., W. Y. P. X. e. a. (2023a). A fault diagnosis method for few-shot industrial processes based on semantic segmentation and hybrid domain transfer learning. *IEEE Transactions on Industrial Informatics*, 53(7):28268–28290.

[Tian, 2023b] Tian, Y. (2023b). Fault diagnosis strategy of industrial process based on multi-source heterogeneous information and deep learning. *Chemical Engineering Research and Design*, 198:459–477.

[Tian et al., 2023] Tian, Y., Li, J., Song, Q., Li, Z., and Huang, X. (2023). Pyramid reconstruction assisted deep autoencoding gaussian mixture model for industrial fault detection. *Information Sciences*, 649:119682.

[van den Oord et al., 2019] van den Oord, A., Li, Y., and Vinyals, O. (2019). Representation learning with contrastive predictive coding.

[Wang et al., 2023] Wang, K., Yan, C., Mo, Y., Yuan, X., Wang, Y., and Yang, C. (2023). Neuron-compressed deep neural network and its application in industrial anomaly detection. *IEEE Transactions on Industrial Informatics*, 19(7):7914–7924.

[Wang et al., 2020] Wang, Y., Yan, J., Sun, Q., Jiang, Q., and Zhou, Y. (2020). Bearing intelligent fault diagnosis in the industrial internet of things context: A lightweight convolutional neural network. *IEEE Access*, 8:87329–87340.

[Yu et al., 2023] Yu, W., Liu, Y., Dillon, T., and Rahayu, W. (2023). Edge computing-assisted iot framework with an autoencoder for fault detection in manufacturing predictive maintenance. *IEEE Transactions on Industrial Informatics*, 19(4):5701–5710.

[Zhang et al., 2024a] Zhang, K., Wen, Q., Zhang, C., Cai, R., Jin, M., Liu, Y., Zhang, J., Liang, Y., Pang, G., Song, D., and Pan, S. (2024a). Self-supervised learning for time series analysis: Taxonomy, progress, and prospects.

[Zhang et al., 2024b] Zhang, X., Zhao, Z., Tsiligkaridis, T., and Zitnik, M. (2024b). Self-supervised contrastive pre-training for time series via time-frequency consistency. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

[Zhu et al., 2023] Zhu, Z., Lei, Y., Qi, G., Chai, Y., Mazur, N., An, Y., and Huang, X. (2023). A review of the application of deep learning in intelligent fault diagnosis of rotating machinery. *Measurement*, 206:112346.