Circuits and Systems
Mekelweg 4,
2628 CD Delft
The Netherlands
https://sps.ewi.tudelft.nl/

SPS-2025-00

M.Sc. Thesis

Machine learning for lifetime prediction of electronic devices

Zhixuan Ge B.Sc.

Abstract

Gallium Nitride High Electron Mobility Transistors (GaN HEMTs) are promising devices for next-generation power electronic systems due to their high efficiency, high power density, and broad applicability in areas including electric vehicles, renewable energy, and communication. Existing studies on Prognostic and Health Management (PHM) of GaN HEMTs focus on the statistical behaviors of multiple devices under specific circumstances, which means individual device variability is neglected in these approaches. This work addresses the gap in individual health status by applying deep learning to achieve the Remaining Useful Lifetime (RUL) prediction of individual p-type GaN HEMT devices under diverse working conditions. In the proposed method, a Temporal Convolutional Network (TCN) integrated with attention mechanisms is developed to extract informative features and emphasize critical features within the measurements. To handle the varying lifetimes of p-GAN HEMT devices tested under different temperatures and stress voltages, we propose a prediction pipeline, which estimates the relative RUL in percentage and then converts it into absolute RUL in seconds. The Leave-One-Group-Out (LOGO) Cross-Validation (CV) is applied to ensure the generalization of the proposed method by testing the model on data collected from the unseen environment.

Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Machine learning for lifetime prediction of electronic devices

Thesis

submitted in partial fulfillment of the
requirements for the degree of

Master of Science

in

Electrical Engineering

by

Zhixuan Ge B.Sc.
born in Changzhou, China

This work was performed in:

Circuits and Systems Group
Department of Microelectronics
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Delft University of Technology
Department of
Microelectronics

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics and Computer Science for acceptance a thesis entitled **"Machine learning for lifetime prediction of electronic devices"** by **Zhixuan Ge B.Sc.** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: 26 September 2025

Chairman:

prof.dr.ir. Justin Dauwels

Advisor:

ir. Shuoyan Zhao

Committee Members:

prof.dr. GuoQi Zhang

Abstract

Gallium Nitride High Electron Mobility Transistors (GaN HEMTs) are promising devices for next-generation power electronic systems due to their high efficiency, high power density, and broad applicability in areas including electric vehicles, renewable energy, and communication. Existing studies on Prognostic and Health Management (PHM) of GaN HEMTs focus on the statistical behaviors of multiple devices under specific circumstances, which means individual device variability is neglected in these approaches. This work addresses the gap in individual health status by applying deep learning to achieve the Remaining Useful Lifetime (RUL) prediction of individual p-type GaN HEMT devices under diverse working conditions. In the proposed method, a Temporal Convolutional Network (TCN) integrated with attention mechanisms is developed to extract informative features and emphasize critical features within the measurements. To handle the varying lifetimes of p-GAN HEMT devices tested under different temperatures and stress voltages, we propose a prediction pipeline, which estimates the relative RUL in percentage and then converts it into absolute RUL in seconds. The Leave-One-Group-Out (LOGO) Cross-Validation (CV) is applied to ensure the generalization of the proposed method by testing the model on data collected from the unseen environment.

Acknowledgments

First and foremost, I want to express my gratitude to my supervisor prof.dr.ir. Justin Dauwels and my advisor ir. Shuoyan Zhao for their invaluable guidance, constant support, and insightful feedback throughout the course of this thesis. Whenever I have any questions, they are always responsive and consistently offer valuable suggestions, which significantly improved the quality of my thesis. Their expertise and encouragement have been instrumental in shaping the direction of my work.

I am also thankful to my colleagues, Jiarui and Joris, for their helpful suggestions and constructive discussions, which contributed to the progress of this thesis.

I sincerely thank prof.dr. GuoQi Zhang for taking the time to join my thesis committee and providing thoughtful suggestions.

Beyond academia, I would like to express my heartfelt gratitude to my friends, especially Tim and Kevin, for their support and encouragement. I sincerely hope that our bond will remain strong and lasting in the years to come.

I am also deeply thankful to my family for their unconditional love, patience, and encouragement. Their support has been a constant source of strength, allowing me to stay grounded and focused even during the most challenging times.

This work is supported by R-PODID, which is funded by the Chips Joint Undertaking and its members, including the top-up funding by National Authorities of Italy, Turkey, Portugal, The Netherlands, Czech Republic, Latvia, Greece, and Romania under grant agreement n° 101112338.

Zhixuan Ge B.Sc.
Delft, The Netherlands
26 September 2025

Contents

List of Figures

List of Tables

Introduction 1

In recent years, the rapid advancement of electric power conversion systems, driven by applications such as electric vehicles [1], renewable energy [2], and communication [3], has imposed increasing demands on the capability, efficiency, and reliability of devices. Due to their wide band gap characteristics, Gallium Nitride High Electron Mobility Transistors (GaN HEMTs) [2] have advantages in switching speed, conduction loss, and thermal performance, making them a promising replacement for traditional silicon-based power devices and a potential next-generation power electronic device [2, 4, 5].

Despite their advantages, the long-term reliability of GaN HEMTs remains a critical concern, especially in critical environments. Performance degradation over time can lead to device breakdown and failure, posing risks to the safety of the overall system. Therefore, accurate prediction of the lifetime of GaN HEMTs is essential for device maintenance and system robustness [2].

Most existing research on lifetime prediction for GaN HEMTs focuses on modeling the survival probability and degradation behavior of a group of devices under certain conditions [4, 6]. Attention is given to the statistical characteristics of a group of HEMTs, rather than the performance of an individual HEMT. While these approaches are valuable for understanding general failure trends, individual variability in device operation under real-world conditions is often overlooked. Such population-level models may not fully exploit real-time measurements to improve prediction accuracy by analyzing the health status of specific devices deployed in practice.

To address this limitation, we propose a Temporal Convolutional Network (TCN)-based method [7] integrated with an attention mechanism [8, 9] that performs remaining useful lifetime (RUL) [10] prediction based on real-time measured signals collected during device operation. The core work of this thesis lies in the design of a prediction model for the gate lifetime of individual p-type Gallium Nitride High Electron Mobility Transistors (p-GaN HEMT) devices under various working conditions. To achieve this, a general TCN-based deep learning model that can predict percentage remaining lifetimes of p-GaN HEMT devices with significant variations in actual lifetimes is developed. Based on the predicted relative RUL on a percentage scale, the actual lifetime is recovered in the proposed prediction pipeline. To secure adaptability across various environments, a method called Leave-One-Group-Out Cross-Validation (LOGO CV) is employed. In this method, devices are grouped according to their operating conditions, and the trained models infer the RUL of p-GaN HEMT devices under unknown conditions, which are not contained in training. This training approach secures adaptability across environments. By incorporating real-time operational data from new environments, the proposed method enhances prediction accuracy and improves generalization across varying scenarios, thereby increasing the overall robustness of power electronics systems.

1.1 Outline

This thesis is structured as follows:

- **Chapter 2. Problem Statement:** First, this chapter introduces the fundamental concepts of GaN HEMTs and reviews recent developments in RUL prediction, a key branch of prognostics and health management, which has been extensively studied in various fields. Subsequently, the chapter identifies the limitations of existing approaches to lifetime prediction of GaN HEMTs and presents the major problem and the corresponding objective of this study.
- **Chapter 3. Methods:** This chapter presents the detailed pipeline of the proposed RUL prediction model for GaN HEMTs. First, time-domain features are manually extracted from the raw measurement data and subsequently selected to remove redundant and irrelevant information. The selected features are then fed into the proposed TCN model with attention mechanisms to estimate the relative RUL. Finally, by combining the predicted relative RUL with the device's actual operating time, the model estimates the absolute RUL and the expected lifetime of the device.
- **Chapter 4. Results and Discussion:** This chapter provides a detailed description of the experimental procedure, results, and corresponding discussion. The experimental procedure includes sample selection, model parameter settings, and the configuration of baseline comparison groups. Results are presented for both relative and absolute RUL, and further incorporate uncertainty prediction to enhance the reliability of the proposed approach. Analysis of the results confirms the effectiveness of the proposed attention-augmented TCN model and demonstrates that the overall workflow successfully achieves RUL prediction for GaN HEMTs.
- **Chapter 5. Conclusion and Future Work:** This chapter summarizes the key contributions and limitations of this thesis, and outlines potential directions for future work and possible improvements.

Problem Statement 2

2.1 Background

2.1.1 Gallium Nitride High Electron Mobility Transistor

Gallium Nitride High Electron Mobility Transistor (GaN HEMT) is a type of field-effect transistor. It is typically built on a heterostructure using Gallium Nitride (GaN) and Aluminum Gallium Nitride (AlGaN). This structure forms a two-dimensional electron gas (2DEG) channel at the interface. The 2DEG has high electron mobility and high carrier density. Compared to traditional silicon-based devices, GaN HEMTs offer higher breakdown voltage, faster switching speed, and lower on-resistance. These features make them suitable for high-power and high-frequency applications [2, 4, 5].

The growing demand for renewable energy has driven the need for advanced power conversion technologies, enabling efficient control and integration of systems in recent years. To support this trend, the development of next-generation semiconductor switching devices has become essential, with goals including higher switching frequencies, reduced switching losses, improved thermal performance, smaller form factors, and higher power density [2].

Although GaN HEMTs perform well in harsh environments, they are prone to performance degradation over time, as all power devices [2]. To ensure the long-term reliability of device deployments with GaN HEMTs, health monitoring is essential. However, most existing studies in this area focus on modeling the degradation or survival probability of a group of GaN HEMTs under specific and consistent stress conditions. For instance, Hua [4] and Tallarico [6] used the Weibull distribution to model the breakdown time, while Moens [5] assumed the time follows a Lognormal distribution.

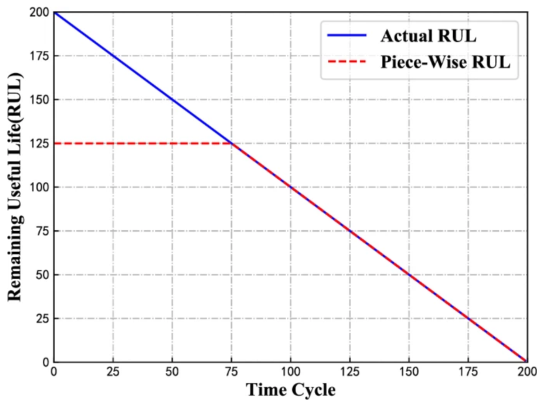
These approaches mainly capture statistical trends across populations, without accounting for individual device variability. This limitation motivates the use of individual lifetime prediction based on real-time measurements of the individual device. The integration of real-time prediction enables lifetime estimation to be adaptively refined in response to device-specific characteristics and operational changes encountered during service.

2.1.2 Remaining Useful Lifetime Prediction

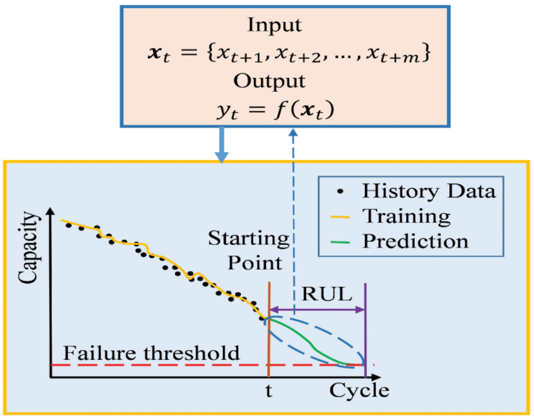
Prognostic and Health Management (PHM) plays a vital role in maintenance strategies by allowing condition monitoring without the need for frequent manual inspections. Traditional health checks often require machine shutdowns and may even cause damage to certain components, increasing operational costs. In contrast, the health status of machinery can be inferred from trends observed in continuously monitored operational data. Based on these measurements, the Remaining Useful Lifetime (RUL) of the equipment can be estimated, which in turn enhances system reliability by allowing for timely and accurate maintenance decisions [10].

RUL prediction can be formulated as a regression problem, and there are two major approaches, direct prediction and indirect prediction, to solve it. Direct prediction creates a projection from input features to the RUL value to estimate the remaining useful lifetime of the machinery directly. As shown in Fig. 2.1(a), the actual RUL is taken as the label of a specific time. As time increases, the RUL decreases linearly correspondingly. However, considering that the degradation pattern is often not observable at the early stage of device operation and the assumption that tools remain stable in the beginning, some studies [10] impose an upper bound on the RUL. For instance, a piecewise RUL curve is used to cap all RUL values above 125, which is shown in the red dashed curve in Figure 2.1(a).

Another strategy is indirect prediction, which constructs a health indicator (HI) through regression first. In this method, the RUL is estimated by performing a multi-step prediction to forecast when this HI will reach a predefined threshold. As shown in Figure 2.1(b), this approach is applied to lithium-ion batteries, where capacity serves as the HI. The model is trained based on data collected up to time t, and then it iteratively predicts future capacity values until the HI falls below the threshold, at which point the RUL is inferred.



(a) Two kinds of labels used in direct RUL prediction. Actual RUL label assumes the degradation is linear, while Piece-Wise RUL label assumes machines are stable in the beginning and linear degradation occurs only after a period of operation [11].



(b) Indirect RUL prediction, in which a non-linear HI is predicted as the target. The lifetime of the machine is determined by the time when HI reaches a specific threshold [12].

Figure 2.1: Two representative approaches for RUL prediction, direct RUL prediction and indirect RUL prediction

Based on the modeling approach employed, RUL prediction methods are generally classified into three categories: model-based methods, data-driven methods, and hybrid methods. As shown in Figure 2.2, these methods differ in underlying principles and applicable scenarios, and have been extensively researched in various fields in the literature [10].

Model-based methods rely on physical models to describe the degradation process

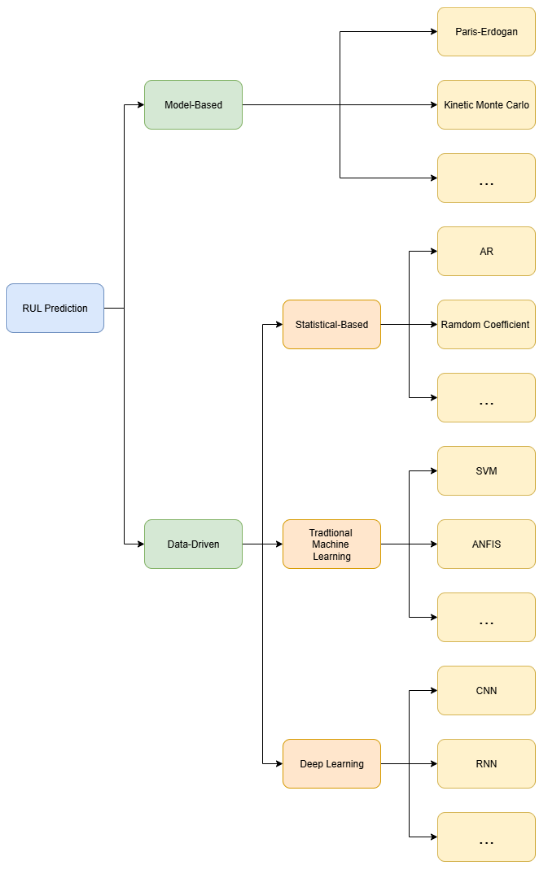


Figure 2.2: Classification of RUL prediction approaches.

of equipment. These approaches typically require expert knowledge to construct mathematical models that capture the internal physical mechanisms of the system, such as the interactions among thermal, electrical, and mechanical factors. For instance, Wang et al. [13] developed an empirical model formulated as a set of ordinary or partial differential equations on the basis of the Paris-Erdogan (PE) model. Röder et al. [14, 15] integrated a microscopic single-particle electrode model with the kinetic Monte Carlo method. In another research, Duong and Raghavan [16] addressed the issues of sample degeneracy and impoverishment in particle filtering by combining it with the Heuristic Kalman algorithm. Cui et al. [17] proposed a prediction architecture based on a time-varying Kalman filter that dynamically selects appropriate models to adapt to different operational stages. While model-based methods offer strong interpretability, their effectiveness heavily depends on the accuracy of the assumed models and prior knowledge, which limits their applicability to complex systems or devices lacking detailed structural information. Owing to the insufficient understanding of the mechanisms of GaN HEMTs, model-based methods are not used in this thesis for RUL estimation.

In data-driven methods, models are constructed by fitting coefficients using abundant data, thereby reducing the reliance on detailed knowledge of device mechanisms. Given the limited understanding of GaN HEMTs, data-driven methods are a better choice for decaying analysis of specific devices. Data-driven methods are further categorized based on the complexity of the modeling techniques, as illustrated in Figure 2.2, into three main subgroups: statistics-based methods, traditional machine learning techniques, and deep learning models.

Statistics-based RUL prediction approaches focus on modeling the probabilistic relationship between variables [18]. These methods assume that the relationship between inputs and outcomes can be described by a conditional Probability Density Function (PDF), allowing the RUL to be inferred from observed data. Representative models in this category include Auto-Regressive (AR) models [19], random coefficient models [20], Wiener process models [21], gamma process models [22], inverse Gaussian process models [23], and Markov models [24].

In contrast to statistical-based approaches that typically rely on predefined probabilistic models and distribution assumptions, Machine Learning (ML) methods are more flexible, aiming to directly learn complex mappings between input features and outputs. Compared to the deep learning-based models employed in this thesis, traditional machine learning techniques have shallow architectures and require fewer training data. However, they retain a certain level of interpretability, as they are often grounded in well-established mathematical theories, even in high-dimensional settings. The size of the training data and the architecture limit the representational capacity of conventional machine learning methods. In the field of RUL prediction, applications of traditional ML models include Support Vector Machines (SVM) [25], Extreme Learning Machines (ELM) [26], Adaptive Neuro-Fuzzy Inference Systems (ANFIS) [27], and Gaussian Process Regression (GPR) [28].

Deep Learning (DL) has emerged as a powerful approach in data-driven RUL prediction, owing to its ability to learn hierarchical representations from raw and high-dimensional data automatically. Compared with traditional machine learning methods, which rely on handcrafted features and shallow model structures, DL models can extract complex nonlinear patterns through their deeper structures. Its diverse multi-layer architectures effectively overcome the limitations of traditional ML techniques in handling nonlinear interactions and complex degradation behaviors. Well-established deep learning models have been widely applied to RUL prediction, including Convolutional Neural Networks (CNN) [29], Recurrent Neural Networks (RNN) [30, 31], Autoencoders [32], and Deep Belief Networks (DBN) [33].

Hybrid methods aim to combine the strengths of model-based and data-driven approaches by incorporating physical knowledge into data-driven models. For instance, Ordóñez et al. [34] developed a hybrid model that incorporated SVM with statistics-based methods by connecting an auto-regressive integrated moving average (ARIMA) model with an SVM. However, similar to model-based methods, they require sufficient understanding of the system's operating mechanisms, which is currently lacking for the degradation of GaN HEMTs. Therefore, hybrid approaches are not adopted in this thesis.

As the measurement data consist of long temporal sequences, deep learning models are a perfect choice for degradation modeling. Therefore, A deep learning based model is adopted as the core of the proposed prediction framework. Specifically, in the relative RUL prediction stage, we propose a model based on TCN, an enhanced variant of CNN, and incorporate an attention mechanism into the architecture. The TCN structure improves the model's robustness of inference by securing causality, while the attention mechanism enhances the model's representational capability by assigning higher weights to key features and time steps, and enables it to capture long-range dependencies.

2.2 Problem Formulation

Building upon the widely applied field of RUL prediction, this thesis extends its application to the dynamic lifetime estimation of individual p-GaN HEMT devices. By incorporating device-specific features observed during operation, the proposed model is capable of learning the unique degradation behaviors of different transistors. This enables a more adaptive approach compared to statistical models.

This thesis specifically addresses the major problem:

> Given a sequence of measured signals from a single p-GaN HEMT device from the start of operation to the breakdown of the gate, as well as the temperature and the stress voltage, how can we estimate the remaining lifetime?

We use the direct prediction in Figure 2.1(a) to predict RUL. Formally, let $\mathbf{F} \in \mathbb{R}^{D_\mathrm{F} \times L}$ denote the dynamic measurement feature matrix consisting of D_F features over L time points, and $\mathbf{s} \in \mathbb{R}^{D_\mathrm{s}}$ denote the static environmental feature vector consisting of D_s features. The model aims to learn a mapping:

$$\hat{\mathbf{y}} = \mathcal{M}(\mathbf{F}, \mathbf{s}), \tag{2.1}$$

in which $\hat{\mathbf{y}} \in \mathbb{R}^L$ is the predicted RUL, and $\mathcal{M}$ denotes the learned prediction model that dynamically updates with the newly observed measurements over time.

To solve this problem, this thesis sets out the following research objectives:

- Develop a dynamic prediction pipeline that separates the process into relative RUL estimation and absolute RUL reconstruction, effectively addressing the variability in device lifetime ranges.
- Develop a deep learning model based on TCNs for predicting the relative RUL of p-GaN HEMTs in percentage from dynamic measurements and static environments.
- Enhance temporal feature and channel representation by integrating attention mechanisms, allowing the model to focus on informative channels and time positions during degradation.
- Design a post-processing method to convert the predicted relative RUL into absolute RUL, expressed in seconds, thereby enabling informative quantified seconds before the breakdown.
- Incorporate probabilistic modeling to quantify prediction uncertainty.
- Apply a leave-one-group-out evaluation scheme to improve the model's generalization under unknown environments.

In the following chapters, the proposed methods for achieving these objectives are described in detail, and their effectiveness is evaluated through experiments.

Methods 3

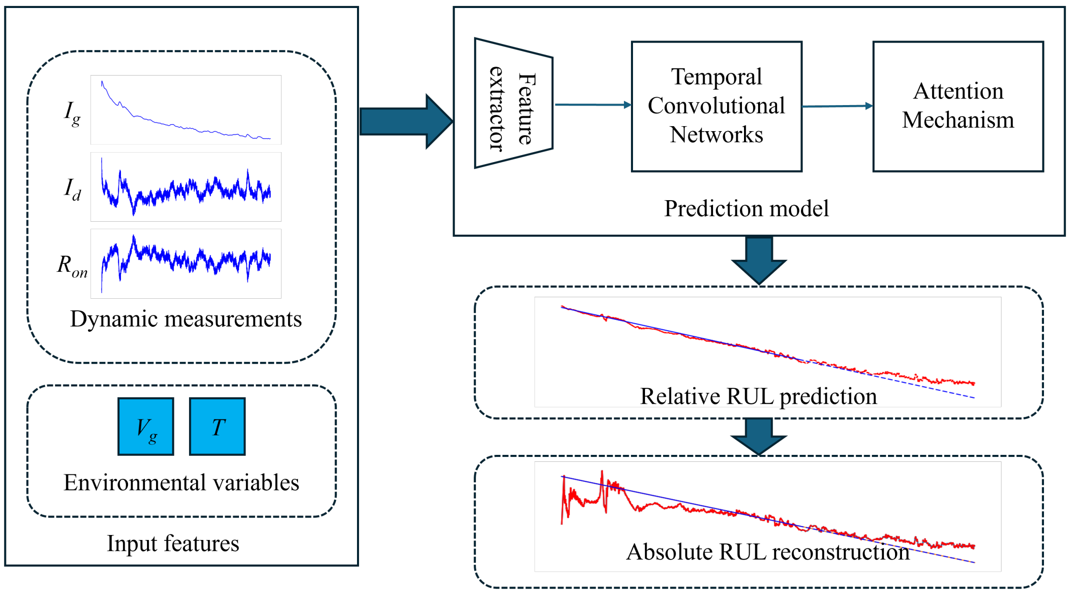


Figure 3.1: Pipeline of dynamic RUL prediction model.

As discussed in chapter 2, most existing studies on GaN HEMTs focus on the overall survival rate of device populations under constant stress conditions [4, 5, 6]. However, with complex and varying manufacturing processes and working conditions, the degradation pattern can be different, which makes it challenging to track individual devices. In contrast, our work shifts the focus from population-level statistics to individual-level behavior. The aim of this thesis is to monitor the degradation of each device separately by introducing RUL prediction into the p-GaN HEMTs domain. This technique allows us to perform dynamic deep learning-based lifetime estimation during operation.

Figure 3.1 illustrates the complete pipeline of our proposed approach. The proposed framework consists of three main steps:

- **Step 1 Feature extraction:** In this step, handcrafted feature extraction and selection are applied to raw measurement data. This step aims to mine informative patterns embedded in the original signals and convert the data into a structured format suitable for processing by the subsequent dynamic prediction model.
- **Step 2 Relative RUL prediction:** This is the step where continuous dynamic prediction takes place. The proposed model processes the preprocessed and sliced

features to estimate the device's relative RUL at each time step based on a Deep Learning model, which provides PHM information by estimating the percentage of lifetime left.

- **Step 3 Absolute RUL reconstruction:** Following Step 2, the proposed model reconstructs the absolute RUL and the estimated total lifetime of the device by combining the predicted results with the elapsed operational time, enabling a more concrete assessment of the device's health status.

Before introducing details of the steps, the basic notation is defined. The degradation process of a p-GaN HEMT device is assumed to be linear. For a device with a total lifetime of L, its remaining useful lifetime decreases continuously from the beginning of operation. Let $y(t)$ denote the RUL at elapsed time t, and L denotes the total lifetime of the device. The values y, t, and L are expressed in seconds. The degradation can thus be formulated as a function of t:

$$y(t) = -t + L, 0 \leq t \leq L, \tag{3.1}$$

which is shown in Figure 3.2(a). As y and t have the same unit, Equation 3.1 exhibits a constant slope of -1. The model is parameterized by a single parameter L. In practice, Equation 3.1 can be discretized and vectorized as

$$\mathbf{y} = \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(t) \\ \vdots \\ y(L) \end{bmatrix} = \begin{bmatrix} L-1 \\ L-2 \\ \vdots \\ L-t \\ \vdots \\ 0 \end{bmatrix}, \quad t = 1, 2, \ldots, L. \tag{3.2}$$

Due to various working conditions of devices, the lifetime of the test devices ranges from a few seconds to dozens of hours. Therefore, directly using absolute RUL values can lead to loss functions being dominated by devices with long lifetimes. This may cause the model to ignore or underestimate the performance on short-lifetime samples, eventually impairing its ability to generalize across operating conditions and capture consistent degradation patterns. Furthermore, excessively large labels may result in unstable gradients during training, potentially causing divergence in the learning process.

To address this inconsistency, absolute RUL, y, is normalized by each device's total lifetime, converting the target into a percentage of the RUL y^*:

$$y^*(t) = \frac{y(t)}{L}, 0 \leq y \leq L. \tag{3.3}$$

Therefore, y^* lies within the interval $[0, 1]$ and Equation 3.1 can be reformulated as

$$y^*(t) = -\frac{1}{L}t + 1, 0 \leq t \leq L, \tag{3.4}$$

which is shown in Figure 3.2(b). Compared with Equation 3.1, Equation 3.4 has a fixed bias of 1 and a slope that depends on the total lifetime L, which means y^* always decays from 1, and the decay speed of y^* is inversely related to L. The corresponding vector form of Equation 3.4 is

$$\mathbf{y}^* = \begin{bmatrix} y(1)^* \\ y(2)^* \\ \vdots \\ y(t)^* \\ \vdots \\ y(L)^* \end{bmatrix} = -\frac{1}{L} \begin{bmatrix} 1 \\ 2 \\ \vdots \\ t \\ \vdots \\ 0 \end{bmatrix} + \mathbf{1}, \quad t = 1, 2, \ldots, L. \tag{3.5}$$

By constraining the range of y^*, normalization not only reduces the scale variance across samples but also encourages the model to learn generic degradation patterns rather than device-specific absolute values. Furthermore, it improves training stability, allows for better loss balancing, and enhances generalization to unseen conditions.

Let $\mathbf{F} \in \mathbb{R}^{D_F \times L}$ denote the original dynamic measurements of a device, where D_F is the number of raw feature channels, and L is the total lifetime, or it can be called the sequence length here. Each device also has a vector of static features $\mathbf{s} \in \mathbb{R}^{D_s}$, in which D_s is the number of environmental variables.

In Step 1, the original time series data $\mathbf{F}$ and the static features $\mathbf{s}$ are preprocessed through preprocessing techniques, including feature extraction and selection, to generate the final input tensor $\mathbf{X}$ for the deep learning model. The details will be discussed in section 3.3

In Step 2, a deep learning model $\mathcal{M}_{\text{rel}}(\cdot)$ takes $\mathbf{X}$ as input and outputs an estimate of the relative remaining useful lifetime, denoted as $\hat{\mathbf{y}}^*$. The process can be formulated as

$$\hat{\mathbf{y}}^* = \mathcal{M}_{\text{rel}}(\mathbf{X}). \tag{3.6}$$

Relative value $\hat{\mathbf{y}}^*$ typically ranges between 0 and 1, where 1 represents the beginning of operation and 0 indicates end-of-life. The architectures of $\mathcal{M}_{\text{rel}}$ will be discussed in section 3.2.

In Step 3, the estimated $\hat{\mathbf{y}}^*$ is used to infer the total lifetime $\hat{\mathbf{L}}$ of the device and absolute remaining useful lifetime $\hat{\mathbf{y}}$ based on the elapsed time t. , which are given by

$$\hat{\mathbf{L}} = [\,\hat{L}(1),\, \hat{L}(2),\, \ldots,\, \hat{L}(t),\, \ldots,\, \hat{L}(L)\,]^\top \tag{3.7}$$

and

$$\hat{\mathbf{y}} = [\,\hat{y}(1),\, \hat{y}(2),\, \ldots,\, \hat{y}(t),\, \ldots,\, \hat{y}(L)\,]^\top. \tag{3.8}$$

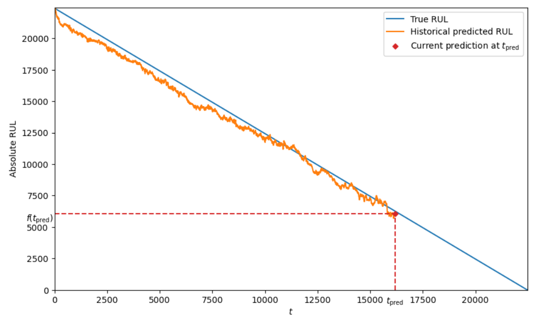
Elements of $\hat{\mathbf{L}}$ and $\hat{\mathbf{y}}$ can be represented as a function of $\hat{y}^*(t)$ and t:

$$\hat{L}(t) = f_L(\hat{y}^*(t), t) \tag{3.9}$$

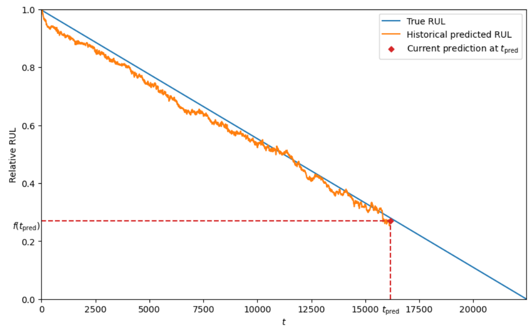
and

$$\hat{y}(t) = f_{\text{abs}}(\hat{y}^*(t), t). \tag{3.10}$$

Step 3 enables the mapping from relative degradation patterns to actual remaining time, providing a quantified and concrete value of lifetime. section 3.3 explains f_L and f_{abs}.



(a) Absolute RUL y, which has a fixed slope -1 and a bias equal to L.



(b) Relative RUL y^*, which has a fixed bias 1 and a slope inversely proportional to $-1/L$.

Figure 3.2: Labels of absolute RUL and relative RUL.

3.1 Feature Engineering

The proposed model not only uses the original feature, but also employs feature extraction and selection to discover hidden information from the measurements. First, manual feature engineering is applied to raw features, and then feature selection is performed. For each device, the measurements can be described as 2 data structures: a dynamic measurement matrix $\mathbf{F} \in \mathbb{R}^{D_F \times L}$ and a static environment vector $\mathbf{s} \in \mathbb{R}^{D_s}$, where D_F and D_s denote the number of dynamic and static variables, respectively. $\mathbf{F}$ can be represented as

$$\mathbf{F} = \begin{bmatrix} \mathbf{f}_1^\top \\ \mathbf{f}_2^\top \\ \vdots \\ \mathbf{f}_{D_F}^\top \end{bmatrix}, \quad \mathbf{f}_n \in \mathbb{R}^L, \quad n = 1, \dots, D_F, \tag{3.11}$$

where each row vector $\mathbf{f}_n$ represents the dynamic sensor readings at time step t, from the beginning of device operation until failure. While the static input is represented as

$$\mathbf{s} = [s_1, s_2, \dots, s_{D_s}], \quad s_i \in \mathbb{R}, \tag{3.12}$$

where each s_i is a constant scalar representing a fixed environment condition during the lifetime of the device.

To extract meaningful representations from the raw time-series data, handcrafted feature engineering is applied to the dynamic measurements $\mathbf{F}$ before features are fed into the dynamic model. Specifically, we choose to extract various time-domain features from the three signals, including amplitude features (maximum, minimum, peak-to-peak, RMS), statistical features (standard deviation, kurtosis, skewness), and waveform shape factors (waveform factor, peak factor, pulse factor, and crest factor). The augmented version $\mathbf{F}^+$ after feature extraction is denoted as

$$\mathbf{F}^+ = \begin{bmatrix} \mathbf{F} \\ \mathbf{F}_{\text{ext}} \end{bmatrix} \in \mathbb{R}^{(D_F + D_{\text{ext}}) \times L}, \tag{3.13}$$

in which $\mathbf{F}_{\text{ext}} \in \mathbb{R}^{D_{\text{ext}} \times L}$ is the extracted features.

3.1.1 Minimum Redundancy Maximum Relevance

Although a wide range of features were initially extracted to capture the underlying degradation patterns, not all features contribute equally to the prediction task. Some of the extracted features may be redundant, carrying overlapping information that can lead to overfitting or degrade model generalization. Others may be sensitive to measurement noise or irrelevant to the RUL, resulting in the introduction of uncertainty and bias to the learning process. Besides, incorporating unnecessary features increases computational complexity. Unnecessary features can lead to a waste of computational resources and slow down training.

Therefore, a feature selection process is applied to retain only the most informative and robust features, ensuring that the model remains efficient, interpretable, and focused on relevant degradation signals. Minimum Redundancy Maximum Relevance (mRMR) algorithm [35] is applied to perform feature selection on the extracted features. mRMR aims to select features that are highly relevant to the target variable while minimizing redundancy among the selected features themselves. Specifically, mRMR evaluates each candidate feature based on two criteria: relevance, which is typically measured by mutual information between the feature and the target, and redundancy, which is measured by mutual information between the candidate and selected features. By balancing these two aspects, mRMR ensures that the selected subset provides diverse informative representations, which can improve both the efficiency and generalization ability of the predictive model.

To formalize the feature selection process, let the i-th extracted feature be denoted as $\mathbf{f}_i^+, i = 1, \dots, D_F + D_{\text{ext}}$. To avoid confusion, the linear degradation target relative RUL is denoted as a vector $\mathbf{y}^*$ in this section. The mutual information $I(\mathbf{f}_i^+, \mathbf{y}^*)$ between $\mathbf{f}_i^+$ and $\mathbf{y}^*$ is computed via entropy estimation from k-nearest neighbors distances [36]. Similarly, entropy estimation is applied to calculate the redundancy $I(\mathbf{f}_i^+, \mathbf{f}_j^+)$, i.e. mutual information between 2 features $\mathbf{f}_i^+$ and $\mathbf{f}_j^+$. Let Ω denote the set consisting of all extracted features, and let S represent the subset of features selected from Ω. In the first iteration of mRMR, S is empty:

$$S = \emptyset, \tag{3.14}$$

which means the algorithm only evaluates relevance. Therefore, the optimization target is

$$\tilde{\mathbf{f}} = \arg\max_{\mathbf{f}_j^+ \in \Omega} I(\mathbf{f}_j^+; \mathbf{y}^*), \tag{3.15}$$

which means selecting the feature with maximum mutual information with the target $\mathbf{y}^*$. The selected feature is added to S. In the remaining iterations, S is not empty anymore. The algorithm tries to find a feature not yet included in S that shares information $I(\mathbf{f}_j^+; \mathbf{y}^*)$ with the target c while exhibiting minimal redundancy information $\sum_{\mathbf{f}_j^+ \in S} I(\mathbf{f}_j^+; \mathbf{f}_i^+)$ with features selected in S. The optimization has become

$$\tilde{\mathbf{f}} = \arg \max_{\mathbf{f}_j^+ \in \Omega \setminus S} \left[I(\mathbf{f}_j^+; \mathbf{y}^*) - \frac{1}{|S|} \sum_{\mathbf{f}_j^+ \in S} I(\mathbf{f}_j^+; \mathbf{f}_i^+) \right]. \tag{3.16}$$

The final selected features are denoted as $\mathbf{F}$, and are defined as follows:

$$\tilde{\mathbf{F}} = \begin{bmatrix} \tilde{\mathbf{f}}_1^\top \\ \tilde{\mathbf{f}}_2^\top \\ \vdots \\ \tilde{\mathbf{f}}_{|S|}^\top \end{bmatrix}, \quad \tilde{\mathbf{f}}_n \in \mathbb{R}^L, \quad n = 1, \ldots, |S|, \tag{3.17}$$

in which $\tilde{\mathbf{f}}_n \in S$ is the n-th selected features, and $|S|$ in the number of selected features.

3.1.2 Data Alignment

To align the static vector $\mathbf{s} \in \mathbb{R}^{D_s}$ with the selected dynamic feature matrix $\tilde{\mathbf{F}} \in \mathbb{R}^{D_{|S|} \times L}$ in the time dimension, $\mathbf{s}$ is broadcast and replicated along the temporal axis, resulting in a matrix $\mathbf{S} \in \mathbb{R}^{D_s \times L}$, where each column is identical to $\mathbf{s}$.

To integrate both dynamic and static information, the $\tilde{\mathbf{F}}$ is concatenated with the broadcast static feature matrix $\mathbf{S}$ along the feature dimension. This results in an augmented feature matrix denoted as $\bar{\mathbf{F}} \in \mathbb{R}^{(|S|+D_s) \times L}$:

$$\bar{\mathbf{F}} = \begin{bmatrix} \tilde{\mathbf{F}} \\ \mathbf{S} \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{f}}_1 & \bar{\mathbf{f}}_2 & \cdots & \bar{\mathbf{f}}_L \end{bmatrix}, \tag{3.18}$$

in which each column of $\bar{\mathbf{F}}$, $\bar{\mathbf{f}}_n \in \mathbb{R}^{|S|+D_s}$, corresponds to a timestamp concatenated with both the measured data of this time step and the replicated static variables.

To prepare the input for the prediction model, a sliding window of length T is applied to the augmented feature matrix $\bar{\mathbf{F}}$ along the temporal axis. For simplicity, denote $N = L - T + 1$. This process slices $\bar{\mathbf{F}}$ into overlapping segments and constructs the input tensor $\mathbf{X} \in \mathbb{R}^{N \times (|S|+D_s) \times T}$. The i-th element of $\mathbf{X}$ can be represented as

$$\mathbf{X}_i = \begin{bmatrix} \bar{\mathbf{f}}_{i-T+1} & \bar{\mathbf{f}}_{i-T+2} & \cdots & \bar{\mathbf{f}}_i \end{bmatrix}, \quad i = T, T+1, \ldots, L, \tag{3.19}$$

corresponding to a windowed segment centered at time step i. For consistency in the following parts, the first dimension of vectors, such as $\hat{\mathbf{y}}$, is truncated to match the dimension of $\mathbf{X}_i$. The length of variables will be N rather than L.

3.2 Dynamic Relative RUL Prediction

In the previous section, the preprocessing steps have generated a dataset for deep learning models. This section will focus on dynamic prediction of relative RUL, which involves continuously estimating the degradation state of a device based on sequential measurements gathered during operation. By continuously updating based on new data measured during operation, dynamic approaches effectively track the trends of GaN HEMTs, thereby improving the accuracy of predictions.

To achieve relative RUL prediction, we develop a neural network that combines the temporal sensitivity of temporal convolutional networks [37] with the adaptive channel and position selection capability of an attention mechanism. The proposed design enables the model to capture both local degradation trends and global temporal dependencies, exhibiting strong robustness to various working conditions.

3.2.1 Temporal Convolution Networks

The proposed model utilizes TCNs, an enhanced form of CNNs, to achieve deeper feature extraction on manually selected input features while capturing long-range temporal dependencies. Before introducing the TCN architecture in detail, it is essential to revisit basic CNNs. This review will facilitate a more precise comparison with the improvements introduced in TCNs and help explain why the TCN is a better model to deal with our temporal task.

3.2.1.1 Revisit of CNN

CNNs are a widely used class of deep learning architectures that focus on capturing local patterns within structured data, such as images, signals, and time series. The core of a CNN is the convolutional layer, which utilizes multiple small, trainable filters to slide across the input data and extract localized patterns. Unlike fully connected networks, where each neuron processes all input features, convolutional layers rely on local receptive fields and shared weights, enabling efficient feature extraction with significantly fewer parameters. For simplicity, assume that the convolution kernel size is an odd number k, with the kernel center located at position t in the one-dimensional sequence $\mathbf{x} \in \mathbb{R}^T$ with one one channel. The convolution is defined as

$$y_t = \sum_{i=-(k-1)/2}^{(k-1)/2} w_i \cdot x_{t+i}, \tag{3.20}$$

where w_i denotes the kernel weights and x_{t+i} is the input sequence element at position $t + i$.

For a more complex case with the input with multiple channels $\mathbf{x} \in \mathbb{R}^{T\times C}$, where T is the sequence length and C is the number of channels, and a kernel $\mathbf{w} \in \mathbb{R}^{k\times C}$, the convolution at position t can be computed as

$$y_t = \sum_{i=-(k-1)/2}^{(k-1)/2} \sum_{c=1}^{C} w_{i,c} \cdot x_{t+i,c}, \tag{3.21}$$

which means y_t is the sum of the convolution on every input channel.

The structure of a CNN is shown in Figure 3.3. Each neuron in a convolutional layer is linked to a limited receptive field, and all units share the same filter parameters. This architectural design not only minimizes the number of trainable weights, thereby accelerating convergence, but also improves generalization performance by reducing the risk of overfitting. Due to their localized receptive fields and weight-sharing mechanisms, CNNs exhibit a strong ability to retain short-term dependencies within sequential data.

In a practical CNN, A convolutional layer is typically followed by non-linear activation functions, pooling layers, and normalization strategies to enhance learning capacity and improve generalization. The activation function introduces nonlinearity into CNNs, pooling ensures a rapid increase of the receptive field, and normalization stabilizes the training process by mitigating internal covariate shift and improving convergence.

CNNs are especially powerful in detecting local patterns; however, their ability to model long-range dependencies is limited by the size of the convolutional kernels and the depth of the network. A commonly used method for expanding the receptive field, pooling, tends to incur a loss of information. In time series applications such as RUL prediction, this constraint can limit performance, motivating the development of extended architectures like TCNs that are better designed for sequential modeling.

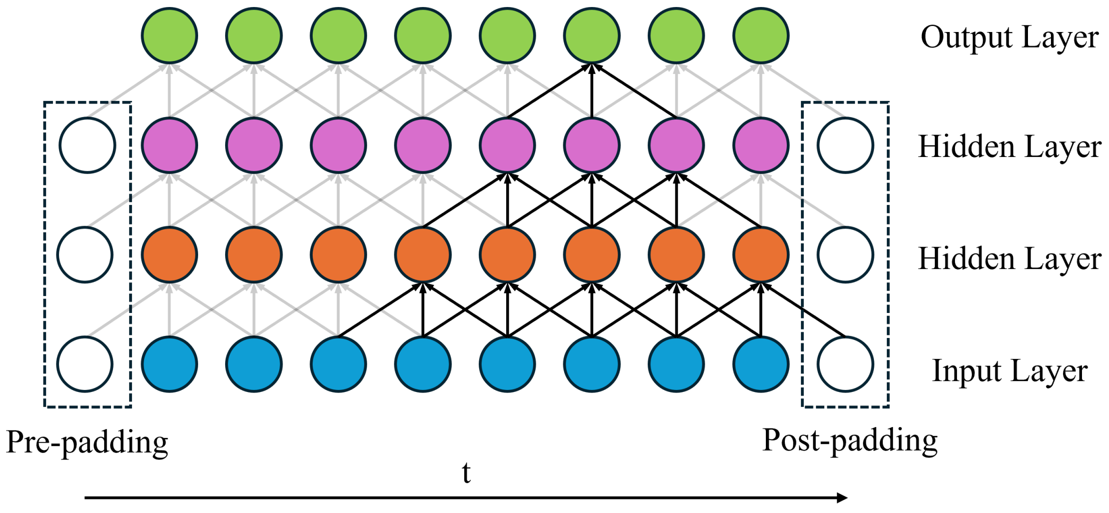


Figure 3.3: Architecture of a vanilla CNN.

3.2.1.2 Architecture of TCN

Building upon the architectural foundation of traditional CNNs, TCNs incorporate several key modifications. The two most important features are causal convolution and dilated convolution. The former ensures temporal causality during training, thereby preventing the model from accessing future measurements. This design choice eliminates the model's peeking at future information, which is unavailable during online deployment and could otherwise lead to performance degradation. The latter, in turn, effectively and efficiently expands the receptive field, enabling the model to capture both short- and long-term temporal dependencies. In addition, TCN incorporates shortcut connections between the input and output of each block to mitigate the vanishing gradient problem commonly encountered in deep networks, thereby enhancing the model's training stability. These design enhancements enable the network to capture both short- and long-term temporal dependencies more effectively, thereby improving overall forecasting performance.

3.2.1.3 Causal Convolution

Causal convolution addresses a fundamental requirement in time series prediction by ensuring that the output at any time step is generated solely from present and past inputs. As depicted in Figure 3.4, this operation strictly enforces causality by computing the output y_t at time t using only previous and current variables $x_1, x_2, \ldots, x_t$, while explicitly excluding future values x_{t+i} $(i > 0)$. This causality constraint is crucial in real-world prognostic scenarios, where future data is inherently unavailable during inference. In standard convolution operations, the output at a given time step is computed using a symmetric window of input values, which may include future data points. While this is suitable for spatial tasks such as image recognition, it poses a critical issue in temporal prediction problems: in real-world deployment, future measurements are not available at inference time. Using such information during training can lead to data leakage, resulting in significant performance degradation when the model is applied in online settings.

Causal convolution addresses this issue by shifting the convolutional filter such that the output at time step t is computed exclusively from the current and previous inputs. This ensures that the model adheres to the principle of temporal causality, maintaining consistency between training and inference behavior. Compared with regular convolution, the filter in causal convolution is shifted to the left (previous side), and the right side is aligned to ensure that only current and past inputs contribute to the output. Given an input sequence $\mathbf{x} \in \mathbb{R}^T$, a filter $\mathbf{w} \in \mathbb{R}^k$ of size k, and a current time step t, the causal convolution is defined as:

$$y_t = \sum_{i=1-k}^{0} w_i \cdot x_{t+i} \tag{3.22}$$

In implementation, to preserve the temporal alignment between input and output sequences, causal convolution employs left-sided zero-padding, effectively allowing the network to retain the sequence length while avoiding future leakage. Figure 3.4 illustrates how a network composed of causal convolutional layers works. Furthermore, stacking multiple causal layers increases the temporal receptive field, enabling each output unit to capture dependencies across a wider historical window. This hierarchical expansion of context facilitates long-range temporal modeling, which is essential for accurate RUL prediction. However, such a design also renders the traditional method of enlarging the receptive field, pooling, impractical, as pooling would disrupt the causal structure. Therefore, TCNs require an alternative mechanism to expand the receptive field.

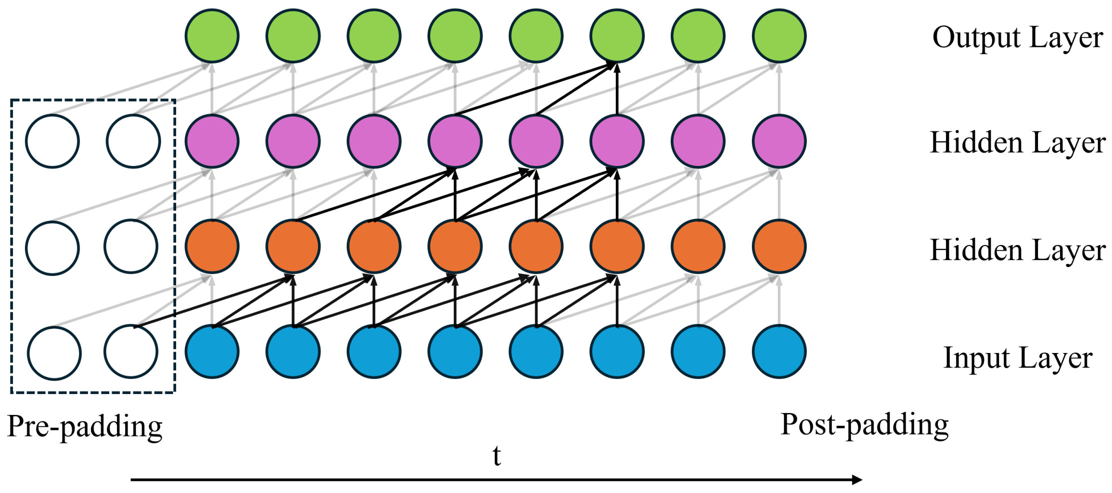


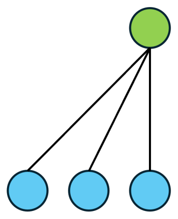
Figure 3.4: Mechanism of causal convolution, in which a neuron relies only on the current and past input features.

3.2.1.4 Dilated Convolution

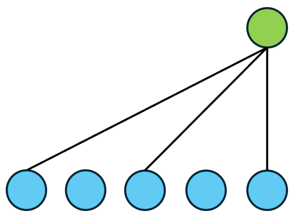
For a model based on causal convolution, if the receptive field is expanded solely by stacking additional convolutional layers, the model can only linearly capture long-range dependencies with respect to the number of layers. The size of the receptive field increases linearly and is limited by the kernel size, which necessitates a deep network with many layers to relate distant temporal elements. However, as the length of the time series increases, this leads to a sharp rise in the number of convolutional kernels, resulting in an excessive number of model parameters. Such complexity increases training time and risks overfitting, vanishing gradients, or exploding gradients.

To address these issues, it becomes essential to adopt a more efficient strategy for expanding the receptive field. In traditional CNNs used for image recognition tasks, receptive fields are commonly enlarged through pooling operations, which reduce dimensionality by applying either average or max filters to the input features. The rationale behind pooling in two-dimensional images is that visual information often contains redundancy. Therefore, essential features are preserved mainly after pooling, and irrelevant variations or noise might be suppressed, which means improving the signal-to-noise ratio. Nevertheless, the pooling operation disrupts the alignment between the input and output sequence lengths in causal convolution and inevitably leads to loss of critical temporal information.

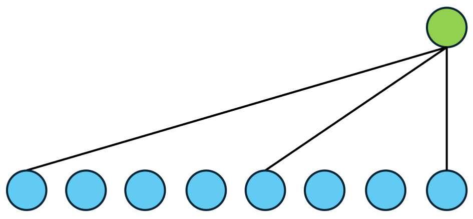
To overcome the limitations of linear receptive field growth, TCN integrates dilated convolution, which effectively enlarges the receptive field by enabling spaced sampling in the input sequence. This method introduces a dilation rate parameter, denoted as d, representing the interval at which input points are sampled. Figure 3.5 depicts the sampling configurations for a convolutional kernel size of $k = 3$ with various dilation rates. In Figure 3.5(a), the non-dilated convolution corresponds to a dilation rate of $d = 1$, in which inputs are continuous. That is how convolution works in a regular convolution. While Figure 3.5(b), Figure 3.5(c), Figure 3.5(d) show that for dilation rates $d > 1$, the sampling skips $d - 1$ neurons between each sampled point.



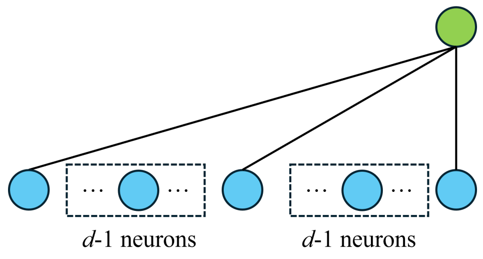
(a) $d = 1$. Distance between 2 inputs is 1. Inputs are continuous.



(b) $d = 2$. Distance between 2 inputs is 2. 1 neuron is skipped.



(c) $d = 4$. Distance between 2 inputs is 4. 3 neuron is skipped.



(d) For any $d \in \mathbb{N}^+$ Distance between 2 inputs is d. $d - 1$ neuron is skipped.

Figure 3.5: Effect of dilated convolution with different dilation rate d.

Given a filter of size k and considering causality, dilation rate d, and an input sequence $\mathbf{x}$, the dilated convolution at this t is computed as

$$y_t = \sum_{i=1-k}^{0} w_i \cdot x_{t+d\cdot i} \tag{3.23}$$

In an application scenario, the dilation rate d increases exponentially with the layer index, following $d = 2^{(n-1)}$, where n denotes the hidden layer number. This exponential growth strategy enables the network to capture the entire global receptive field with a minimal number of layers while avoiding the gridding effect, in which some neurons between the first and the last inputs are not convolved if d is not selected correctly. Consequently, a neuron at a higher layer can integrate information from all preceding and current input neurons. Figure 3.6 illustrates a three-layer causal dilated convolutional network with an input sequence length of $l = 8$, and kernel size $k = 2$. In this architecture, a neuron in the output layer establishes connections to every input neuron occurring at or before its time step, thus effectively encoding the entire past sequence history after only three hidden layers.

3.2.1.5 Dropout

Dropout [38] is an effective method to prevent overfitting by randomly disabling neurons during the training phase with a predefined probability. This forces the neural

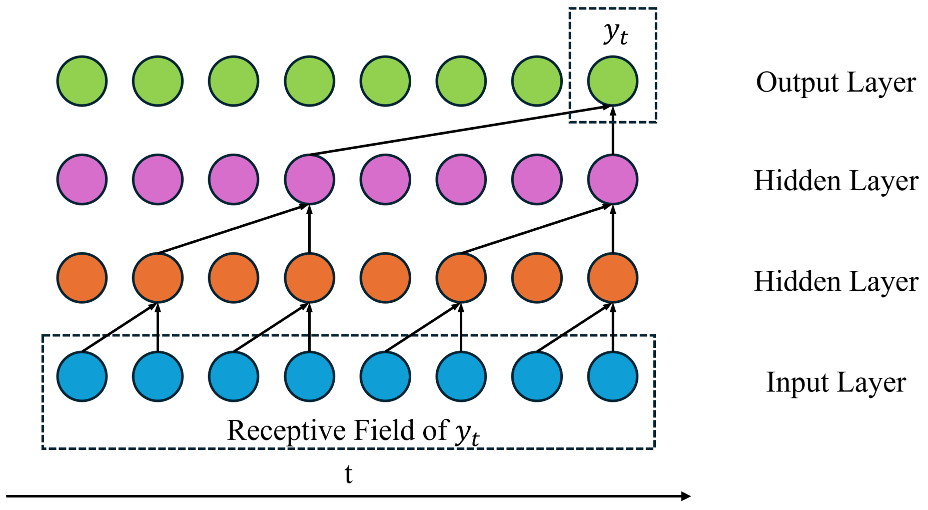


Figure 3.6: A causal dilated convolutional network, in which $k = 2$.

network to learn more fundamental and stable feature representations, reducing the risk of complex dependencies among neurons. Through multiple training iterations, the ensemble of subnetworks generated by different dropout patterns effectively averages out overfitting tendencies, leading to improved generalization performance. In the inference phase, the dropout rate is set to 0, which means all neurons are fully operational to generate predictions. The TCN architecture integrates dropout layers to strengthen its resilience and robustness.

3.2.1.6 Residual Module

Residual module was proposed in Residual networks (ResNets) [39]. Although deeper neural networks theoretically possess stronger representational capacity, in practice, their performance is often hindered by gradient vanishing or exploding phenomena. These issues lead to degradation in accuracy, making deeper networks perform worse than shallower ones. The residual module offers an effective architectural solution to this challenge by enabling identity mappings through skip connections, enhancing gradient flow, and preventing performance degradation as the network grows deeper.

A conventional feedforward neural network aims to learn a direct mapping from an input x to a target output $H(x)$, where $H(x)$ represents the desired complex transformation, and is entirely modeled by the network layers:

$$H(x) = F(x), \tag{3.24}$$

in which $F(x)$ denotes the full nonlinear transformation applied to x. In CNNs, $F(x)$ is typically implemented using convolutional layers, activation functions, and dropout.

However, in a residual learning framework, the network instead focuses on learning the residual function between the input and the output. Rather than approximating $H(x)$ directly, the network is restructured to model:

$$H(x) = F(x) + x, \tag{3.25}$$

in which $H(x)$ still represents the desired final output, but now $F(x)$ represents the residual mapping, i.e., the change or adjustment that needs to be applied to the input x to obtain $H(x)$. $F(x)$ is the main path that the network is trained to learn features. This design enables the network to learn identity mappings more easily when necessary, and to focus on modeling only the difference between the input and the target output. This has been shown to ease the optimization process, especially in deep architectures, and improve gradient flow during training.

As input and output channels differ in CNNs, a 1×1 convolution is employed to align dimensions. This transformation does not introduce additional parameters beyond reshaping the input. The output channels of the residual path are matched to those of the main convolutional path, allowing element-wise addition of the main path and the residual path.

3.2.2 Attention Mechanism

Initially proposed in the context of natural language processing and machine translation, attention mechanisms [8] allow a network to weigh different input elements according to their relevance to the current task, significantly improving both interpretability and performance. In recent years, it has emerged as a popular component in deep learning architectures, enabling models to focus on the most informative parts of the input dynamically.

3.2.2.1 Self Attention Mechanism

Self-attention mechanism [8] is a widely known form of attention mechanism. It operates by computing weighted relationships between all input elements through a set of learnable projections known as query $\mathbf{Q} = \mathbf{X}\mathbf{W}^Q$, which determines what each time step wants to know, key $\mathbf{K} = \mathbf{X}\mathbf{W}^K$, which represents what each time step can offer, and value $\mathbf{V} = \mathbf{X}\mathbf{W}^V$, the actual information content weighed by attention. All of them are linear projections of the same input sequence $\mathbf{X}$ via linear transform matrix $\mathbf{W}^Q$, $\mathbf{W}^V$, and $\mathbf{W}^K$. The attention is calculated as

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}, \tag{3.26}$$

in which d_k is the factor that scales the sum to avoid saturation of the softmax function. While highly expressive, self-attention involves substantial computational complexity and is primarily designed to capture pairwise dependencies in long sequences.

3.2.2.2 Lightweight Attention

Unlike self-attention mechanisms that compute query-key-value interactions across sequence elements, lightweight approaches, which focus on the importance of elements in different positions, rather than the interaction between modeling pairwise interactions across time or sequence positions, are used in the proposed method. In this mechanism, importance scores for individual channels or positions are generated using lightweight computations, typically involving global average pooling and a subsequent fully connected transformation. The resulting attention map is then broadcast and applied to the input sequence through element-wise multiplication, therefore modulating the feature representation according to learned relevance. This process enhances the model's ability to selectively focus on critical degradation signals without incurring significant computational cost. These modules are particularly well-suited for convolutional architectures in vision and sensor-based tasks, where the goal is to enhance representational power by emphasizing informative features while maintaining computational efficiency. As our task involves p-GAN HEMT degradation monitoring, where different sensor channels may exhibit varying levels of predictive significance, lightweight attention methods are particularly well-suited for our application, as they enable the model to dynamically identify and emphasize the most informative sensor signals throughout the degradation process. In comparison to self-attention, these light methods offer significant advantages in terms of parameter efficiency, interpretability, and deployment on resource-constrained devices. Two types of lightweight attention mechanisms, temporal attention and channel attention, are employed in the proposed methods.

3.2.2.3 Convolutional Block Attention Module

The Convolutional Block Attention Module (CBAM) [9] was originally proposed in the field of computer vision, where it combines Channel Attention (CA) and Spatial Attention (SA) to enhance feature representation. The spatial attention component was designed to help the model focus on critical object regions in an image while suppressing irrelevant background information.

In the context of RUL prediction, we adopt a similar architectural concept but with a different interpretation. Instead of spatial locations in an image, we aim to regulate the importance of different time steps within a temporal slice, as these time points may vary in relevance depending on their proximity to the prediction horizon. To avoid semantic confusion, this component is referred to as Temporal Attention (TA) rather than Spatial Attention, as it operates along the time dimension rather than the spatial dimension typically found in image-based tasks.

Figure 3.7 illustrates the overall architecture of the CBAM module, which is composed of a Channel Attention module followed by a Temporal Attention module. The two components are applied sequentially to refine the feature representation through successive attention weighting. In our framework, the temporal features extracted by the previous layer are further refined by CBAM, which consists of two sequential submodules: the Channel Attention module and the Temporal Attention module.

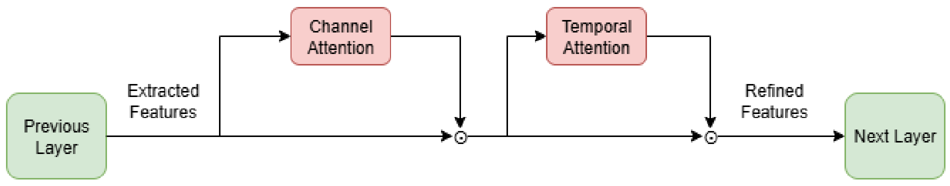


Figure 3.7: Architecture of CBAM.

3.2.2.4 Channel Attention Module

Figure 3.8 shows how CA calculates the weights of channels. The resulting vectors are then passed through a bottleneck structure based on a shared Multi-Layer Perceptron (MLP) layer, composed of two fully connected layers, which allows the network to learn nonlinear dependencies among feature channels. This structure enables the model to capture global inter-channel relationships, emphasizing the most informative channels while suppressing those that are irrelevant or noisy.

Given an feature tensor $\mathbf{F}^{\text{in}} \in \mathbb{R}^{C \times T}$ from the previous layer, where C is the number of channels and T is the sequence length, channel-wise attention tries to compute channel descriptors $\mathbf{F}^{c}_{\text{avg}}$ and $\mathbf{F}^{c}_{\text{max}} \in \mathbb{R}^{C}$, which are then used to generate attention weights that scale each channel accordingly.

Firstly, spatial information is first compressed along the temporal dimension using both global average pooling and global max pooling, reducing each channel's temporal sequence to a single scalar descriptor. These two descriptors respectively capture the average and the most salient activations of each channel. Global information across the sequence (or temporal locations) is aggregated using pooling operations. Specifically, there are

$$\mathbf{F}^{c}_{\text{avg}} = \text{GAP}_t(\mathbf{F}^{\text{in}}) = \frac{1}{T}\sum_{t=1}^{T} \mathbf{F}^{\text{in}}_{:,t}, \quad \mathbf{F}^{c}_{\text{max}} = \text{GMP}_t(\mathbf{F}^{\text{in}}) = \max_{t \in [1,T]} \mathbf{F}^{\text{in}}_{:,t}, \tag{3.27}$$

where $\text{GAP}_t(\cdot)$ and $\text{GMP}_t(\cdot)$ denote global average pooling and global max pooling over time steps, respectively. These two descriptors capture different aspects of global channel-wise activation: average trends and strong activations.

After processing, the outputs from the average-pooled and max-pooled branches are aggregated through element-wise summation and passed through a sigmoid activation to generate channel-wise attention weights via the bottleneck structure with a shared MLP. These weights are subsequently applied to the original feature map via channel-wise multiplication, thus modulating each channel's contribution to the final representation. Both descriptors are passed through a shared bottleneck network to model non-linear inter-channel dependencies and compute the attention weights, as

$$\mathbf{a}^{c}_{\text{avg}} = \mathbf{W}_2 \cdot \text{ReLU}(\mathbf{W}_1 \cdot \mathbf{F}^{c}_{\text{avg}}), \quad \mathbf{a}^{c}_{\text{max}} = \mathbf{W}_2 \cdot \text{ReLU}(\mathbf{W}_1 \cdot \mathbf{F}^{c}_{\text{max}}), \tag{3.28}$$

in which $\mathbf{W}_1 \in \mathbb{R}^{\frac{C}{r} \times C}$ and $\mathbf{W}_2 \in \mathbb{R}^{C \times \frac{C}{r}}$ are trainable parameters in the bottleneck structure, and $\text{ReLU}(\cdot)$ denotes the ReLU function. Notice that the weights $\mathbf{W}_1$ and $\mathbf{W}_2$ are shared. The outputs are combined to yield the final attention vector

$$\mathbf{a}_c = \sigma(\mathbf{a}^{c}_{\text{avg}} + \mathbf{a}^{c}_{\text{max}}), \tag{3.29}$$

in which $\sigma(\cdot)$ denotes the sigmoid function. The weight of channels $\mathbf{a}_c$ is the sum of the refined average and the maximum of temporal positions. Finally, the original feature tensor is reweighted by the attention vector using channel-wise multiplication

$$\mathbf{F}^{\text{out}} = \mathbf{a}_c \odot \mathbf{F}^{\text{in}}. \tag{3.30}$$

This mechanism allows the network to emphasize more informative feature channels while suppressing irrelevant or noisy ones. In time series applications such as RUL prediction with multivariate sensors, channel-wise attention helps to automatically learn which sensors (or derived features) contribute more significantly at each stage of the degradation process. Moreover, it introduces minimal computational overhead, making it suitable for real-time or embedded predictive maintenance scenarios.

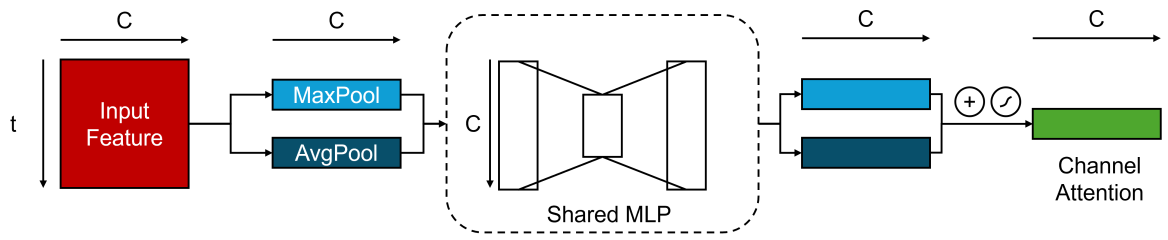


Figure 3.8: Architecture of channel attention module. The Shared MLP part block is rotated by 90 degrees for visual clarity.

3.2.2.5 Temporal Attention Module

The temporal attention module adopts a strategy similar to that of channel attention. Specifically, global max pooling and average pooling are applied along the channel dimension to compress it into a single value for each time step, thereby capturing both the most prominent and the average response at each temporal location. Unlike the channel attention module, which uses fully connected layers for nonlinear transformation, the temporal attention mechanism takes into account the local continuity and correlation inherent in time series data, as well as the extended length of time series inputs. Therefore, instead of employing a fully connected bottleneck, we use a one-dimensional convolution to extract localized temporal features. The resulting feature map is then directly passed through a sigmoid activation function to obtain attention weights for each time step. Notably, this design eliminates the reconstruction step typically found in other attention modules, resulting in a more efficient and interpretable mechanism for modulating temporal features.

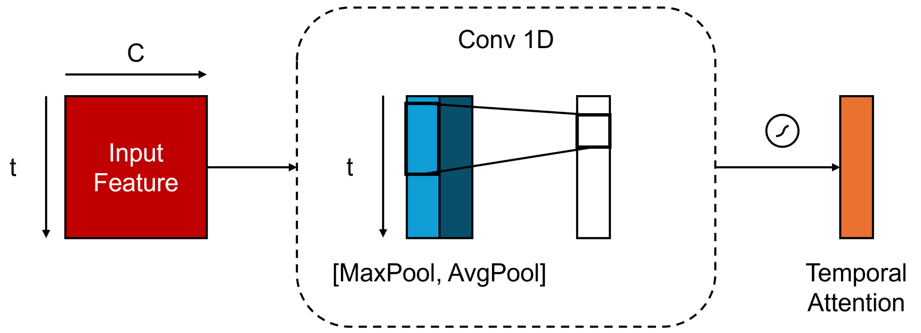


Figure 3.9: Architecture of temporal attention module.

Temporal attention is based on the assumption that not all time steps contribute equally to the final prediction in time series modeling. Some time steps carry more critical information due to sudden transitions, fault signatures, or degradation trends. To address this, we introduce a temporal attention mechanism that dynamically adjusts the importance of each time step by learning a time-dependent weighting function. This mechanism helps the model focus on key temporal regions while downplaying irrelevant or noisy intervals.

Similarly, given an intermediate feature matrix $\mathbf{F}^{\text{in}} \in \mathbb{R}^{C\times T}$, where C is the number of channels and T is the sequence length, temporal attention computes a scalar importance score for each time step. First, the input is compressed along the channel dimension using global pooling, as

$$\mathbf{F}^{t}_{\text{avg}} = \text{GAP}_c(\mathbf{F}^{\text{in}}) = \frac{1}{C}\sum_{c=1}^{C}\mathbf{F}^{\text{in}}_{c,:}, \quad \mathbf{F}^{t}_{\text{max}} = \text{GMP}_c(\mathbf{F}^{\text{in}}) = \max_{c\in[1,C]}\mathbf{F}^{\text{in}}_{c,:}, \tag{3.31}$$

where $\text{GAP}_c(\cdot)$ and $\text{GMP}_c(\cdot)$ denote global average pooling and global max pooling over time steps, respectively. These two descriptors $\mathbf{F}^{c}_{\text{avg}}$ and $\mathbf{F}^{c}_{\text{max}}$ capture the average and maximum activation patterns over time. To model local temporal dependencies more effectively than fully connected layers, a lightweight convolutional layer is used:

$$\mathbf{a}_t = \sigma\left(\text{Conv1D}([\mathbf{F}^{t}_{\text{avg}}|\mathbf{F}^{t}_{\text{max}}])\right). \tag{3.32}$$

The attention vector $\mathbf{a}_t$ is then broadcast and applied to the original feature map along the temporal axis

$$\mathbf{F}^{\text{out}} = \mathbf{a}_t \odot \mathbf{F}^{\text{in}}. \tag{3.33}$$

By introducing temporal attention in this form, the model is able to automatically emphasize the most informative time steps during degradation progression, which is especially valuable in tasks such as RUL prediction, anomaly detection, or fault localization in multivariate sensor data.

3.2.3 Improved CBAM Attention Module

In addition to the fundamental CBAM model, we further employ an alternative framework that incorporates an improved attention module named Improved CBAM Attention (IAT)[40], which replaces the original CBAM module. Compared with standard CBAM, the IAT module introduces stochastic pooling alongside global max pooling and average pooling within the attention generation process. Stochastic pooling serves as a regularization technique that introduces controlled randomness to improve model robustness and reduce information loss. Unlike max pooling, which selects the strongest response, or average pooling, which treats all features equally, stochastic pooling assigns a selection probability to each activation based on its magnitude. Specifically, activations with larger values are more likely to be selected, but the selection is not deterministic. This mitigates the over-reliance on extreme values and enhances the model's generalization capacity, similar in spirit to dropout.

3.2.4 Attention-Augmented TCN Block

Attention-Augmented TCN Block is proposed as the key component of the dynamic prediction network. In each block, the CBAM module is incorporated into TCN to enhance the important information extracted by TCN. As illustrated in Figure 3.10, each Block cell is composed of two stacked temporal convolution layers. For simplicity, activation functions and dropout layers following each temporal convolution are not added to the diagram. A residual path connects the input and output of the temporal convolution layers to ensure information flow. The CBAM module is placed after the convolution to reinforce the most informative features. The overall model, TCN+CBAM, is constructed by stacking multiple such Attention-Augmented TCN Blocks.

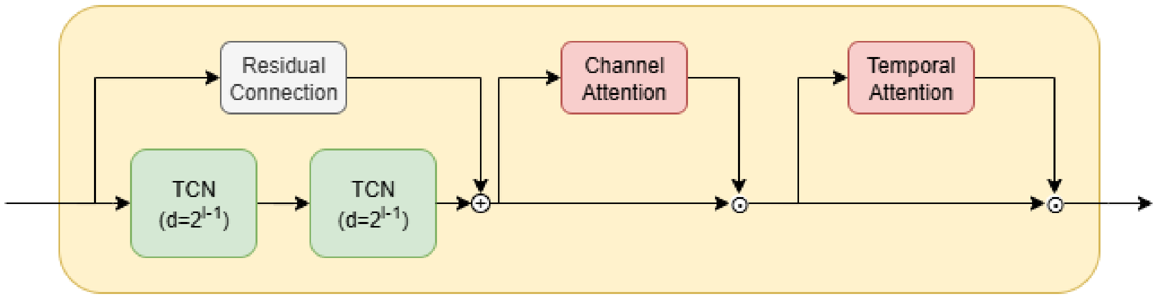


Figure 3.10: Attention-Augmented TCN Block.

3.2.5 Training Objective

To train the proposed model, the Mean Squared Error (MSE) with an L2 regularization term is employed as the loss function. MSE is a widely used objective in regression tasks, measuring the average squared difference between the predicted values and the ground truth. The loss is defined as

$$\mathcal{L}_{\text{Total}} = \mathcal{L}_{\text{MSE}} + \lambda \cdot \mathcal{L}_{\text{reg}} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i^* - y_i^*)^2 + \lambda\|\mathbf{w}\|_2 \tag{3.34}$$

where $\hat{y}_i^*$ denotes the predicted relative RUL, y_i^* denotes the true relative RUL label, N is the number of samples, λ is the strength of the regularization, and $\|\mathbf{w}\|_2$ denotes the squared L2 norm of the weight vector, equal to the sum of the squared weights. This loss function penalizes larger deviations more severely, encouraging the model to make precise predictions. Given the continuous and bounded nature of the label y_i^* in the relative RUL prediction task, MSE serves as a practical choice for training. Regularization ensures that the model remains simple and generalizable by penalizing weights and encouraging smooth learning behavior.

3.3 Absolute RUL Reconstruction

In the section, the relative lifetime left $\hat{y}^*$ of the device is estimated, giving an indication of the degradation process. However, we also want to know how many days are left on the device. That requires methods to quantify the days by reconstructing the absolute RUL from the relative RUL.

If the total life L of the device is known, that will be a trivial task, as we can calculate the absolute RUL by scaling

$$\hat{\mathbf{y}}(t) = L\hat{\mathbf{y}}^*(t). \tag{3.35}$$

However, this is meaningless because it is a clear data leakage. If the true lifetime L is already known, any estimation of the RUL becomes necessary, since L can be directly substituted into Equation 3.1 to obtain the exact RUL. Recalling Equation 3.4, we can find a breakthrough by exploiting the known elapsed time t. By rearranging terms and inverting the numerator and denominator, L can be rewritten as a function of $\text{RUL}_{\%}$ and t.

$$L(t) = \frac{t}{1 - y^*(t)}. \tag{3.36}$$

Thus, an estimator of L can be obtained

$$\hat{L}(t) = \frac{t}{1 - \hat{y}^*(t)}. \tag{3.37}$$

Substituting $\hat{L}$ into Equation 3.1, the absolute RUL is reconstrued

$$\hat{y}(t) = -t + \hat{L}(t), 0 \leq t \leq L. \tag{3.38}$$

Or we can skip $\hat{L}$ to get the absolute RUL based on the relative RUL

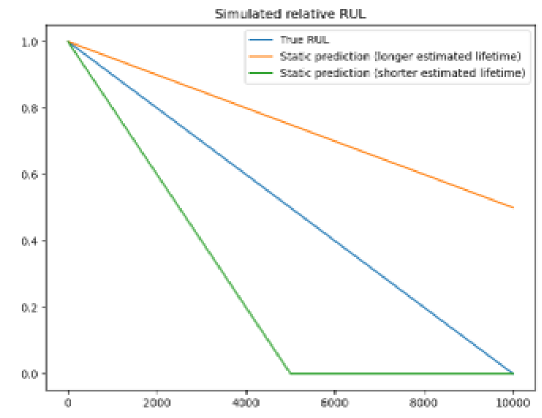
$$\hat{y}(t) = \frac{\hat{y}^*(t)}{1 - \hat{y}^*(t)} t \tag{3.39}$$

Eventually, we have found a basic estimator of $\hat{y}(t)$ that depends on the real lifetime L.

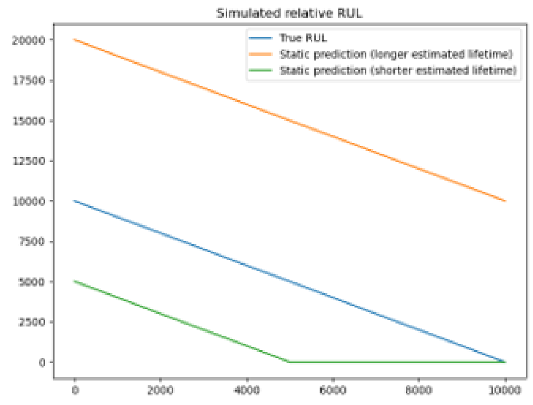
3.4 Static Prediction

To explore the correlation between device lifetime and initial measurements such as current and environmental variables, a static prediction model is tested. Unlike the dynamic prediction model, which operates on sliced sequences of time-varying measurements, the static prediction model relies solely on initial features, including environmental parameters and the first recorded measurements, to directly estimate the total device lifetime, expressed in logarithmic scale.

Due to its design, the static model is extremely limited in the amount of training data it can use, as each trained device only provides one training sample. This constraint makes it a lightweight but highly constrained baseline.



(a) Simulated relative RUL based on static prediction.



(b) Simulated absolute RUL based on static prediction.

Figure 3.11: Two kinds of simulated RUL based on static prediction.

To enable a fair comparison between the static and dynamic prediction models, a simulated RUL sequence must be generated based on the lifetime prediction $\hat{L}$ from the static model. Following the same linear degradation assumption, a static prediction sequence $\hat{\mathbf{y}}$ is constructed to represent the estimated RUL at each time step. Figure 3.11 illustrates such a simulation under relative and absolute RUL prediction cases.

In the case of relative RUL prediction, the simulated RUL has a different slope but the same start value of 1.

- If $\hat{L} < L$ (i.e., the predicted lifetime is shorter than the ground truth), the predicted RUL linearly decays to zero at $t = \hat{L}$, after which the remaining time steps are padded with zeros:

$$\hat{y}_t = \begin{cases} 1 - \frac{t}{\hat{L}}, & t \leq \hat{L}, \\ 0, & t > \hat{L}. \end{cases} \tag{3.40}$$

- If $\hat{L} \geq L$ (i.e., the predicted lifetime is longer than the ground truth), the RUL sequence is truncated at L:

$$\hat{y}_t = 1 - \frac{t}{\hat{L}}, \quad 0 \leq t \leq L. \tag{3.41}$$

Similarly, in the case of absolute RUL prediction, the simulated RUL has the same slope of -1 but a different bias, which means it will be parallel to the true RUL.

- If $\hat{L} < L$:

$$\hat{y}_t = \begin{cases} \hat{L} - t, & t \leq \hat{L}, \\ 0, & t > \hat{L}. \end{cases} \tag{3.42}$$

- If $\hat{L} \geq L$:

$$\hat{y}_t = \hat{L} - t, \quad 0 \leq t \leq L. \tag{3.43}$$

This formulation enables a direct performance comparison through standard metrics, which will be explained and analyzed in section 4.2.

4 Results and Analysis

4.1 Experimental setup

4.1.1 Dataset description

The datasets used in this thesis were collected by the University of Bologna (UNIBO) on the p-GaN HEMT device model SGT65R65AL from STMicroelectronics. They tested the gate lifetimes of multiple p-GaN HEMT devices under various working conditions. During their tests, each device underwent a run-to-failure test under constant gate voltage V_G and temperature T conditions, with measurement data collected at 2-second intervals. Additionally, all samples were tested under a fixed drain voltage $V_D = 0.002V$. There are 60 devices tested under various V_G and T conditions. For each specific combination V_G, T, the number of tested devices ranges from 1 to 6, with most combinations containing 3 or 4 test devices.

Raw variables are shown in Figure 4.1. During the run-to-failure test on a device, the drain current I_D and gate current I_G were measured at 2-second intervals until the breakdown of the gate. Besides, the dataset includes the on-resistance R_{on}, which was calculated based on $R_{\text{on}} = V_D/I_D = 0.002V/I_D$, and its normalized version $R_{\text{on(norm)}}$ based on the first measured R_{on} during operation. In other words, R_{on} is a variable that depends on I_D and can be regarded as the scaled reciprocal of I_D.

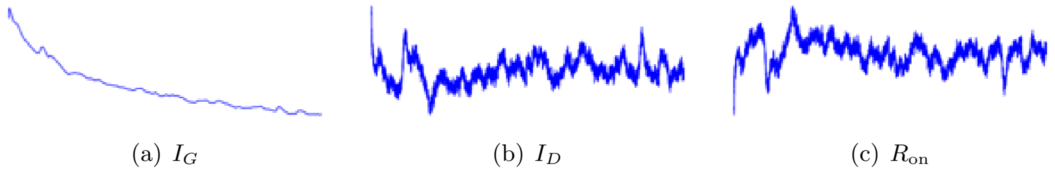

(a) I_G (b) I_D (c) R_{on}

Figure 4.1: Dynamic measurements of GaN HEMT Dataset.

Due to the presence of only two independent dynamic measurements in the collected data, manual feature engineering is performed to extract more informative representations from the available signals. As the I_G appears stable over time, while I_D and R_{on} exhibit significant noise over time, a second-order Savitzky–Golay filter is applied to I_D for smoothing. The smoothed I_D is then used to recompute the corresponding R_{on}, rather than relying on the original noise R_{on} values provided in the dataset. Additionally, the normalized on-resistance $R_{\text{on(norm)}}$ is not used, as it is simply a linear transformation of R_{on}, while other normalization techniques will be applied in the further steps.

Subsequently, the steps explained in section 3.1 are applied to extract and select features. The final choice of features is shown in Table 4.2. A total of 18 features, including I_G, I_D, and R_{on}, are selected from the 39 previously extracted features using the

Variable		Format	Description
Dynamic measurements	Time t	Timestamp (every 2s)	Collected from the start to the failure
	Gate current I_G	A	-
	Drain current I_D	A	-
	On-resistance R_{on}	Ω	V_D/I_D
	Normalized on-resistance $R_{\text{on(norm)}}$	-	Ron normalized
Static environments	Drain voltage V_D	V	Source grounded, fixed among all conditions
	Temperature T	C°	Temperature, fixed during the test
	Stress voltage V_G	V	Bias voltage stressed on the device, fixed during the test

Table 4.1: Descriptions of measurements from datasets.

mRMR algorithm. Additionally, logarithmically transformed t is applied as a feature. Since degradation often correlates with the time that the device has been in operation, explicitly providing the elapsed time can improve the model's ability to align patterns across samples with varying lifetimes. However, the total lifetime varies among various conditions, so a logarithmic transformation is applied to compress the range of time and improve the long-term stability while preserving temporal information by reducing the effect of large t. As working conditions V_G and T are decisive features of device behavior, both are selected as the input of the dynamic prediction model. On the other hand, V_G is not used as it is a constant among all devices, which means V_G cannot provide any helpful information in prediction. In total, 21 features are retained for the final input.

4.1.1.1 Hyperparameters

Preliminary experiments indicate that setting the sliding window size to 60 yields optimized model performance. Further increasing this value does not lead to noticeable performance improvement, but significantly increases the training time. With this window size, slicing the raw signal generates a large amount of training data, resulting in a high number of samples per epoch and thus enabling rapid model convergence. The number of epochs is set to 5, which is sufficient to guarantee model convergence. The kernel size of the TCN, the reduction ratio of the CA, and the kernel size of the TA are all set to commonly used values, as shown in Table 4.3. The regularization coefficient λ is set to 0.01, which effectively suppresses part of the noise and reduces prediction error. However, further increasing this value causes the model to output nearly constant values. Other parameters are listed in Table 4.3.

Although the receptive field of a TCN requires at least five stacked layers to cover an input sequence of length 60 in theory fully, preliminary experiments revealed that model

Variable	Description
t_{log}	Logarithmic transformed device operating time
V_G	Stress voltage
T	Temperature
I_G	Gate current
I_D	A Drain current
R_{on}	On resistance
$I_{G\text{-min}}$	Minimum of I_G
$I_{G\text{-max}}$	Maximum of I_G
$I_{G\text{-p2p}}$	Peak to Peak value of I_G
$I_{G\text{-RMS}}$	Root Mean Square of I_G
$I_{G\text{-std}}$	Standard deviation of I_G
$I_{G\text{-CLF}}$	Crest Factor of I_G
$I_{G\text{-CRF}}$	Clearance Factor of I_G
$I_{G\text{-IF}}$	Impulse Factor of I_G
$I_{D\text{-max}}$	Maximum of I_D
$I_{D\text{-RMS}}$	Root Mean Square of I_G
$R_{\text{on-min}}$	Minimum of R_{on}
$R_{\text{on-max}}$	Maximum of R_{on}
$R_{\text{on-RMS}}$	Root Mean Square of R_{on}
$R_{\text{on-std}}$	Standard deviation of R_{on}
$R_{\text{on-skew}}$	Skewness of R_{on}

Table 4.2: Final features used in dynamic relative RUL prediction.

performance converges to a stable value when the number of layers reaches 3. A higher number of hidden layers did not yield detectable improvements. This phenomenon can be attributed to two factors. First, the attention module builds connections between inputs over a long distance. Second, each cell contains two dilated convolutional layers, which further expand the receptive field linearly. Therefore, the model stack 3 Augmented TCN cells to predict RUL.

Parameter	value	Parameter	value
Window size	60	λ	0.01
Number of kernels in each layer	[16, 32, 64]	Epoch	5
Kernel size of TCN	3*1	Learning rate	0.0001
Reduction of CA	8	Batch size	256
Kernel size of TA	7		

Table 4.3: Hyperparameters used in dynamic relative RUL prediction.

4.1.2 Leave-One-Group-Out Cross-Validation

During training, we adopt a Leave-One-Group-Out (LOGO) Cross-Validation (CV) strategy. The dataset is first grouped based on the operating conditions, specifically the combinations of the two environmental variables, temperature T and stress voltage

V_G. Samples within the same group are tested under identical environmental conditions, and thus might exhibit similar degradation patterns. Distribution of groups is shown in Figure 4.2. In each fold of the LOGO CV, one group is held out as the test set, while all remaining groups are used for training. The performance of the model is assessed at the group level.

In this setting, the operating conditions of the test samples are entirely unknown to the model during training, requiring the model to infer degradation behavior under new $\{T, V_G\}$ based on prior knowledge. This evaluation scheme enhances the model's generalization ability and validates its capability to transfer across different thermal and electrical conditions. It also simulates real-world deployment scenarios, where the model must learn and adapt to new environments based on known information. Overall, this setup provides a stricter and more practical assessment of the model's robustness and resistance to overfitting.

4.1.3 Metrics

To quantitatively evaluate the performance of the RUL prediction model, we adopt five commonly used regression metrics: MSE, Root Mean Squared Error (RMSE), Normalized RMSE (NRMSE), Mean Absolute Percentage Error (MAPE), and Pearson Correlation Coefficient (PCC). Defined by

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2, \tag{4.1}$$

MSE measures the average squared difference between the predicted and true RUL values. It is the loss function during training as well. MSE penalizes larger errors more heavily and is sensitive to outliers.

RMSE is the square root of MSE, which provides an interpretable measure of prediction accuracy in the same unit as the target variable, given by

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}. \tag{4.2}$$

NRMSE normalizes RMSE by a reference value, such as the range or mean of the true RUL, given by

$$\mathrm{NRMSE} = \frac{\mathrm{RMSE}}{y_{\max} - y_{\min}}, \tag{4.3}$$

in which $y_{\max}$ and $y_{\max}$ are the maximum and minimum of the true label.

MAPE quantifies the average percentage error and is scale-independent, given by

$$\mathrm{MAPE} = \frac{100\%}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right|. \tag{4.4}$$

Symmetric Mean Absolute Percentage Error (SMAPE) is a variant of MAPE that addresses its instability when the true values are close to zero, given by

$$\text{SMAPE} = \frac{100\%}{N} \sum_{i=1}^{N} \frac{|y_i - \hat{y}_i|}{(|y_i| + |\hat{y}_i|)/2}. \tag{4.5}$$

The Pearson correlation coefficient evaluates the linear correlation between the predicted and actual RUL sequences. As the true RUL is assumed to be linear, Pearson provides a useful perspective on the shape of prediction, given by

$$\text{PCC} = \frac{\sum_{i=1}^{N}(y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})}{\sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}\sqrt{\sum_{i=1}^{N}(\hat{y}_i - \bar{\hat{y}})^2}}, \tag{4.6}$$

in which $\bar{y}$ and $\bar{\hat{y}}$ are the mean of the true values and the prediction.

4.1.4 Selection of Normal Device

Preliminary studies also revealed substantial variation in prediction performance across different devices. By comparing predictions and measurements of each device, it was observed that a practical prediction highly relied on the quality of I_G. Restricted by the availability of measurements, the performance of the model is still sensitive to the pattern in I_G Specifically, when I_G has a decreasing trend on a long time scale, the model is able to provide meaningful RUL predictions. Conversely, in the absence of such a trend, the model tends to fail, yielding random outputs. Based on this pattern, the devices were categorized into normal devices, which exhibit predictable degradation, and anomalous devices, which do not. Figure 4.3 illustrates prediction on the 2 classes and their corresponding I_G.

Further analysis shows that the likelihood of a device being anomalous is related to its operating conditions. As shown in Figure 4.2, anomalous devices predominantly occur under high V_G and low T conditions, whereas increasing T and reducing V_G increases the probability of normal degradation. In general, anomalous tend to have a shorter lifetime. For groups with all anomalous devices, the lifetime of devices is on the scale of minutes or seconds. Moreover, anomalous devices in a group with both classes have a shorter lifetime than their normal group members. Additional examples are provided in Appendix Figure A.1 and Figure A.2.

Since prediction on anomalous devices is not feasible, we propose to separate the two classes manually and focus on prediction for normal devices. To validate this approach, we conducted cross-experiments across different classification sets. Let $\mathcal{S}$ denote the set of all samples, which consists of normal samples $\mathcal{N}$ and anomalous samples $\mathcal{A}$, i.e.,

$$\mathcal{S} = \mathcal{S}_N \cup \mathcal{S}_A, \quad \mathcal{S}_N \cap \mathcal{S}_A = \emptyset.$$

Cross-experiments were conducted by training on the complete sample set $\mathcal{S}$ and on the normal subset $\mathcal{S}_n$, respectively, to evaluate model performance. The first two rows of the table correspond to test sets consisting solely of normal samples $\mathcal{S}_N$, with training performed on $\mathcal{S}_N$ and $\mathcal{S}$, respectively. It can be observed that training on

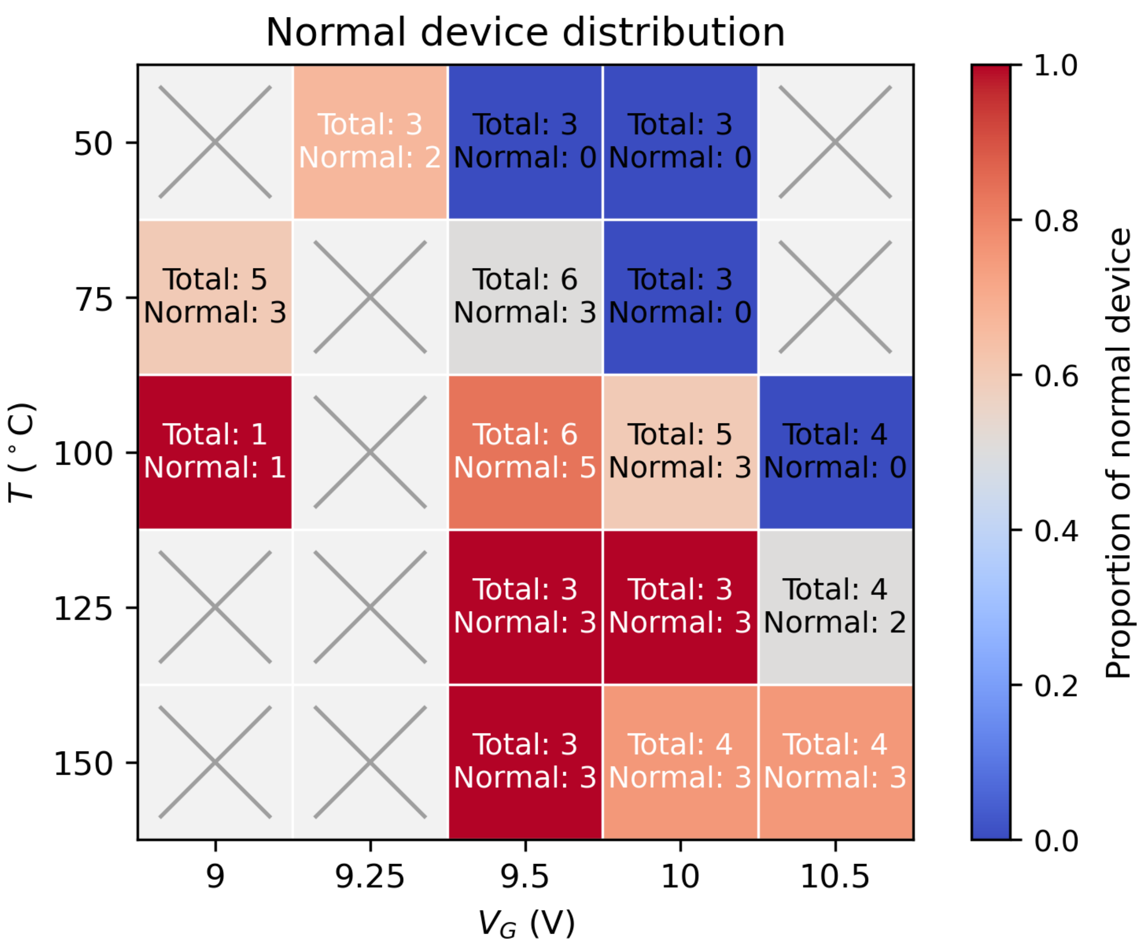
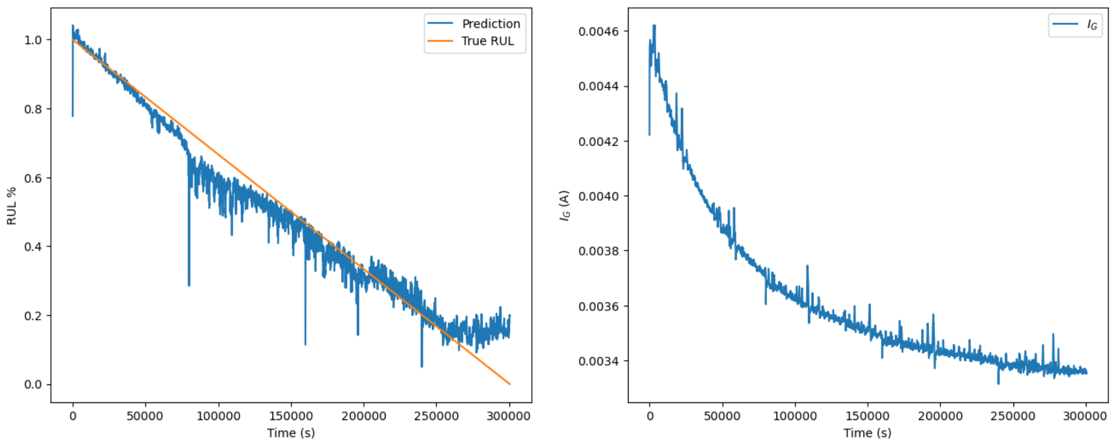


Figure 4.2: Distribution of total devices and normal devices under different gate voltages V_G and temperatures T. Each cell represents a group of devices under the same V_G and the same T, showing both the total number of devices and the number of normal devices. The color of the cell indicates the proportion of normal devices in this group. Conditions without any tested devices are marked with a cross.

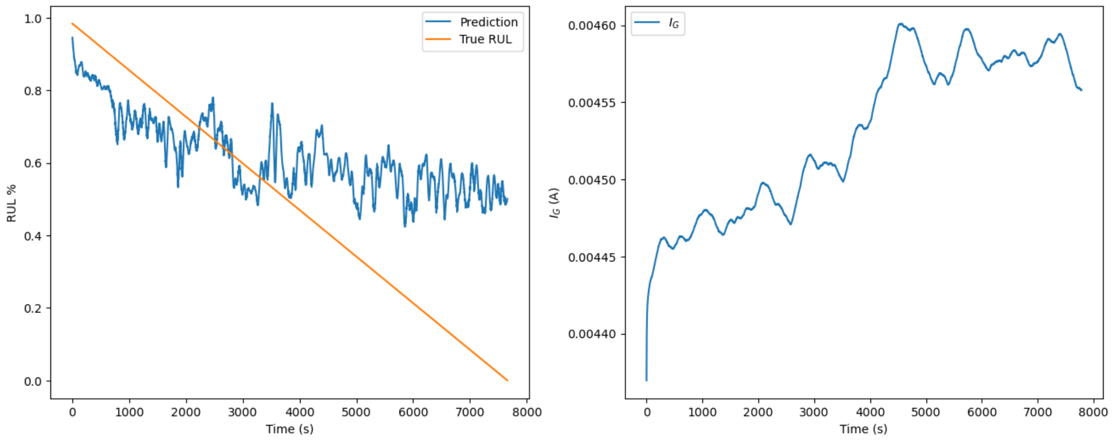
the smaller subset $\mathcal{S}_N$ yields better predictive accuracy, indicating that the presence of anomalous data $\mathcal{S}_A$ in $\mathcal{S}$ does not enhance performance but instead introduces noise.

Rows 2,3, and 4 in the table are trained on the complete dataset $\mathcal{S}$, and the average performance is reported on test sets $\mathcal{S}_N$, $\mathcal{S}_A$, and $\mathcal{S}$, respectively. The results on $\mathcal{S}_N$, discussed earlier, show slightly inferior performance compared with models trained solely on $\mathcal{S}_N$. In contrast, testing on $\mathcal{S}_A$ produces substantial prediction errors, with a PCC of 0.426, reflecting a very weak linear correlation. This finding is consistent with Figure 4.2, where the model fails to provide meaningful predictions for anomalous samples. Row 4 presents the mixed prediction results on the entire dataset $\mathcal{S}$, where the overall error increases and the PCC decreases significantly.

These findings suggest that the underlying degradation mechanisms differ significantly depending on operating environments. In conclusion, prediction on anomalous



(a) Prediction on a normal device with corresponding I_G.



(b) Prediction on an anomalous device with corresponding I_G.

Figure 4.3: Examples of normal and anomalous devices.

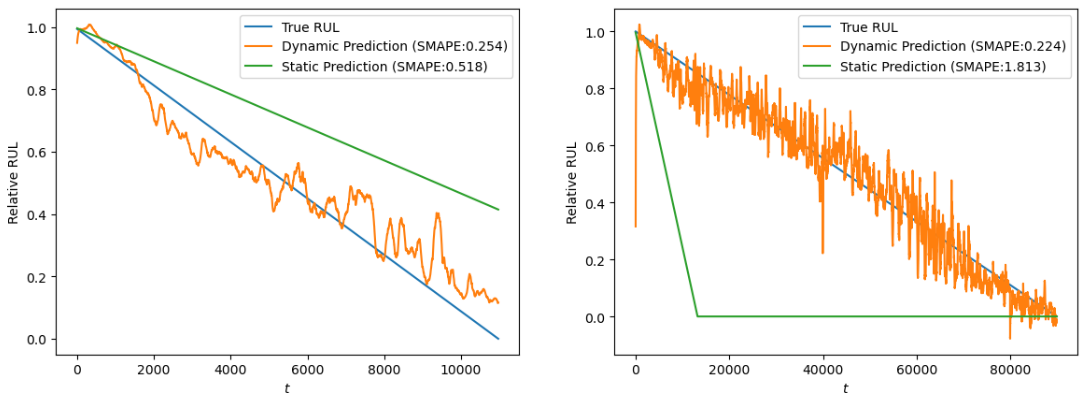
Training set	Test set	NRMSE	SMAPE	PCC
$\mathcal{S}_N$	$\mathcal{S}_N$	**0.0825**	**0.247**	**0.949**
$\mathcal{S}$	$\mathcal{S}_N$	0.1118	0.306	0.932
$\mathcal{S}$	$\mathcal{S}_A$	0.2898	0.508	0.426
$\mathcal{S}$	$\mathcal{S}$	0.1168	0.313	0.920

Table 4.4: Distribution of error among different splits of training sets and test sets.

devices is infeasible and yields only random results, leading to inaccurate error analysis and misrepresenting the model's true performance. Moreover, including anomalous devices in the training set introduces additional noise, resulting in degraded model performance. Therefore, this thesis will focus exclusively on the class of 33 normal devices that enable meaningful RUL prediction.

4.2 Results of Static Prediction

Figure 4.4 shows two kinds of common results in static prediction. Static prediction relies solely on the initial measurements and does not fully exploit the informative signals available during operation. The results of static prediction are based on a Gaussian Process Regression (GPR) model. Due to the limited information content and potential bias in the initial readings, the model often fails to provide accurate lifetime estimates. Therefore, static prediction exhibits larger errors, including SMAPE and other metrics. Although static prediction yields exactly accurate results with errors lower than SMAPE in rare cases, considering the risk of misjudging device lifetime, it is still not wise to entirely rely on static prediction. The limitations of static prediction validate the necessity of adopting dynamic prediction, which leverages information from measurements in operation.



(a) Prediction on Device 14, in which the predicted lifetime based static prediction is longer than the true lifetime.

(b) Prediction on Device 31, in which the predicted lifetime based static prediction is shorter than the true lifetime.

Figure 4.4: Comparison between dynamic prediction and static prediction.

4.3 Results of Relative RUL Prediction

Model	MSE	NRMSE	MAE	SMAPE	PCC
TCN+CBAM	**0.00944**	**0.0825**	**0.0643**	0.2472	0.9486
TCN+IAT	0.00949	0.0829	0.0648	**0.2466**	**0.9488**
TCN+Transformer	0.01174	0.0953	0.0742	0.2796	0.9430
TCN	0.01080	0.0897	0.0686	0.2599	0.9434
CNN	0.01315	0.1003	0.0782	0.2840	0.9390
LSTM	0.01806	0.1117	0.0831	0.2962	0.9165
TCN+CBAM (anomalies incl.)	0.01606	0.1118	0.0829	0.3060	0.932

Table 4.5: Comparison of errors in relative RUL prediction across methods.

Table 4.5 presents the quantitative evaluation of different models on the relative

RUL prediction task using multiple metrics. TCN+CBAM is the model we proposed by stacking Attention-Augmented TCN blocks. TCN+IAT is the model in which stochastic pooling is added to the attention mechanisms. TCN+Transformer consists of 2 components. The first components are a few vanilla TCN blocks without attachments. The second part consists of self-attention blocks that follow the stacked TCN. The model trained on all tested devices $\mathcal{S}$, including anomalous devices, is marked as TCN+CBAM (anomalies incl.). A CNN and an LSTM are also tested as baselines. Among all models, the proposed TCN+CBAM model achieves the best performance overall with the lowest errors in MSE, NRMSE, and MAE. Though its adaptive version, TCN+IAT, outperforms, their scores are very close, which means they have almost the same performance in SMAPE and PCC.

Compared to the baseline CNN model, all proposed TCN-based models achieve notable improvements in prediction performance, demonstrating the superiority of the modified TCN architecture.

However, TCN+IAT, the variant of TCN+CBAM, does not achieve noticeable improvements on CBAM, which suggests that the integration of the newly introduced structure does not lead to significant further gains. Instead, the two models share similar SMAPE and PCC, while TCN+CBAM still maintains its advantage in MSE, NRMSE, and MAE after considering randomness. This suggests that the stochastic pooling branch in IAT fails to introduce additional informative signals. One possible explanation is that specific dominant channels contribute the majority of the valuable information, making the combination of max and average pooling sufficient. Therefore, the introduction of stochastic pooling might bring more noise than new information. Moreover, the performance may have already approached its upper bound, and the limited improvements introduced by IAT might be insufficient to surpass the capabilities of its basis architecture. It is also worth noting that attention mechanisms are primarily designed to address issues of feature selection or long-range dependencies. Incorporating IAT may increase the model complexity without yielding significant performance gains.

Although the Transformer architecture has shown remarkable success in fields such as large language models (LLMs), its application to this task does not yield a significant performance improvement. On the contrary, incorporating a Transformer into the TCN model results in degraded performance. Experimental results show that its performance is inferior to that of the CNN model, and only slightly better than the baseline CNN. The reasons can be attributed as follows: First, the Transformer requires significantly more parameters compared to lightweight modules such as CBAM and IAT used in the proposed network. Although overlapping slicing was employed during preprocessing to generate a relatively large training set, it may still be insufficient for effectively optimizing a Transformer-based architecture. Moreover, the data distribution is inherently constrained by the limited number of tested HEMT devices, which may cause overfitting when using a high-capacity model like the Transformer. Second, RUL prediction relies heavily on local degradation trends. TCN, with its dilated convolution structure, inherently captures local temporal patterns and inductive biases that suit this task well. In contrast, the Transformer's global self-attention mechanism may dilute important local signals, especially when operating on limited data. Additionally, the integration design may not have been optimal for this specific task. Without properly embedding temporal information or preserving local sequential order, the Transformer module might have disrupted rather than enhanced the feature extraction process. Therefore, these findings suggest that lightweight, convolution-friendly attention mechanisms, such as CBAM or IAT, are better suited attention mechanisms for modeling localized degradation behavior.

The model based on TCN serves only as an ablation group for the proposed model. Both added lightweight mechanisms, TCN and IAT, surpass the plain TCN in all evaluation metrics. These results validate that the integration of channel-wise and temporal attention allows the model to better focus on informative features by refining specific locations and suppressing irrelevant variations of the features extracted by TCN.

Due to the use of short fixed-length slices as input to the model, the long-term temporal dependencies that LSTM is designed to capture may not be fully utilized. However, expanding the sequence length fails to improve the prediction performance of LSTM, because errors caused by noise are accumulated during the recursive updating process of LSTM. Figure 4.13 shows the results of such noise accumulation, in which the prediction results of LSTM have higher noise than other tested models. In the case of Device 41, the predictions of LSTM even saturate, further highlighting the sensitivity to noise of LSTM. In contrast, CNNs are more effective at extracting local patterns within limited contexts, making them better suited for short sequence inputs. The characteristics of CNN may explain why CNN outperforms LSTM in these experiments.

Additionally, complete results of the model trained on all devices TCN+CBAM (anomalies incl.) are shown in Table 4.5. The resulting performance degrades across all metrics, confirming that the presence of outliers negatively impacts model behavior, which supports the effectiveness of the preprocessing step involving anomaly removal.

Figure 4.5 illustrates the central tendency and the dispersion of tested errors, including NRMSE and SMAPE, on groups of different operational environments among different models. In comparison with the baseline CNN, the proposed models exhibit a noticeable downward shift in both the box range and the locations of the median and mean, indicating improved prediction performance. In general, it further confirms the previous conclusion that the TCN adaptations contribute to improvements in prediction performance and that the introduction of the attention mechanism provides additional enhancement. Besides, it is noticeable that most models exhibit a mean value of errors lower than the median for NRMSE, indicating a left-skewed error distribution. This suggests that the model performs consistently across most groups and generates outstandingly accurate predictions in some cases, resulting in a low average error. In contrast, SMAPE does not consistently follow such a pattern. A potential reason is the sensitivity of SMAPE during the phase close to device failure, when RUL is a very small value, leading to an inflation of error. Moreover, it is observed that training the model on $\mathcal{S}$ not only increases the prediction error, but also results in a substantial increase in variance. This further motivates the need for careful sample selection.

To better understand model behavior under varying degradation conditions, we visualize RUL prediction results for several representative devices among the tested

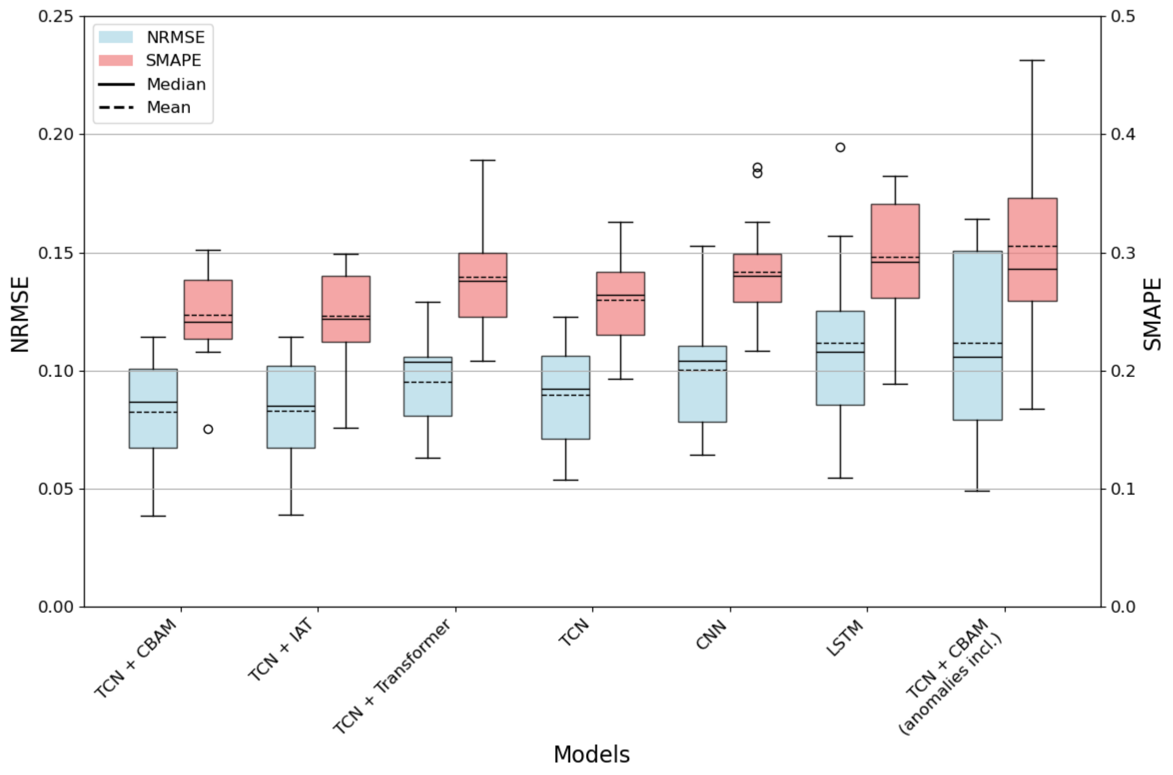


Figure 4.5: Boxplots of prediction errors (NRMSE and SMAPE) for different models. Each data point used in the boxplot is the average error of a certain group under the same T and V_G.

devices. For clarity of description, the devices are categorized into three classes based on the order of magnitude of their lifetimes: $L < 10,000\,\text{s}$, $10,000\,\text{s} < L < 100,000\,\text{s}$, and $L > 100,000\,\text{s}$. Figure 4.6, Figure 4.7 and Figure 4.8 show the results of these 3 classes respectively.

For the device with $L < 10,000\,\text{s}$, which is shown in Figure 4.6, prediction results exhibit a relatively smooth variation curve. The TCN-based model demonstrates improved alignment with the true RUL curve compared to baseline models, CNN, and LSTM. Based on the TCN model, the introduction of attention mechanisms further enhances the model's representational capacity. As a result, the prediction curves of TCN+CBAM and TCN+IAT are highly consistent with the ground truth. However, a defect is observed in the final stage of operation, where the predicted RUL does not drop to zero. This phenomenon is common across multiple GaN HEMTs. A potential explanation is the increased prediction uncertainty near the end of life, which will be further discussed in section 4.4. Moreover, the performance of the TCN+CBAM model trained on the entire dataset is consistent with the earlier discussion, exhibiting a noticeable deviation from the ground truth.

In the case of $10,000\,\text{s} < L < 100,000\,\text{s}$, which is shown in Figure 4.6, the noise in the prediction results becomes significantly pronounced at this time scale, whereas the proposed model demonstrates strong noise robustness. Figure 4.7(a) shows that

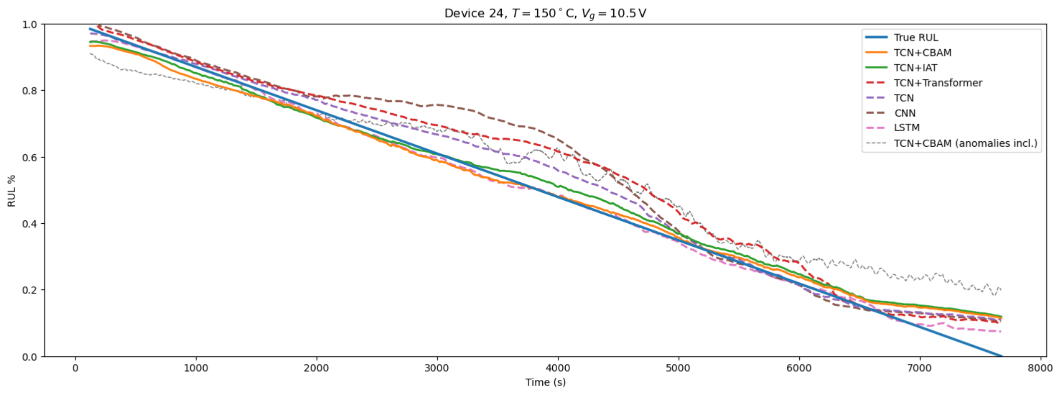


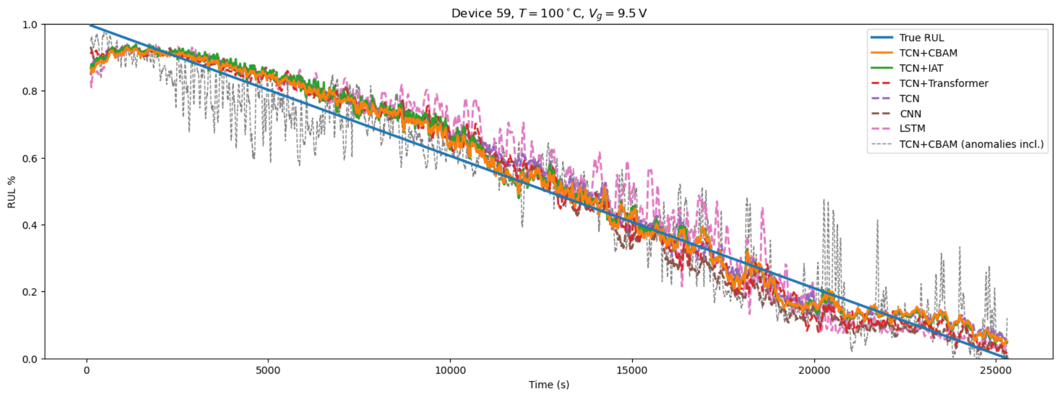
Figure 4.6: Relative RUL prediction on devices with lifetime $L < 10,000\,\mathrm{s}$, in which models predict smooth results.

TCN models integrated with an attention mechanism exhibit the highest consistency with the label, maintaining a stable trend throughout all phases of the degradation process. Their prediction curves are relatively smooth, without significant fluctuations or systematic deviations. In contrast, the baseline CNN model produces locally unstable predictions with occasional abrupt jumps. It also lacks the ability to model long-term temporal dependencies. While the TCN performs more stably than CNN, it still shows greater fluctuations compared to its attention-enhanced variants, indicating limited capacity for fine-grained feature extraction. Besides, LSTM and TCN+CBAM (anomalies included) demonstrate a weaker capability to capture the degradation behavior accurately.

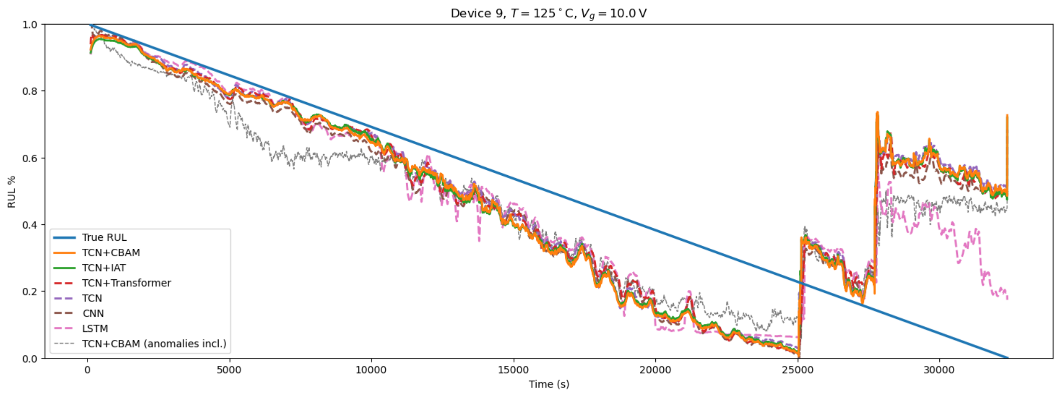
Figure 4.7(b) shows another defect that happens in a few samples of this scale. The predicted RUL drops to zero prematurely, followed by an abrupt increase to a relatively high value, after which the prediction resumes its degradation trend. This phenomenon results in the proposed model failing to outperform the others on these devices, though it has an advantage before the abrupt point. As mentioned subsection 4.1.4, the prediction results are strongly related to I_G, and this phenomenon is caused by a sudden jump in I_G. The underlying reason may be that the actual degradation process of such devices differs from that of other HEMTs. However, due to the limited number of devices, we did not perform further categorization. This issue requires more in-depth investigation into the degradation behavior of GaN HEMTs.

From the perspective of devices with $L > 100,000\,\mathrm{s}$, which is shown in Figure 4.8, the noise becomes further amplified, increasing the requirement for the model to track the main degradation trend under disturbances. Although the proposed model also suffers from noise like comparison in Figure 4.8(a), it exhibits the best noise suppression capability among all models. In the end, TCN+CBAM and TCN+IAT converge to zeros as the noise gradually diminishes in the final stages.

While in Figure 4.8(b), the prediction shows a decelerating degradation trend, rather than a linear degradation trend. This phenomenon can be attributed to both the intrinsic behavior I_G, which is strongly related to prediction, and the use of logarithmic time $t_{\log}$. Specifically, I_G itself exhibits a decelerating degradation trend, and due



(a) Prediction on Device 59, in which prediction exhibits noise.



(b) Prediction on Device 9, in which an abrupt increase occured in the later stage.

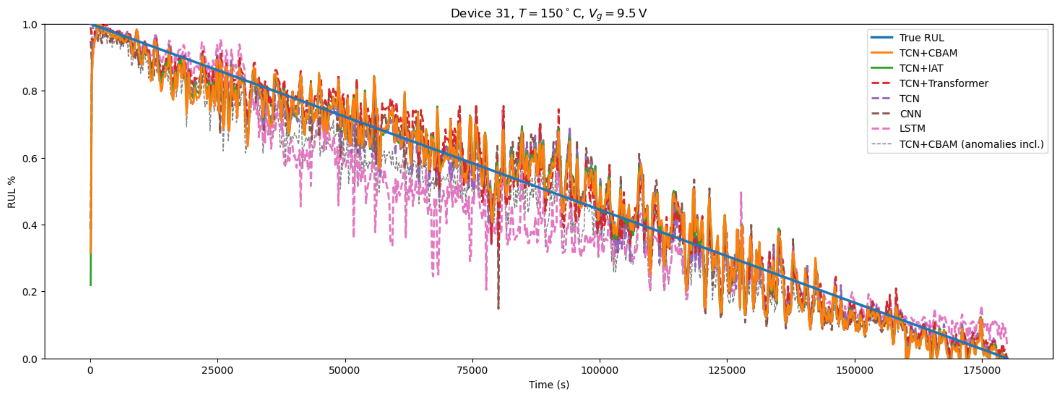
Figure 4.7: Relative RUL prediction on devices with lifetime $10,000\,\mathrm{s} < L < 100,000\,\mathrm{s}$.

to the large time scale of operation, the logarithmic transformation of time becomes less sensitive to time increments in the late stages, which further contributes to the flattening of the predicted degradation curve.

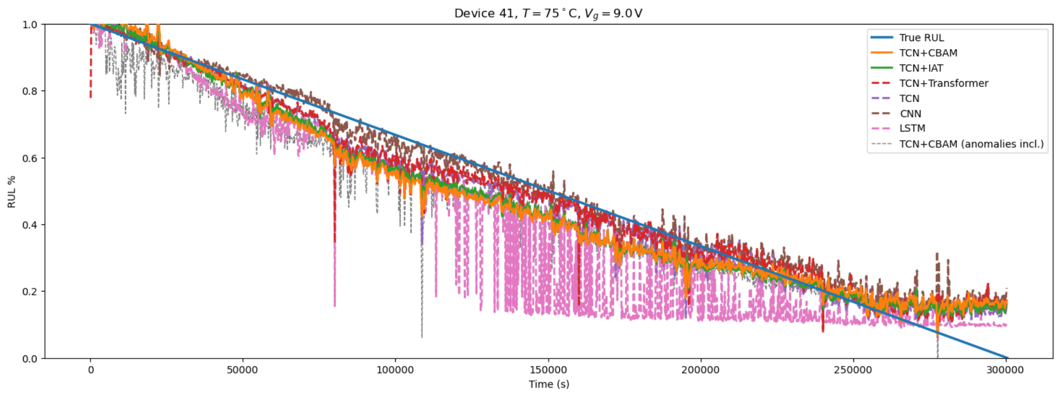
Figure 4.9 illustrates the prediction performance of the proposed TCN+CBAM model across 33 different test devices under varying combinations of gate voltage (V_G) and temperature (T). The results are sorted based on groups. The blue line represents the true relative RUL, which is assumed to decrease linearly, while the orange line denotes the predicted RUL generated by the prediction model.

In general, the model demonstrates strong generalization capabilities across diverse operational conditions. In most devices, the predicted RUL curve closely follows the linear degradation trend of the ground truth. For example, devices operating at moderate stress levels, e.g., Device 10, exhibit high prediction fidelity, with smooth and monotonic predictions, as well as minimal deviation from the true RUL.

In some noisy cases, such as Device 42, Device 31, or Device 66, the predicted RUL shows increased variance, which reflects the challenge of learning accurate degradation signals under rapidly deteriorating or irregular sensor measurements. Nonetheless, even in these cases, the model successfully captures the overall degradation trend, maintaining a reasonably accurate shape and convergence.



(a) Prediction on Device 31, in which prediction exhibits strong noise.



(b) Prediction on Device 41, in which prediction exhibits a decelerating degradation trend.

Figure 4.8: Relative RUL prediction on devices with lifetime $L > 100,000\,\text{s}$.

Furthermore, the model shows robustness to varying the lifetime of the device. No matter shorter sequences like Device 15 and Device 17 or longer sequences like Device 10, the model still manages to generate stable and interpretable predictions without abrupt failure in performance. This indicates that the TCN backbone, along with the attention mechanism introduced by CBAM, contributes to effective temporal representation and adaptability to diverse time series.

Another notable observation is the model's ability to handle slight prediction lag or lead at early time steps. While some predicted curves start slightly below or above the ground truth, like what happened in Device 3 and Device 22, the difference is small and gradually converges over time, showing the model's capacity for self-correction during progressive degradation.

However, certain devices exhibit prediction defects at the end of life, such as failure to converge to zero (Device 6) and abrupt fluctuations (Device 9) in the predicted RUL, which can be observed in Figure 4.9. In Device 6, though the predictions are close to the true values during the early and middle stages of operation, they deviate in the later stage. The predicted degradation slows down and fails to converge to zero. In Device 9, the predicted values reach zero earlier than the ground truth. However, they subsequently increase abruptly to a higher level. Then, decaying continues from the high value as the new starting point. The same sharp increase occurs again, but before reaching zero, and the decay continues after the second increase. A possible reason is that the breakdown occurs when the predicted value first reaches zero, but this does not immediately lead to device failure. Rather, the device continues to operate in a new state for minutes until the final failure. These prediction anomalies may be attributed to irregular sensor behaviors and limitations of the data. Further investigation, including research about the physical status of devices and monotonic constraints of the prediction model, is required to fully understand these cases and solve them in a general architecture.

To sum up, the results of relative RUL prediction have validated the following:

- Transition from vanilla CNN to TCN enhances the capability of extracting temporal information from time series data, while the integration of attention mechanisms in the CBAM and IAT modules further improves the performance. On the other hand, integration of the Transformer into the TCN framework leads to a higher risk of overfitting, thereby degrading the overall performance.
- Although static prediction can accurately estimate the device lifetime in a few cases, it generally fails to provide reliable results due to the limited information available in the initial values. In contrast, dynamic prediction methods can make more reliable estimates by continuously updating predictions based on information gathered during operation.
- Anomalous devices cannot be effectively predicted with the available information and should be excluded from model training to avoid potential interference.

The results also highlight several potential directions for future research, including solving convergence issues and abrupt jumps in the late stage, as well as methods to detect and alternative approaches for predicting anomalous devices.

4.4 Uncertainty Prediction via Gaussian Negative Log-Likelihood

In section 4.3, it is hypothesized that convergence failure during the late stage could be caused by increased prediction uncertainty. Besides, we also want to further research the reliability and interpretability of RUL predictions. Therefore, uncertainty estimation is incorporated into the proposed TCN+CBAM model.

In this study, Gaussian Negative Log-Likelihood (GNLL) is adopted to model aleatoric uncertainty, which arises from inherent noise in sensor measurements and operating conditions. GNLL is defined as

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{(y_i - \hat{y}_i)^2}{2\sigma_i^2} + \frac{1}{2}\log\sigma_i^2\right). \tag{4.7}$$

Both $\hat{y}_i$, which represents the mean of prediction, and σ_i, which represents the standard deviation of prediction, are the outputs of the predictive model. In practice, as the

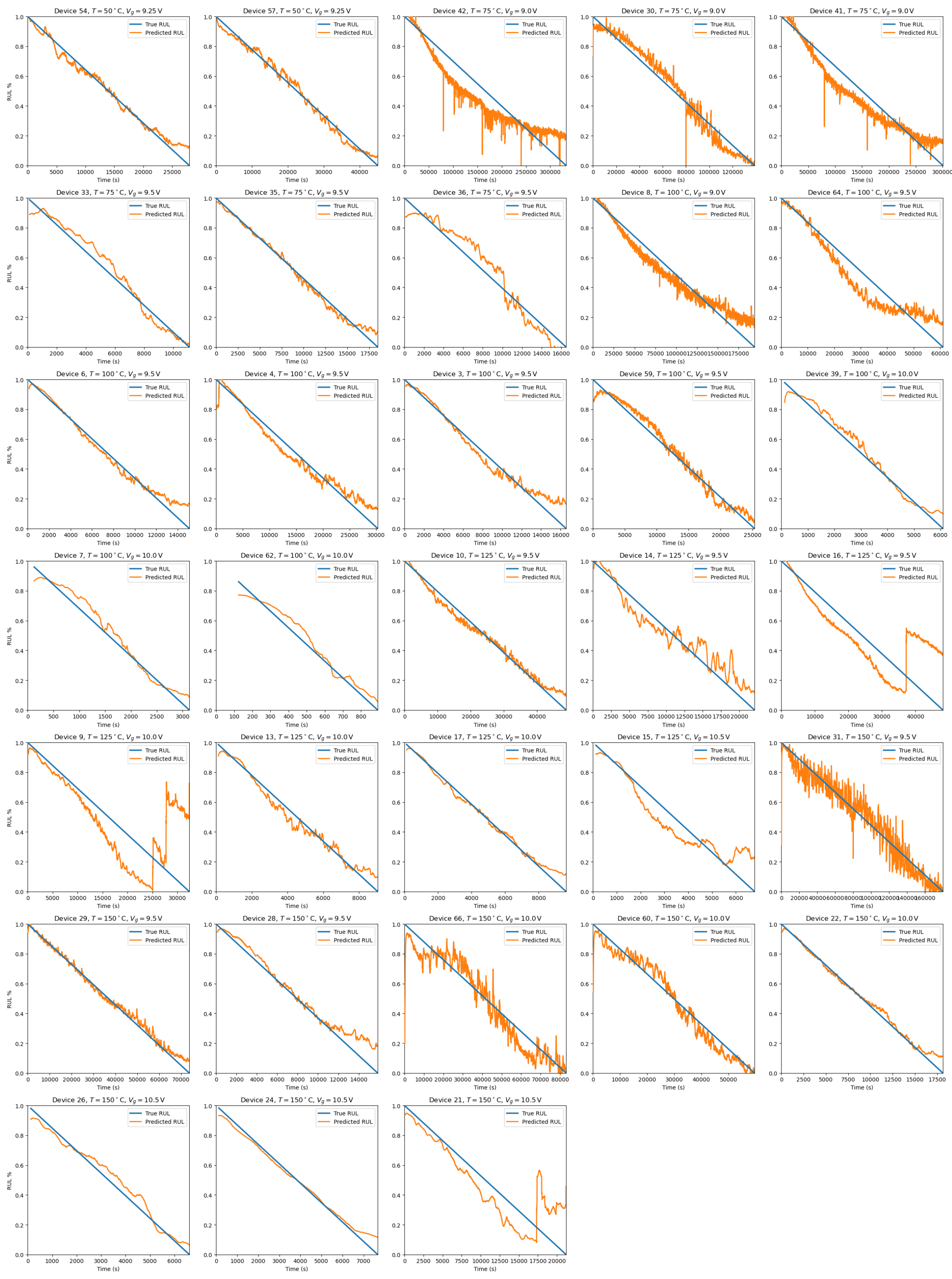


Figure 4.9: Relative RUL prediction results on all tested devices.

regular regression model provides unbounded outputs, the model is configured to predict $s_i = \log \sigma_i^2$, rather than the non-negative value σ_i. Therefore, Equation 4.7 can be reformulated as

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{(y_i - \hat{y}_i)^2}{2 \exp(s_i)} + \frac{1}{2} s_i \right) \tag{4.8}$$

Unlike MSE used in previous sections, GNLL enables the model to learn not only a point estimate but also the confidence associated with each prediction by assuming that the target variable follows a Gaussian distribution and trains the model to predict both the mean and variance of this distribution. Equation 4.7 allows the model to express uncertainty as heteroscedastic that varies across different inputs. By minimizing GNLL, the model is encouraged to assign a higher σ to predictions where the residual error is high, effectively reducing the risk of overconfident predictions on noisy or ambiguous samples. Application of GNLL provides a more robust and informative prediction output, which is critical for the application areas of GaN HEMT, where reliability has priority.

To evaluate the quality of uncertainty estimation, we add three widely used metrics based on previous metrics, including Prediction Interval Coverage Probability (PICP), Mean Prediction Interval Width (MPIW), and Continuous Ranked Probability Score (CRPS).

PICP quantifies the proportion of ground truth values that fall within the predicted confidence interval, and is calculated as

$$\mathrm{PICP} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I} \left[y_i \in \left[\hat{y}_i^L, \hat{y}_i^U \right] \right], \tag{4.9}$$

where $\hat{y}_i^L$ and $\hat{y}_i^U$ are the lower and upper bounds of the prediction interval for the i-th sample. In the experiments, a 95% uncertainty interval is selected, assuming a Gaussian distribution of the prediction errors.

MPIW measures the average width of the prediction intervals and is defined as

$$\mathrm{MPIW} = \frac{1}{N} \sum_{i=1}^{N} \left(\hat{y}_i^U - \hat{y}_i^L \right). \tag{4.10}$$

A narrow MPIW generally implies more confident predictions. If the prediction interval is narrow but fails to cover the true values, it may indicate overconfidence and a lack of reliability in uncertainty estimation. A wide MPIW typically reflects great uncertainty in the predictions, which means less informative or overly conservative predictions, reducing the practical utility of the model.

CRPS provides an evaluation of the predicted cumulative distribution against the ground truth and generalizes MAE to probabilistic forecasts. It is computed as

$$\mathrm{CRPS}(F, y) = \int_{-\infty}^{+\infty} \left(F(z) - \mathbb{I}(z \geq y) \right)^2 dz, \tag{4.11}$$

where $F(z)$ denotes the predictive cumulative distribution function. In the context of GNLL, $F(z)$ is the Cumulative Distribution Function (CDF) of the Gaussian distribution $\Phi(\cdot)$. A closed-form expression of CRPS can be formulated as

$$\mathrm{CRPS}(\hat{y}_i, \sigma_i, y_i) = \sigma_i \left[\frac{y_i - \hat{y}_i}{\sigma_i} \left(2\Phi\left(\frac{y_i - \hat{y}_i}{\sigma_i}\right) - 1 \right) + 2\phi\left(\frac{y_i - \hat{y}_i}{\sigma_i}\right) - \frac{1}{\sqrt{\pi}} \right], \quad (4.12)$$

in which $\phi(\cdot)$ is the PDF of the Gaussian distribution. CRPS rewards predictions that are both accurate and well-calibrated, penalizing distributions that are either too narrow (overconfident) or too wide (underconfident). Compared to GNLL itself, which may be sensitive to extreme values or significant variances, CRPS provides a more balanced evaluation of the predictive distribution's shape and coverage, making it especially suitable for evaluating uncertainty-aware RUL prediction models.

Table 4.6 shows the result of uncertainty prediction. Compared to the point-estimation model, the GNLL-based model shows a slightly higher prediction error, with MSE increasing from 0.0094 to 0.0113 and MAE rising from 0.0643 to 0.0663. However, as the model now focuses not only on minimizing the mean error but also on learning the predictive distribution, this performance degradation in accuracy is expected and acceptable. This trade-off is justified by the ability to provide confidence-aware predictions, which are essential for downstream decision-making in reliability-critical applications.

More importantly, the model now provides uncertainty-aware predictions, allowing us to assess its reliability using probabilistic metrics. Specifically, PICP reaches 0.732, indicating that approximately 73% of the ground truth values fall within the predicted confidence intervals, while the ideal value should be 95%. MPIW is 0.185, which suggests that the predicted intervals are not relatively short. Metrics of PICP and MPIW indicate that the model underestimates predictive uncertainty, failing to cover tail deviations caused by long-term, correlated noise. However, CRPS of 0.0526 reflects a reasonable overall quality of the probabilistic forecasts, integrating both the sharpness and calibration of the prediction distribution.

Although the PICP and MPIW appear suboptimal under ideal Gaussian assumptions, the results can be explained when considering the characteristics of the real noise of the measurements. The practical noise is non-Gaussian and exhibits temporal dependencies. A fluctuation of noise in prediction can persist over hundreds of seconds. Therefore, the patterns of noise violate the i.i.d. and Gaussian assumptions underpinning the GNLL-based uncertainty model, resulting in suboptimal performance. Nevertheless, CRPS still shows a positive signal that the overall predictive distribution still maintains reasonable shape and calibration in the central region, reflecting that the model performs reliably in most time steps.

To further interpret the behavior of uncertainty-aware RUL prediction, we visualize representative predictions from three test devices. Figure 4.10 shows the 3 representative uncertainty prediction with 95% confidence intervals. These figures prove the hypothesis of uncertainty increments in the late stage in section 4.3. The uncertainty interval increases significantly during operation, with its width in the late phase reaching several times that of the initial phase.

Figure 4.10(a) illustrates the prediction results of Device 7, in which most parts of

Metrics	Values
MSE	0.0113
NRMSE	0.0875
MAE	0.0663
SMAPE	0.266
Pearson	0.943
PICP	0.732
MPIW	0.185
CRPS	0.0526

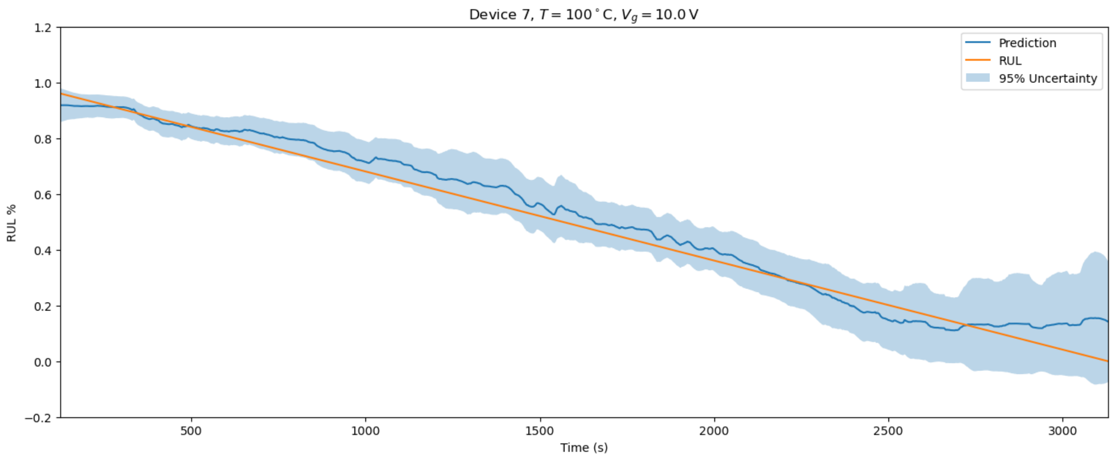
Table 4.6: Results of uncertainty prediction evaluation.

True RUL are covered in the uncertainty interval. In this case, the prediction results are close to the true values, and the prediction has a wide uncertainty interval. Furthermore, the standard deviation expands notably when the predicted value diverges from the actual degradation trend in the late stage. This phenomenon can be explained by the balanced nature of GNLL. When the predicted value deviates from the ground truth, the model increases the predicted uncertainty to compensate for the loss.

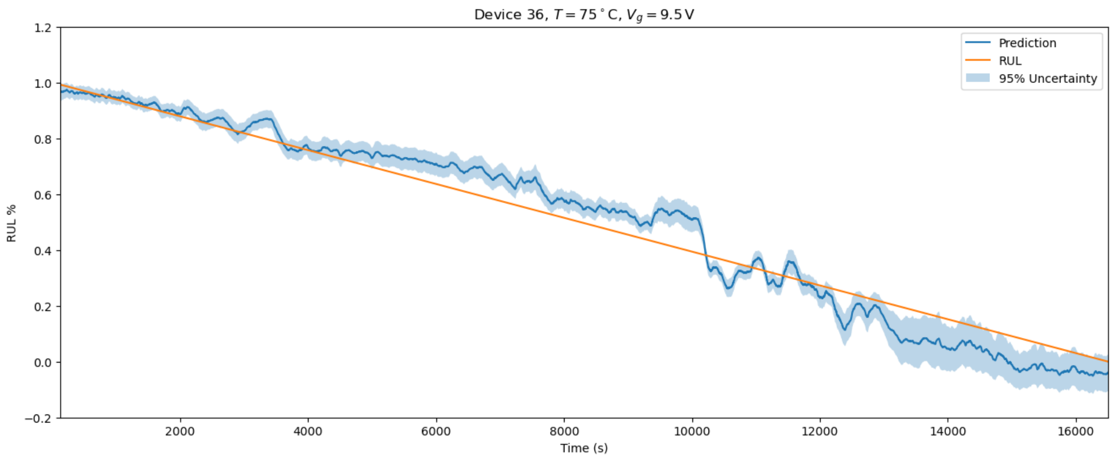
In contrast, Device 36 in Figure 4.10(b) demonstrates a narrow uncertainty interval. Throughout the entire degradation process, the predicted RUL is still close to the ground truth but with a certain distance from it and a few intersections, and the uncertainty band remains relatively tight, indicating a high level of confidence in the model's prediction. Therefore, though errors are not high, this device has a very small PICP. Around $t = 10,000\,\text{s}$, the prediction decreases and crosses the ground truth sharply. As the prediction is getting closer to the true RUL, the model gets more confident and provides a very small uncertainty.

When the noise becomes larger, the pattern of uncertainty will be different. Figure 4.10(c) shows such an example of Device 59. The degradation signal contains strong oscillations and noise throughout the timeline. Correspondingly, the uncertainty band is consistently wider, especially in regions with intense fluctuation. This suggests that the predicted standard deviation adapts to the local signal complexity, capturing model uncertainty caused by volatile patterns or anomalies.

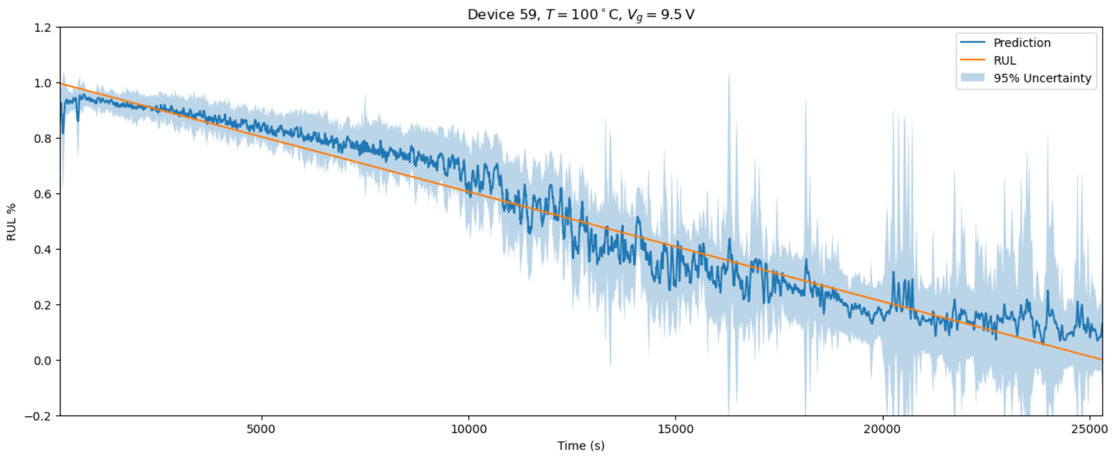
To sum up, uncertainty prediction introduces a trade-off between a slight increase in prediction error and the ability to quantify prediction uncertainty. This allows for a more comprehensive evaluation of model performance in realistic device-level RUL prediction scenarios. Furthermore, the analysis of prediction results supports the earlier hypothesis that uncertainty tends to increase in the later stages of degradation. However, due to variations in data distribution across devices, the model exhibits different levels of under- or over-confidence. This highlights the need for further design improvements, as well as adjustments in model architecture and hyperparameters, to achieve a more generalized solution that performs consistently across most devices.



(a) Device in which the majority of the ground truth is encompassed within the uncertainty interval.



(b) Device in which the majority of the ground truth is encompassed outside the uncertainty interval.



(c) Device in which the uncertainty is heavily affected by noise.

Figure 4.10: Visualization of prediction results and their uncertainty intervals.

4.5 Results of Total Lifetime Estimation and Absolute RUL Reconstruction

Relative RUL $\hat{y}^*$ has been estimated in section 4.3, which functions as a percentage of lifetime left. This section will return to this topic and show the result of the estimated total lifetime $\hat{L}$ and reconstructed absolute RUL $\hat{y}$.

Several practical adjustments are made in implementation, compared to the theoretical formulation. Since the predicted relative RUL values may fall outside the $[0, 1]$ range, we first apply truncation to ensure that all predicted values $\hat{y}$ lie within this interval. To avoid division-by-zero issues during the early stages of prediction, a small constant $\varepsilon = 0.01$ is added to the denominator during the reconstruction process. That gives us the actual step to estimate $\hat{L}$

$$\hat{L}(t) = \frac{t}{1 - \hat{y}^*(t) + \varepsilon}. \tag{4.13}$$

As a fixed term is added to the denominator when calculating $\hat{L}$, if we put the result $\hat{L}$ in Equation 3.38, errors will be accumulated. Therefore, it is always to use $\hat{y}^*$ to reconstruct $\hat{y}$, which means

$$\hat{y}(t) = \frac{t\hat{y}^*(t)}{1 - \hat{y}^*(t) + \varepsilon} \tag{4.14}$$

Model	$\hat{y}$			$\hat{L}$	
	NRMSE	SMAPE	PCC	MSE	MAE
TCN+CBAM	**0.236**	**0.416**	**0.691**	**0.0589**	**0.166**
TCN+IAT	0.240	0.418	0.684	0.0608	0.168
TCN+Transformer	-	0.445	0.627	-	0.186
TCN	0.239	0.421	0.669	0.0598	0.166
CNN	0.281	0.462	0.598	0.0902	0.189
LSTM	0.278	0.455	0.584	0.0895	0.194

Table 4.7: Results of Absolute RUL and Estimated Total Lifetime

The results of reconstructed $\hat{L}$ and $\hat{y}^*$ are listed in Table 4.7. Given the differences in temporal scales among devices, NRMSE, SMAPE, and PCC were adopted to assess the prediction performance of $\hat{y}$. As true L remains constant across samples, normalization was applied to both the estimated values $\hat{L}$ and L in calculating MSE and MAE.

The overall conclusion is similar to what was previously observed in relative RUL prediction. Compared with vanilla CNN, TCN has better performance in all metrics. Both TCN+CBAM and TCN+IAT achieve the lowest values across most error metrics, except for PCC, indicating superior accuracy in RUL prediction and validating the improvements brought by the attention mechanism. However, the reconstructed results reveal many new and distinct insights.

Due to the overfitting issue of the Transformer model, it continuously outputs a value of 1 during the initial stage for certain samples. As a result, the prediction error is significantly amplified in the division. This issue is further exacerbated when computing

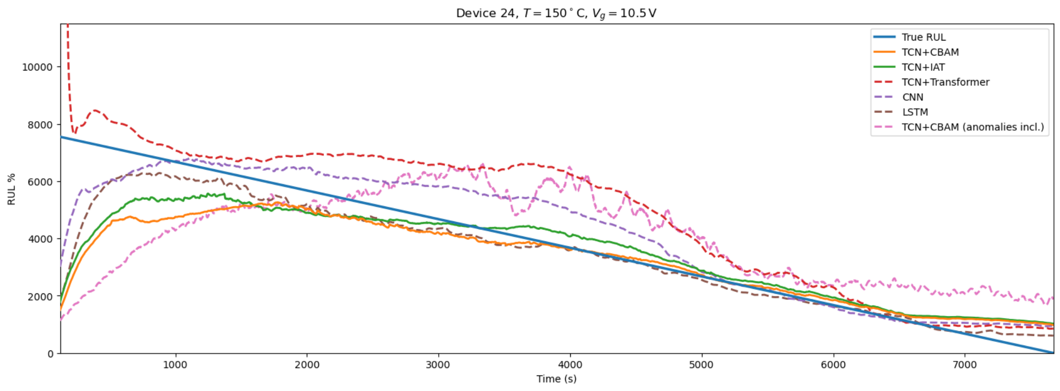


Figure 4.11: Absolute RUL Prediction on device with Lifetime $L < 10,000\,\mathrm{s}$, in which models predict smooth results.

squared error metrics such as MSE and NRMSE, leading to abnormal NRMSE of $\hat{y}$ and MSE of $\hat{L}$. Therefore, the 2 values are not presented. This result further supports the earlier conclusion that the limited dataset used in this thesis is still insufficient to adequately train a Transformer-based model, resulting in performance degradation of outputs and fatal errors in reconstruction.

Moreover, in the relative RUL prediction task, LSTM performed noticeably worse than the other baseline, CNN. However, in the reconstruction-based evaluation, LSTM showed slightly better performance overall, with only PCC and MAE being inferior to CNN. This is likely because the reconstruction approach amplifies errors in the early stage of operation, during which both models exhibit similar behavior. As a result, their overall performance appears quite comparable.

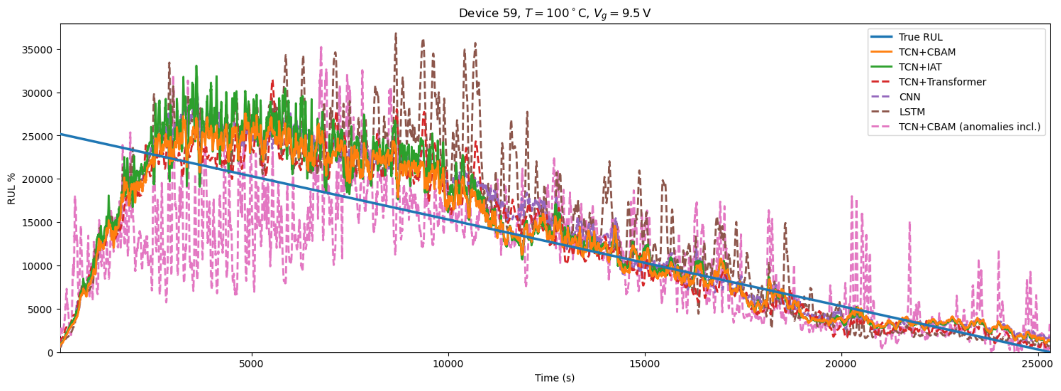
Figure 4.11, Figure 4.12 and Figure 4.13 and the reconstructed results respectively corresponding to Figure 4.6, Figure 4.7 and Figure 4.8.

Figure 4.11 is the reconstructed RUL of a device with lifetime $L < 10,000\,\mathrm{s}$. The proposed model exhibits a relatively slow start-up behavior, which means the predicted relative RUL is smaller than the ground truth in the initial stage. However, the proposed model quickly catches up with the ground truth and maintains close to the true RUL, indicating reasonable tracking performance and validating its superiority. This figure also explains the failure of TCN+LSTM models in NRMSE and MSE, as it gives a high estimation several times larger than the ground truth.

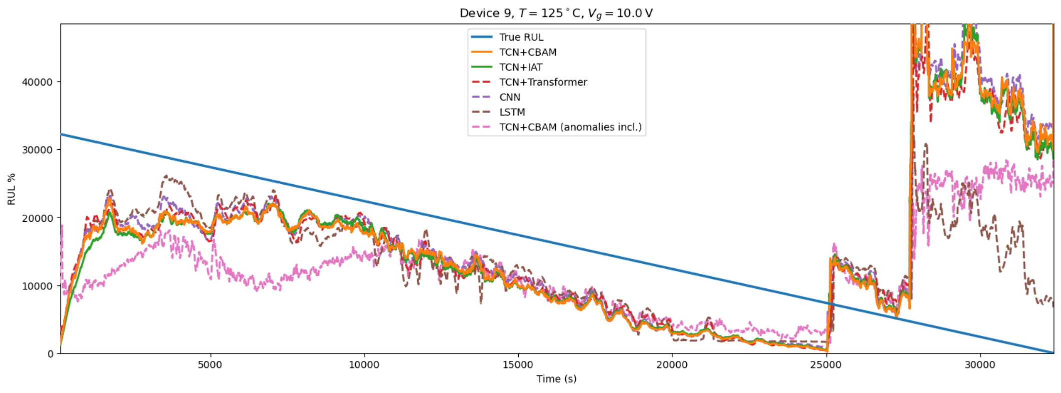
Figure 4.12 shows the reconstructed RUL of a device with lifetime $10,000\,\mathrm{s} < L < 100,000\,\mathrm{s}$. Figure 4.12(a) shows that the noise is amplified by the term in the denominator. In the beginning, when t is small and $\hat{y}^*$ is close to 1. As the operation continues, $\hat{y}^*$ will decrease as t increases. Therefore, fluctuation in the TCN+CBAM and TCN+IAT also decreases during operation. Constrained by its representational capability, the LSTM model remains inadequate in effectively managing noisy input data. Figure 4.12(a) shows the noise is amplified when t

Figure 4.13 illustrates the reconstructed RUL of a device with lifetime $L > 100,000\,\mathrm{s}$. Similarly, predictions also show a noise decrease during operation.

Figure 4.15 and Figure 4.14 shows the estimated $\hat{L}$ and reconstructed $\hat{y}$ among all



(a) Prediction on Device 59, in which prediction exhibits noise.



(b) Prediction on Device 9, in which an abrupt increase happened in the later stage.

Figure 4.12: Absolute RUL Prediction on Device with Lifetime $10,000\,\text{s} < L < 100,000\,\text{s}$.

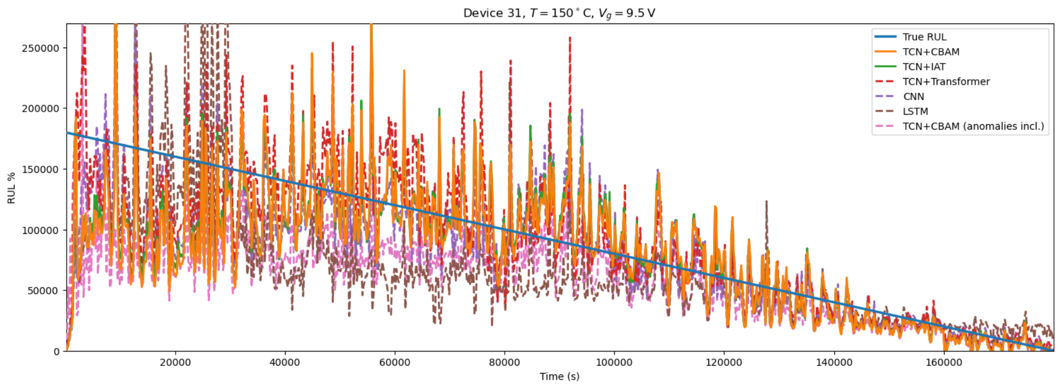
devices based on TCN+CBAM model.

The estimated values of $\hat{L}$ across different devices reveal several typical behaviors.

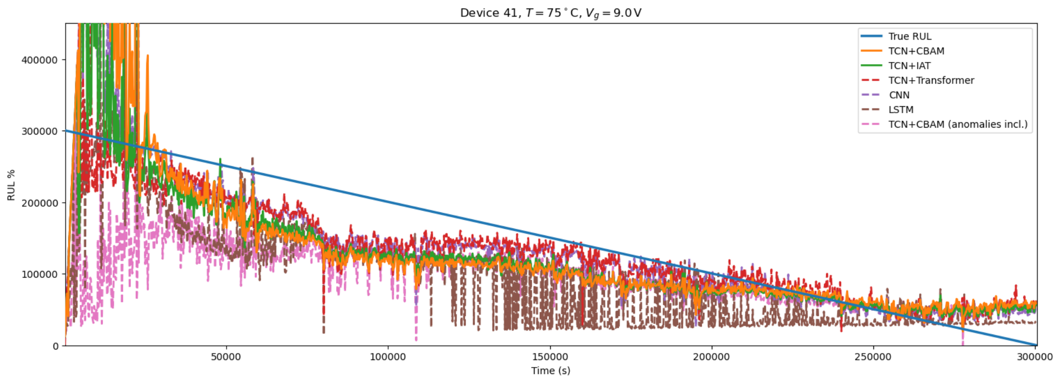
- 1. In many samples, $\hat{L}$ exhibits sharp peaks or rapid increases during the initial stage. This is consistent with the theory that degradation cannot be detected in the beginning. Therefore, the predicted RUL $\hat{y}$ remains nearly constant close to 1 in the initial stage. The estimation of $\hat{L}(t) = t/(\hat{y}(t) + \varepsilon)$ becomes a proportional function of t with a fixed slope, which explains the sharp increase in the beginning.

- 2. During operation, $\hat{L}$ gradually stabilizes as the degradation process progresses. This suggests that once the model identifies a consistent degradation trend, it can provide a reliable estimation of the degradation rate. However, some samples display irregular fluctuations or even rising trends in the later stages, which often correspond to cases where the predicted RUL fails to converge to zero or where data noise dominates. These results indicate that the robustness of $\hat{L}$ estimation is closely tied to the quality and stability of relative RUL prediction.

Although absolute RUL prediction has been achieved effectively, the issues observed in relative RUL prediction and L estimation are similarly reflected in $\hat{y}$. For instance,



(a) Prediction on Device 31, in which prediction exhibits strong noise.



(b) Prediction on Device 41, in which prediction exhibits a decelerating degradation trend.

Figure 4.13: Absolute RUL Prediction on Device with Lifetime $L > 100,000\,$s.

Device 42 shows accelerating degradation, rather than linear degradation. Device 24 fails to converge to 0.

An effective estimator can be a promising solution. For now, all $\hat{L}(t_i)$ and $\hat{y}(t_i)$ are calculated based on the instantaneous value of $\hat{y}^*(t_i)$. An estimator exploiting $\hat{y}(t = t_i) = f(\hat{y}^*(1), \hat{y}^*(2), \ldots, \hat{y}^*(t_i))$ might suppress fluctuations while preserving the overall degradation trend.

To sum up, the results of absolute RUL prediction and total lifetime can be concluded as follows:

- Proposed TCN+CBAM and TCN+IAT models realize the complete pipeline of RUL prediction from relative RUL to absolute RUL and outperform comparisons.
- Features of the estimated total lifetime and absolute RUL are explored and explained based on concrete theory and experimental results. Potential solutions for current issues are proposed.

The results also highlight several potential directions for future research, including solving convergence issues and abrupt jumps in the late stage, as well as methods to detect and alternative approaches for predicting anomalous devices.

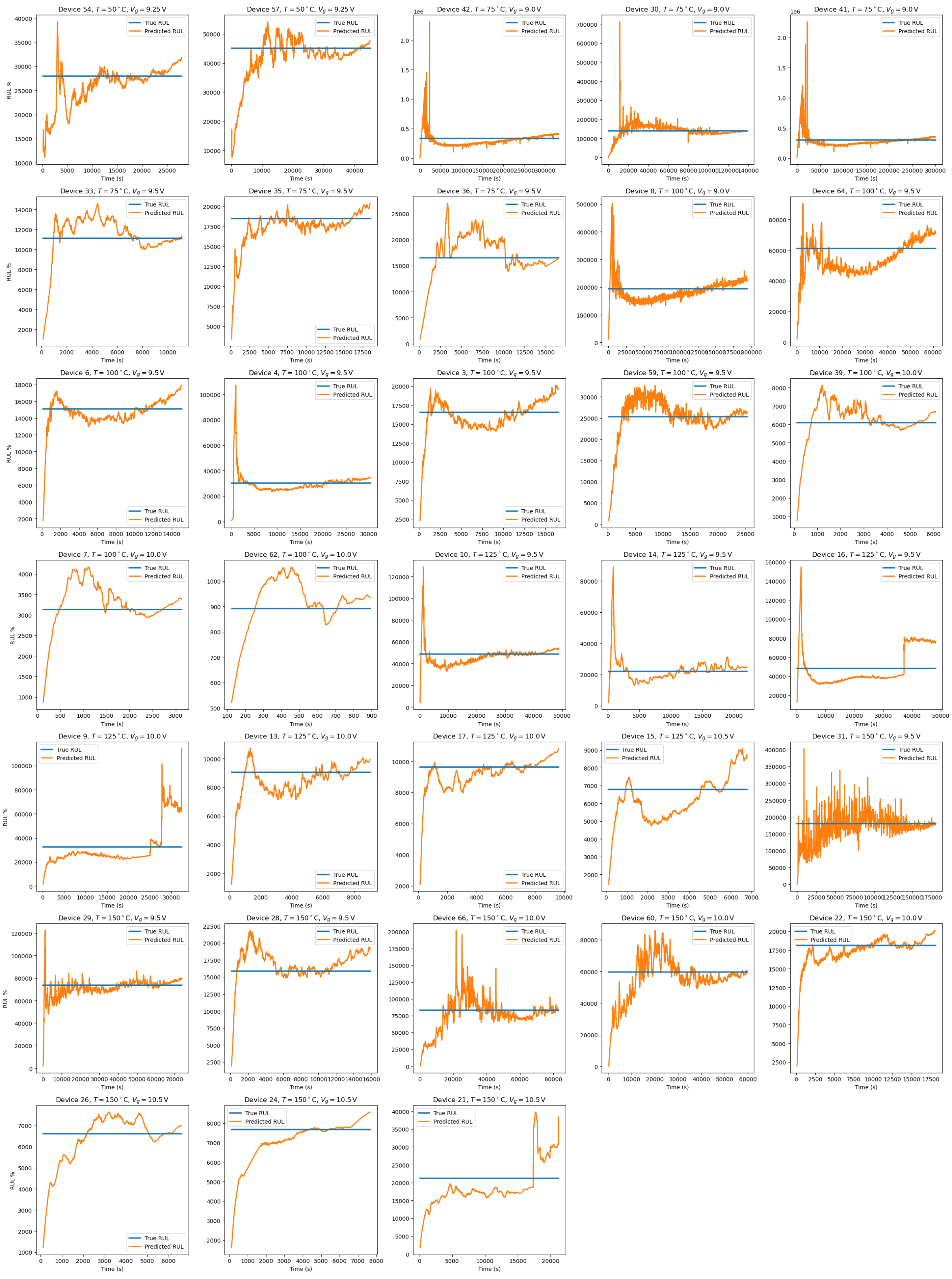


Figure 4.14: Estimated L results on all tested devices.

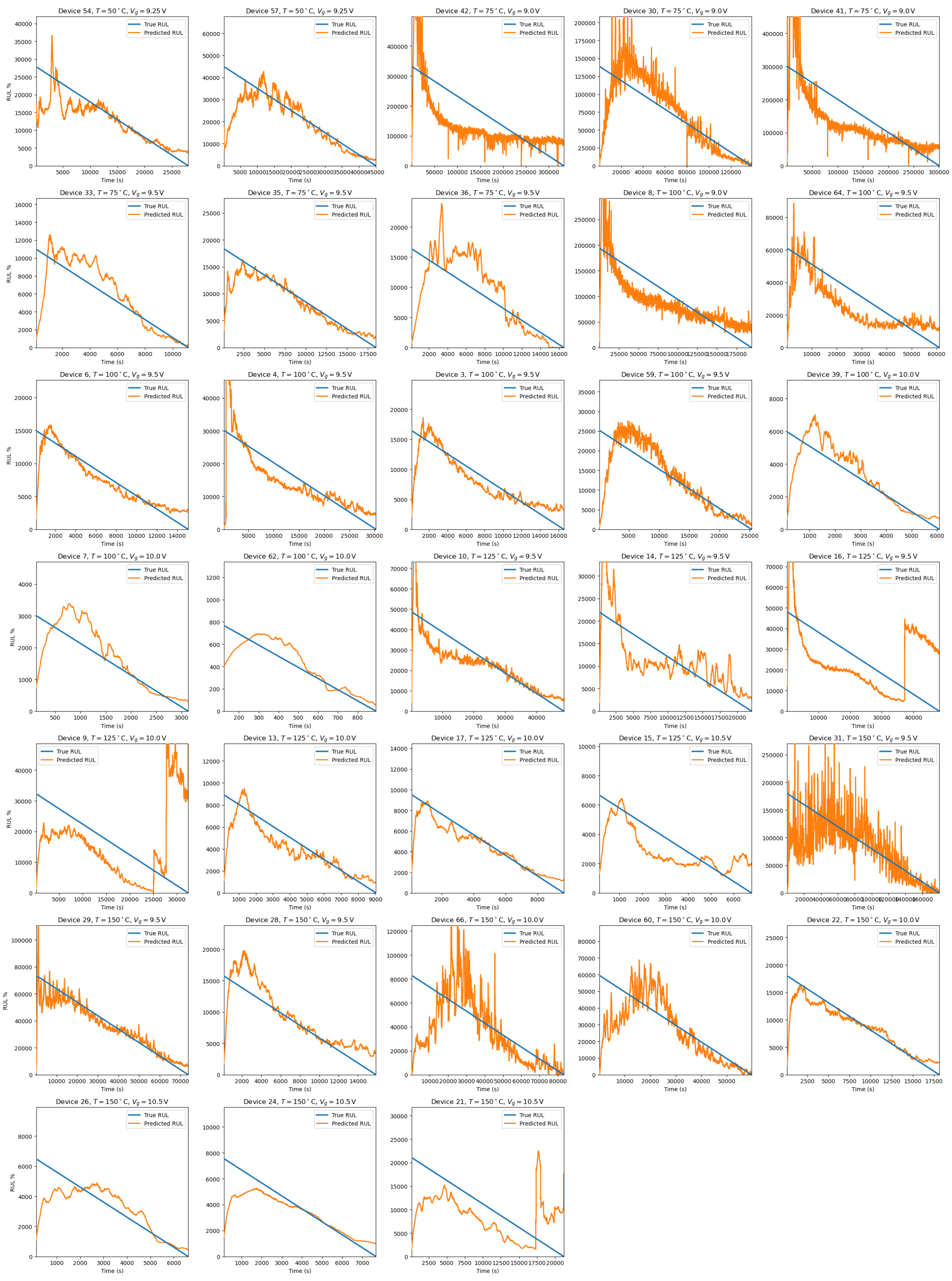


Figure 4.15: Absolute RUL prediction results on all tested devices.

5 Conclusion and Future Work

5.1 Conclusion

GaN HEMTs, as a promising power electronics device, offer significant advantages in terms of switching performance, efficiency, and thermal characteristics. These transistors, based on wide bandgap materials, enable faster switching frequencies, lower conduction losses, and higher power densities compared to traditional silicon-based devices. Their ability to operate under various conditions makes them particularly suitable for modern power conversion applications. As power electronic systems continue to evolve toward higher performance and miniaturization, GaN HEMTs have emerged as a key enabling technology, capable of meeting the increasingly demanding requirements in efficiency, compactness, and reliability.

However, ensuring the long-term reliability of GaN HEMTs remains a key challenge. Over time, performance degradation may occur, increasing the likelihood of device failure and threatening the stability of the overall system. As a result, accurately predicting device lifetime becomes crucial for preventive maintenance and reliable system operation. Previous studies have primarily focused on modeling the survival rate of GaN HEMTs on a statistical level. These population-based approaches typically emphasize group-level statistics, rather than capturing the nuanced degradation behavior of individual devices. While helpful for observing general reliability patterns, they often fail to account for device-to-device variability that occurs in practical deployment scenarios. Moreover, these models do not fully utilize the value of real-time measurement data, which could potentially enhance prediction accuracy at the individual device level.

To overcome these limitations, this thesis presents a prediction framework that incorporates a TCN augmented with CBAM. The framework is designed to estimate the RUL of p-GaN HEMT devices using dynamic signal measurements collected during actual operation. Different from traditional lifetime prediction methods in GaN HEMT, which focus on the statistics of a group under specific environments. This work designed a comprehensive pipeline that predicts the lifetime of individual HEMTs based on environmental conditions and operational measurements. The key contributions of this thesis are as follows:

- Considering the significant variations in the lifetime of GaN HEMT devices under different operating conditions, a step-by-step prediction framework is designed. First, a deep learning model is employed to estimate the relative RUL. Then, a conversion method is proposed to transform the predicted relative RUL into absolute RUL expressed in seconds, enabling the estimation of the total device lifetime.
- Based on RUL prediction techniques, a deep learning model was developed for

lifetime estimation of GaN HEMTs. Manually extracted time-domain features were selected using the mRMR algorithm to construct the training dataset. By integrating attention mechanisms into the TCN architecture, which is designed for time series processing, the model further enhances its ability to capture temporal degradation patterns. The effectiveness of the proposed model was validated through experimental results.

- To improve generalization across varied operating environments, the model is trained and evaluated using a Leave-One-Group-Out Cross-Validation scheme. Devices are clustered based on their environmental conditions, and the model learns from all but one group used for testing. This strategy enables the model to make lifetime predictions under unseen conditions, ensuring its adaptability to new scenarios. By incorporating real-time degradation information and learning from operational variability, the proposed framework improves prediction accuracy and robustness, supporting more reliable health management of GaN-based power electronic systems.

- To enhance the reliability and interpretability of RUL prediction, uncertainty quantification was incorporated into the proposed framework. By adopting GNLL as the training loss, the model was able to output both predicted RUL values and their associated uncertainty intervals. This capability supports more informed and risk-aware decision-making in practical applications.

5.2 Future Work

Although this thesis has achieved several meaningful results, it has also revealed some challenges that require further investigation. Future research directions include the following:

- Further research on the deep learning model can be done. In this thesis, the ablation study is limited to a TCN without the CBAM module. Future research can focus on the contribution of each component to the improvements, including the channel attention module, the temporal attention module, and the residual path. Additionally, integrating state-of-the-art architectural elements may further enhance the model's accuracy and generalization capability.

- As the model is intended to be deployed on memory-constrained devices in industry, it is essential to conduct a comprehensive evaluation of its computational efficiency. Future work should include analysis of key performance indicators such as floating-point operations (FLOPs) and memory footprint. It is necessary to conduct experiments by deploying the model on hardware to evaluate its practical performance.

- This estimated absolute RUL is based on instantaneous predictions. If future research can apply some advanced estimators to exploit historical predictions fully, the noise can be suppressed, and the accuracy can be further improved.

- In this thesis, the proposed model cannot handle anomalous devices tested under extreme environments. Further investigation into the physical characteristics of these samples and their degradation behaviors is necessary. In particular, future work should aim to develop mechanisms capable of consistently predicting the lifetime of both normal and anomalous devices, or dedicated approaches explicitly tailored for anomalous devices could be explored to provide effective lifetime estimation solutions.

Bibliography

[1] K. F. Rahman, S. Falina, M. F. P. Mohamed, H. Kawarada, and M. Syamsul, "The role of gallium nitride in the evolution of electric vehicles: energy applications, technology, and challenges," *Applied Physics Reviews*, vol. 11, no. 3, 2024.

[2] C.-T. Ma and Z.-H. Gu, "Review of GaN HEMT applications in power converters over 500 W," *Electronics*, vol. 8, no. 12, p. 1401, 2019.

[3] H. Lu, M. Zhang, L. Yang, B. Hou, R. P. Martinez, M. Mi, J. Du, L. Deng, M. Wu, S. Chowdhury, *et al.*, "A review of GaN RF devices and power amplifiers for 5G communication applications," *Fundamental Research*, vol. 5, no. 1, pp. 315–331, 2025.

[4] M. Hua, C. Liu, S. Yang, S. Liu, K. Fu, Z. Dong, Y. Cai, B. Zhang, and K. J. Chen, "Characterization of leakage and reliability of SiN_x gate dielectric by low-pressure chemical vapor deposition for GaN-based MIS-HEMTs," *IEEE Transactions on Electron Devices*, vol. 62, no. 10, pp. 3215–3222, 2015.

[5] P. Moens and A. Stockman, "A physical-statistical approach to AlGaN/GaN HEMT reliability," in *2019 IEEE International Reliability Physics Symposium (IRPS)*, pp. 1–6, IEEE, 2019.

[6] A. N. Tallarico, S. Stoffels, P. Magnone, N. Posthuma, E. Sangiorgi, S. Decoutere, and C. Fiegna, "Investigation of the p-GaN gate breakdown in forward-biased GaN-based power HEMTs," *IEEE Electron Device Letters*, vol. 38, no. 1, pp. 99–102, 2016.

[7] F. Deng, Y. Bi, Y. Liu, and S. Yang, "Remaining useful life prediction of machinery: a new multiscale temporal convolutional network framework," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, p. 1–13, Jan. 2022.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017.

[9] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "CBAM: Convolutional block attention module," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19, 2018.

[10] Y. Lei, N. Li, L. Guo, N. Li, T. Yan, and J. Lin, "Machinery health prognostics: a systematic review from data acquisition to RUL prediction," *Mechanical Systems and Signal Processing*, vol. 104, p. 799–834, May 2018.

[11] Y. Mo, Q. Wu, X. Li, and B. Huang, "Remaining useful life estimation via transformer encoder enhanced by a gated convolutional unit," *Journal of Intelligent Manufacturing*, vol. 32, p. 1997–2006, Mar. 2021.

[12] D. Chen, W. Hong, and X. Zhou, "Transformer network for remaining useful life prediction of lithium-ion batteries," *IEEE Access*, vol. 10, pp. 19621–19628, 2022.

[13] J. Wang, R. X. Gao, Z. Yuan, Z. Fan, and L. Zhang, "A joint particle filter and expectation maximization approach to machine condition prognosis," *Journal of Intelligent Manufacturing*, vol. 30, p. 605–621, Oct. 2016.

[14] F. Röder, R. D. Braatz, and U. Krewer, "Multi-scale simulation of heterogeneous surface film growth mechanisms in lithium-ion batteries," *Journal of The Electrochemical Society*, vol. 164, pp. E3335–E3344, Jan. 2017.

[15] F. Röder, R. D. Braatz, and U. Krewer, "Direct coupling of continuum and kinetic Monte Carlo models for multiscale simulation of electrochemical systems," *Computers & Chemical Engineering*, vol. 121, pp. 722–735, 2019.

[16] P. L. T. Duong and N. Raghavan, "Heuristic Kalman optimized particle filter for remaining useful life prediction of lithium-ion battery," *Microelectronics Reliability*, vol. 81, pp. 232–243, 2018.

[17] L. Cui, X. Wang, H. Wang, and J. Ma, "Research on remaining useful life prediction of rolling element bearings based on time-varying Kalman filter," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 6, pp. 2858–2867, 2020.

[18] X.-S. Si, W. Wang, C.-H. Hu, and D.-H. Zhou, "Remaining useful life estimation – a review on the statistical data driven approaches," *European Journal of Operational Research*, vol. 213, no. 1, pp. 1–14, 2011.

[19] T. Escobet, J. Quevedo, and V. Puig, "A fault/anomaly system prognosis using a data-driven approach considering uncertainty," in *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7, 2012.

[20] F. Wang, S. Tang, X. Sun, L. Li, C. Yu, and X. Si, "Remaining useful life prediction based on nonlinear random coefficient regression model with fusing failure time data," *Journal of Systems Engineering and Electronics*, vol. 34, no. 1, pp. 247–258, 2023.

[21] Z. Huang, Z. Xu, X. Ke, W. Wang, and Y. Sun, "Remaining useful life prediction for an adaptive skew-Wiener process model," *Mechanical Systems and Signal Processing*, vol. 87, pp. 294–306, 2017.

[22] K. Song and L. Cui, "A common random effect induced bivariate gamma degradation process with application to remaining useful life prediction," *Reliability Engineering & System Safety*, vol. 219, p. 108200, 2022.

[23] X. Chen, X. Sun, X. Si, and G. Li, "Remaining useful life prediction based on an adaptive inverse Gaussian degradation process with measurement errors," *IEEE Access*, vol. 8, pp. 3498–3510, 2020.

[24] W. Li and T. Liu, "Time varying and condition adaptive hidden Markov model for tool wear state estimation and remaining useful life prediction in micro-milling," *Mechanical Systems and Signal Processing*, vol. 131, pp. 689–702, 2019.

[25] Z. Chen, S. Cao, and Z. Mao, "Remaining useful life estimation of aircraft engines using a modified similarity and supporting vector machine (SVM) approach," *Energies*, vol. 11, no. 1, 2018.

[26] D. Ji, C. Wang, J. Li, and H. Dong, "A review: data driven-based fault diagnosis and RUL prediction of petroleum machinery and equipment," *Systems Science & Control Engineering*, vol. 9, p. 724–747, Jan. 2021.

[27] A. wahhab Lourari, T. Benkedjouh, B. El Yousfi, and A. Soualhi, "An ANFIS-based framework for the prediction of bearing's remaining useful life," *International Journal of Prognostics and Health Management*, vol. 15, no. 1, 2024.

[28] Y. Zhang, D. Liu, J. Yu, Y. Peng, and X. Peng, "EMA remaining useful life prediction with weighted bagging GPR algorithm," *Microelectronics Reliability*, vol. 75, pp. 253–263, 2017.

[29] G. Jiang, W. Zhou, Q. Chen, Q. He, and P. Xie, "Dual residual attention network for remaining useful life prediction of bearings," *Measurement*, vol. 199, p. 111424, Aug. 2022.

[30] F. O. Heimes, "Recurrent neural networks for remaining useful life estimation," in *2008 International Conference on Prognostics and Health Management*, pp. 1–6, 2008.

[31] S. Zheng, K. Ristovski, A. Farahat, and C. Gupta, "Long short-term memory network for remaining useful life estimation," in *2017 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pp. 88–95, 2017.

[32] T. Berghout, L.-H. Mouss, O. Kadri, L. Saïdi, and M. Benbouzid, "Aircraft engines remaining useful life prediction with an adaptive denoising online sequential extreme learning machine," *Engineering Applications of Artificial Intelligence*, vol. 96, p. 103936, 2020.

[33] M. Haris, M. N. Hasan, and S. Qin, "Early and robust remaining useful life prediction of supercapacitors using BOHB optimized deep belief network," *Applied Energy*, vol. 286, p. 116541, 2021.

[34] C. Ordóñez, F. Sánchez Lasheras, J. Roca-Pardiñas, and F. J. de Cos Juez, "A hybrid ARIMA–SVM model for the study of the remaining useful life of aircraft engines," *Journal of Computational and Applied Mathematics*, vol. 346, pp. 184–191, 2019.

[35] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[36] B. C. Ross, “Mutual information between discrete and continuous data sets,” *PloS one*, vol. 9, no. 2, p. e87357, 2014.

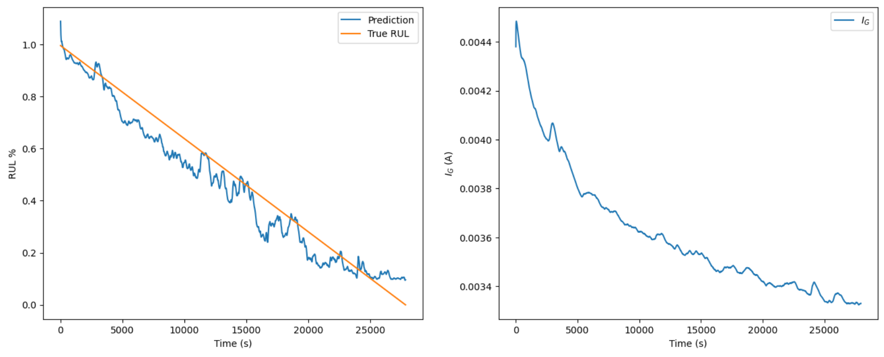
[37] S. Bai, J. Z. Kolter, and V. Koltun, “An empirical evaluation of generic convolutional and recurrent networks for sequence modeling,” *arXiv preprint arXiv:1803.01271*, vol. 10, 2018.

[38] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

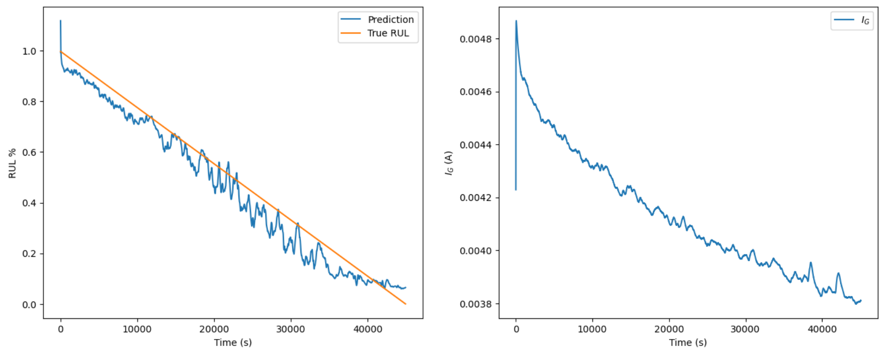
[39] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[40] W. Liao, Z. Miao, S. Liang, L. Zhang, and C. Li, “A composite improved attention convolutional network for motor imagery EEG classification,” *Frontiers in Neuroscience*, vol. 19, p. 1543508, 2025.

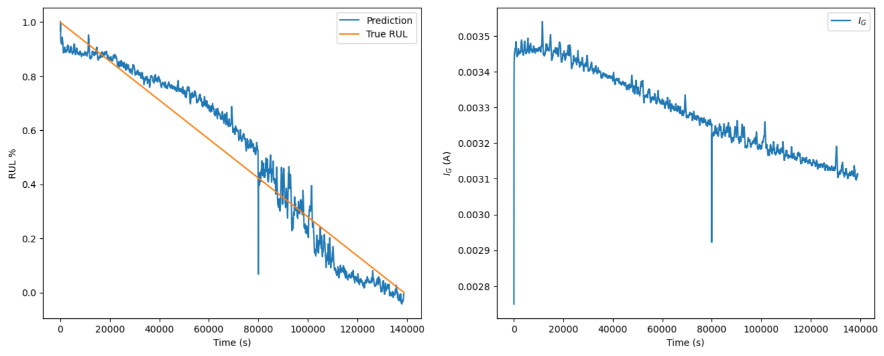
A
Examples of normal and anomalous devices



(a) Device 59

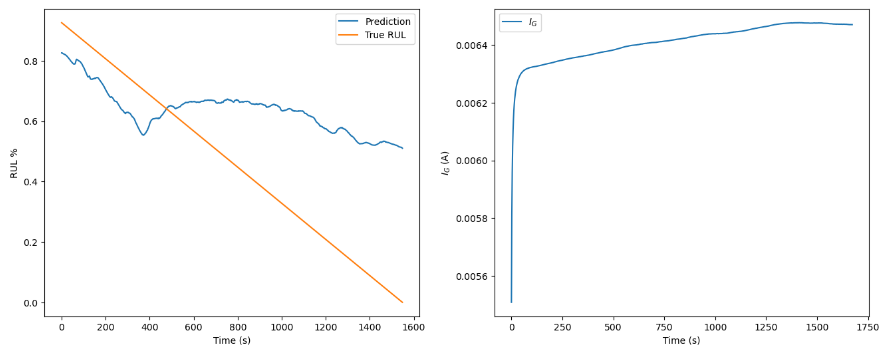


(b) Device 57

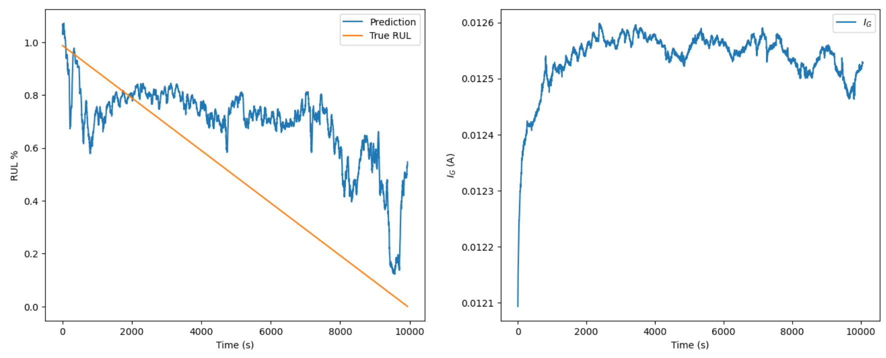


(c) Device 30

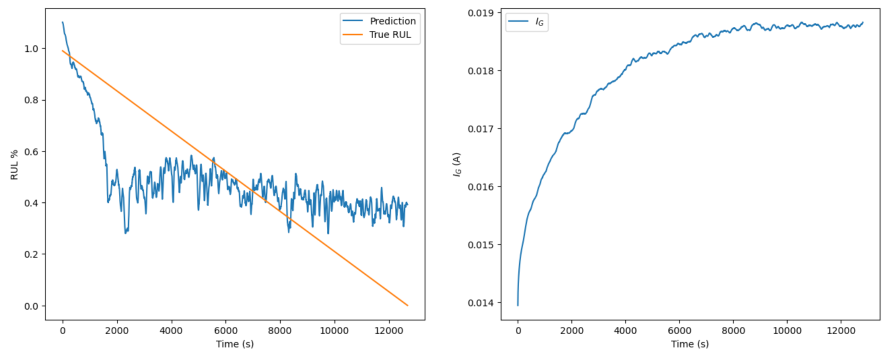
Figure A.1: Examples of normal devices and corresponding I_G.



(a) Device 47



(b) Device 61



(c) Device 5

Figure A.2: Examples of anomalous devices and corresponding I_G.