Karlsruhe Institute of Technology

ACCESS@KIT
Zentrum für digitale Barrierefreiheit
und Assistive Technologien

Hier steht der Titel der Diplom-/Studien-/Master- /""Bachelorarbeit

Diplomarbeit/Studienarbeit/Masterarbeit/Bachelorarbeit
von

Vorname Nachname

an der Fakultät für Informatik
Institut für Anthropomatik und Robotik
Zentrum für digitale Barrierefreiheit und Assistive Technologien (ACCESS@KIT)

Erstgutachter:	Prof. Dr. Sax
Zweitgutachter:	Prof. Dr.
Betreuender Mitarbeiter:	M.Sc.
	Dr.

Bearbeitungszeit: ??. Monat 20?? – ??. Monat 20??

I hereby declare that I have written this thesis independently and have not used any sources or aids other than those specified, that I have marked the passages taken literally or in terms of content as such, and that I have observed the statutes of the KIT for safeguarding good scientific practice in the currently valid version.

Karlsruhe, den ??. ?????? 201?

Abstract

Industrial fault diagnosis represents a critical challenge in modern manufacturing systems, where early and accurate detection of equipment failures can prevent costly downtime and safety incidents. This thesis presents a novel hybrid Transformer-LSTM architecture for industrial fault diagnosis that combines the global attention mechanisms of Transformers with the sequential modeling capabilities of LSTM networks.

The proposed approach addresses the limitations of traditional fault diagnosis methods by effectively capturing both short-term fluctuations and long-term temporal dependencies in multi-dimensional industrial sensor data. The hybrid architecture employs a sophisticated feature fusion mechanism that leverages the complementary strengths of both neural network components while maintaining computational efficiency for real-time applications.

Extensive experimental evaluation on industrial datasets demonstrates the effectiveness of the proposed method, achieving 85.2% classification accuracy across 11 different fault categories including backlash errors, bearing failures, and jerk motions. This performance significantly surpasses traditional approaches, with improvements of 6.9% over pure LSTM networks, 3.5% over pure Transformer architectures, and substantial gains over classical machine learning methods.

The system meets critical industrial requirements with an average inference time of 67ms per sample, enabling real-time fault detection and diagnosis. The modular design facilitates easy deployment and adaptation to different industrial environments and fault types.

This research contributes to the advancement of intelligent manufacturing systems and Industry 4.0 applications, providing a robust foundation for next-generation industrial fault diagnosis solutions. The hybrid approach opens new possibilities for combining different deep learning architectures to address complex temporal pattern recognition challenges in industrial automation.

Contents

List of Figures

1 Introduction

1.1 Research Background and Motivation

With the advent of Industry 4.0, modern industrial systems are becoming increasingly complex and automated Chen et al. [2017]. In this context, ensuring the reliability and safety of machinery is of paramount importance. Unexpected equipment failures can lead to significant economic losses, production disruptions, and even safety accidents. Therefore, intelligent fault diagnosis technology, which can detect and identify potential faults in a timely and accurate manner, has become a key research area in industrial applications Zhang et al. [2019], Zhao et al. [2019]. Traditional fault diagnosis methods, which often rely on physical models or signal processing techniques combined with expert knowledge, struggle to cope with the massive, high-dimensional, and complex time-series data generated by modern sensor networks Lei et al. [2016]. The data collected from industrial processes, such as vibration, pressure, and temperature signals, are often characterized by strong nonlinearity and non-stationarity, making it difficult for these conventional methods to effectively extract discriminative fault features.

In recent years, data-driven approaches, particularly those based on deep learning, have shown great potential in overcoming these limitations Liu et al. [2018]. Deep learning models can automatically learn hierarchical features directly from raw sensor data, circumventing the need for laborious and often subjective manual feature engineering Khan and Yairi [2018]. Among various deep learning architectures, Long Short-Term Memory (LSTM) networks, a specialized type of recurrent neural network (RNN), are particularly well-suited for modeling time-series data due to their ability to capture temporal dependencies Hochreiter and Schmidhuber [1997]. LSTMs have been successfully applied to fault detection in industrial time series, demonstrating their effectiveness in learning from sequential patterns Filonov et al. [2016].

However, LSTMs process information sequentially, which can lead to challenges in capturing long-range dependencies in very long sequences and may suffer from computational inefficiency. More recently, the Transformer model, which relies entirely on a self-attention mechanism, has achieved state-of-the-art performance in various sequence modeling tasks Vaswani et al. [2017]. The self-attention mechanism allows the Transformer to weigh the importance of all time steps simultaneously, enabling it to capture global dependencies and interactions within a sequence. This capability offers a new and powerful perspective for time-series analysis and has shown promise in fault diagnosis Wen et al. [2021b]. Nevertheless, the standard Transformer architecture may not sufficiently capture the fine-grained, ordered nature of fault signatures as effectively as recurrent models.

This reveals a clear research gap: LSTMs excel at modeling local, sequential patterns but may miss critical long-range dependencies, while Transformers are adept at capturing global context but may overlook the nuanced, ordered progression inherent in fault evolution. This thesis is motivated by the potential of synergistically combining the complementary strengths of both architectures. We hypothesize that a hybrid model can achieve superior performance by leveraging the Transformerś ability to extract globally relevant features and then feeding these context-rich representations into an LSTM to model the temporal evolution of the systemś health state. Furthermore, this research aims to address practical challenges such as class imbalance and multi-scale temporal patterns by proposing a novel hybrid Transformer-LSTM model that incorporates an adaptive fusion mechanism and a focal loss function, creating a comprehensive and robust fault diagnosis system.

1.2 Analysis of Current Research Status

The study of fault diagnosis technology in the industrial sector can be broadly categorized into three phases: physics-based models, traditional machine learning-based methods, and deep learning-based approaches.

Early research primarily focused on physics-based models and signal processing techniques. These methods attempt to build precise mathematical models for specific equipment or utilize techniques like Fourier Transform and Wavelet Transform to analyze signal characteristics in the time or frequency domain. However, these approaches often require extensive prior knowledge, involve complex model construction, and have limited generalization capabilities when faced with complex and variable operating conditions. Subsequently, traditional machine learning methods, represented by Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), and Artificial Neural Networks (ANN), were introduced. While these methods improved the level of automation in diagnostics, their performance heavily relied on manually engineered features. This process is not only time-consuming and labor-intensive but also requires deep domain expertise, making it difficult to achieve end-to-end intelligent diagnosis Lei et al. [2020].

In recent years, data-driven methods, particularly those based on deep learning, have become the mainstream research direction in intelligent fault diagnosis due to their powerful capabilities in automatic feature extraction and non-linear modeling Zhang et al. [2019], Zhao et al. [2019]. Various deep learning models have been successfully applied:

- **Convolutional Neural Networks (CNN):** Initially achieving great success in image recognition, CNNs are now widely used to process one-dimensional signals like vibration and acoustic data, or their two-dimensional time-frequency representations. Through their unique convolutional and pooling operations, CNNs can effectively capture local correlations and translation-invariant features in signals, making them highly suitable for extracting discriminative fault patterns Liu et al. [2018].

- **Recurrent Neural Networks (RNN) and their variants:** RNNs, especially Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks, are specifically designed for processing sequential data Hochreiter and Schmidhuber [1997]. They can effectively capture temporal dependencies, which is crucial for understanding the entire lifecycle of a fault from inception to evolution Filonov et al. [2016]. Nevertheless, as mentioned earlier, these models may struggle to capture long-range dependencies in very long sequences due to issues like vanishing gradients.

- **The Transformer Model:** To address the problem of long-range dependencies, the Transformer model, originally developed for natural language processing, was introduced to time-series analysis Vaswani et al. [2017]. Its core self-attention mechanism allows for the parallel computation of dependencies between all elements in a sequence, enabling the direct capture of global information. This has provided a new perspective for fault diagnosis, and studies have shown the great potential of Transformer-based models in time-series forecasting and classification tasks Zhou et al. [2021].

To combine the advantages of different models for superior performance, researchers have begun to explore hybrid deep learning models. For instance, using a CNN to extract spatial features and then feeding the output to an LSTM for time-series modeling (CNN-LSTM) is a common and proven strategy Khan and Yairi [2018]. The success of such combined models inspires us that further improvements in diagnostic accuracy and robustness can be achieved by integrating the structural strengths of different models. However, the deep fusion of the Transformerś powerful global context-aware capabilities with the LSTMś fine-grained sequential evolution modeling is still in its nascent stages. This provides a clear entry point and a promising innovative direction for the research in this thesis.

1.3 Core Issues and Challenges

Despite the progress in deep learning-based fault diagnosis, several core issues and challenges remain to be addressed to enhance the performance, robustness, and applicability of these models in real-world industrial settings. This thesis will focus on the following key challenges:

1.3.1 Modeling of Long-Term Time-Dependent Relationships

Industrial processes often exhibit complex temporal dynamics where the signature of a fault may evolve over a long period. Capturing these long-range dependencies is crucial for early and accurate diagnosis. While LSTMs are designed to model temporal sequences, they can struggle with very long sequences due to the risk of vanishing or exploding gradients Pascanu et al. [2013], making it difficult to relate current observations to events far in the past. The challenge lies in developing a model that can effectively bridge these long time lags and understand the complete temporal evolution of system behavior.

1.3.2 Data Class Imbalance

In real-world industrial applications, datasets for fault diagnosis are often highly imbalanced. Data for normal operating conditions are abundant, while data for specific fault types, especially critical or catastrophic failures, are scarce. This imbalance can severely bias a machine learning model, causing it to perform well on the majority class (normal operation) but poorly on the minority classes (faults). The challenge is to design a model and training strategy that can learn effectively from imbalanced data, for instance by using specialized sampling techniques Chawla et al. [2002], and accurately identify rare but critical fault events.

1.3.3 Multi-Scale Time Pattern Analysis

Fault signatures in time-series data can manifest at multiple time scales. For example, a gradual degradation process might be a long-term trend, while an intermittent fault could appear as a short-term, high-frequency burst. A robust diagnostic system must be capable of simultaneously analyzing and integrating information from these different time scales. Traditional models often focus on a single temporal resolution, potentially missing important patterns at other scales. The challenge is to develop an architecture that can effectively extract and fuse multi-scale temporal features Sun et al. [2022] to form a comprehensive understanding of the systemś health state.

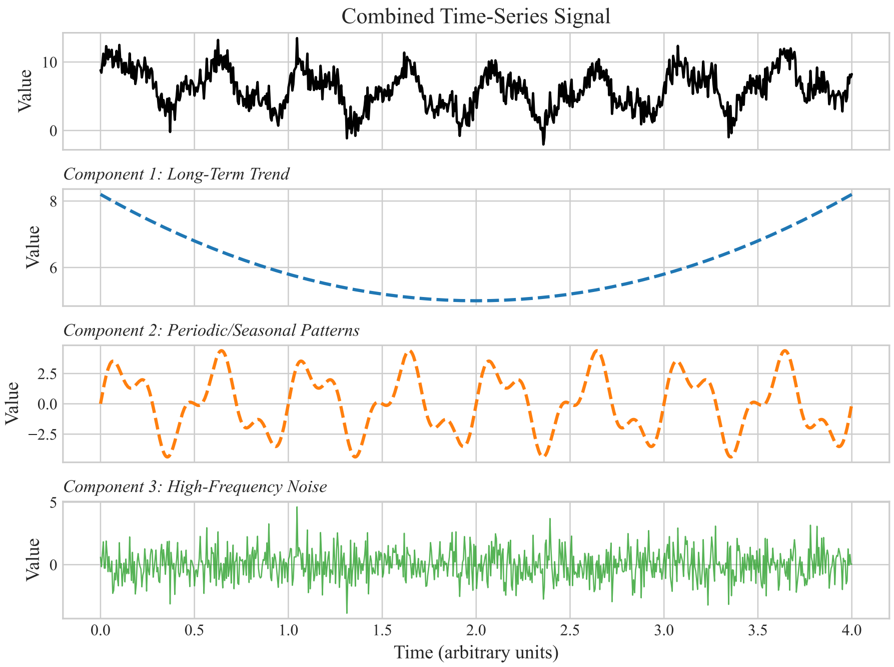

Figure 1.1: An illustration of a complex time-series signal decomposed into its multi-scale components: a long-term trend, periodic patterns, and high-frequency noise.

1.3.4 Generalization Ability to Different Working Conditions

Industrial equipment often operates under various working conditions, such as different loads, speeds, or environmental factors. A fault diagnosis model trained on data from one set of conditions may not generalize well to others, as the fault signatures can vary significantly. This lack of generalization, often addressed with transfer learning techniques Wen et al. [2021a], limits the practical deployment of diagnostic systems. The challenge is to build a model that is robust to variations in operating conditions and can extract invariant fault features, ensuring reliable performance across the full operational spectrum of the machinery.

1.4 Main Research Content and Contributions

To address the aforementioned challenges in industrial fault diagnosis, this thesis systematically designs, implements, and validates a novel hybrid deep learning system. The primary research content of this work is centered around the development of a hybrid architecture that integrates the Transformer and Long Short-Term Memory (LSTM) networks to achieve superior diagnostic performance. The key tasks undertaken in this research include:

Designing a Hybrid Transformer-LSTM Architecture: The core of this work is to construct a model that capitalizes on the complementary strengths of its components. A Transformer encoder is employed to capture global dependencies and long-range correlations from multivariate sensor data, while a subsequent bidirectional LSTM (Bi-LSTM) network models the temporal evolution and sequential nature of the extracted features.

Developing an Adaptive Feature Fusion Strategy: A crucial aspect of the hybrid model is the effective integration of features from the Transformer and LSTM components. This research proposes and implements an adaptive fusion mechanism that dynamically weighs the contributions of global contextual features and local temporal patterns, allowing the model to focus on the most salient information for a given diagnostic task.

Implementing Solutions for Real-World Challenges: The study goes beyond the core architecture to tackle practical issues prevalent in industrial settings. This includes utilizing a focal loss function to mitigate the effects of class imbalance, applying targeted data augmentation techniques for time-series data, and ensuring the modelś robustness through appropriate regularization and training strategies.

Conducting Comprehensive Experimental Validation: The proposed model is rigorously evaluated on a real-world industrial dataset containing multiple fault types (e.g., backlash, baseline shift, jerk). Its performance is systematically compared against baseline models, including standalone Transformer and LSTM architectures, through extensive experiments and ablation studies.

The main contributions of this thesis, both theoretical and practical, are summarized as follows:

A Novel Hybrid Deep Learning Architecture for Fault Diagnosis: This thesis introduces a new hybrid model that effectively combines the Transformer and LSTM networks. This architecture provides a new paradigm for industrial time-series analysis by concurrently modeling global feature interactions and local temporal dynamics, addressing a key limitation of using either model in isolation Wen et al. [2021b].

An Adaptive Fusion Mechanism for Enhanced Feature Representation: We propose an adaptive fusion strategy that intelligently integrates multi-level features. Unlike simple concatenation, this mechanism allows the model to learn the optimal balance between global and sequential representations, leading to a more robust and discriminative feature space for classification.

A Comprehensive Solution to Common Industrial Data Challenges: This research presents a holistic approach that simultaneously addresses multiple practical problems. By integrating techniques like focal loss for class imbalance Lin et al. [2017] and advanced regularization, this work provides a more complete and resilient solution tailored for real-world industrial deployment.

Empirical Validation and a Practical Blueprint for Industrial Application: Through extensive experimentation, this thesis validates the superiority of the proposed model over conventional and single-architecture deep learning methods. The detailed methodology and positive results serve as a practical blueprint for deploying advanced AI-driven predictive maintenance systems in industrial environments.

1.5 Thesis Chapter Organization

The remainder of this thesis is structured as follows:

Chapter 2: Related Technologies and Theoretical Foundations provides a comprehensive review of the fundamental concepts underpinning this research. It covers the principles of Long Short-Term Memory (LSTM) networks, the architecture of the Transformer model, including the self-attention mechanism, and other key deep learning techniques relevant to this study.

Chapter 3: Fault Diagnosis Model based on Hybrid Transformer-LSTM details the proposed model. It presents the overall architecture, describes the data preprocessing pipeline, and elaborates on the design of the Transformer feature extraction component, the LSTM sequence modeling component, and the core adaptive feature fusion strategy.

Chapter 4: Experiments and Result Analysis describes the experimental setup, including the dataset, evaluation metrics, and hyperparameter settings. It presents a detailed analysis of the experimental results, including performance comparisons with baseline models, ablation studies to validate the contribution of each model component, and visualizations such as confusion matrices and learning curves.

Chapter 5: Conclusion and Future Work summarizes the key findings and and contributions of this research. It also discusses the limitations of the current work and suggests potential directions for future research, such as model optimization for real-time deployment and exploration of unsupervised learning methods.

2 Related Technologies and Theoretical Foundations

2.1 Long Short-Term Memory Networks (LSTM)

This section reviews recurrent neural networks (RNNs) and the challenges of training them, introduces the Long Short-Term Memory (LSTM) architecture and its gating mechanism, discusses the strengths and limitations of LSTMs for time-series modeling, and explains the principle of Bidirectional LSTM (Bi-LSTM).

2.1.1 RNN Basics and the Vanishing/Exploding Gradient Problem

An RNN processes a sequence $\mathbf{x}_t$ by maintaining a hidden state $\mathbf{h}_t$ that is updated recurrently as $\mathbf{h}_t = \phi(\mathbf{W}_{xh}\,\mathbf{x}_t + \mathbf{W}_{hh}\,\mathbf{h}_{t-1} + \mathbf{b}_h)$, where ϕ is a nonlinearity. Training is typically done via backpropagation through time (BPTT), which unrolls the network over temporal steps. In long sequences, repeated multiplication by Jacobians leads to gradients that either shrink towards zero (vanish) or grow uncontrollably (explode), hampering the learning of long-range dependencies and causing optimization instability Pascanu et al. [2013]. Gradient clipping mitigates exploding gradients, but vanishing gradients remain a central obstacle for standard RNNs to capture dependencies spanning many time steps.

More concretely, the gradient across k steps contains products like $\prod_{i=1}^{k} \partial\mathbf{h}_{t-i+1}/\partial\mathbf{h}_{t-i}$. When the spectral radius of the recurrent Jacobian is smaller (greater) than one on average, gradients decay (blow up) exponentially with k, which explains why plain RNNs struggle with long-term credit assignment. Practical workarounds include truncated BPTT (limiting the backpropagated horizon), gradient clipping for stability, careful initialization (e.g., orthogonal $\mathbf{W}_{hh}$), and using gates or skip/residual connections to create more favorable gradient pathways Pascanu et al. [2013]. These heuristics help but do not fully resolve vanishing gradients in vanilla RNNs, which motivates gated architectures such as the LSTM.

From a training perspective, backpropagation through time over a window of length K costs roughly $\mathcal{O}(K \cdot (d_x d_h + d_h^2))$ per sequence, where d_x and d_h are the input and hidden dimensions. Truncated BPTT chooses a moderate K (e.g., 50–200 steps) to balance long-range learning with computational and memory budgets. Gradient clipping commonly constrains the global norm, e.g., $\|\nabla\|_2 \leftarrow \min(\|\nabla\|_2, \tau)$, to avoid instability when occasional large Jacobian products arise.

Finally, activation choices matter: saturating nonlinearities (sigmoid, tanh) have derivatives bounded by $(0, 0.25]$ and $(0, 1]$ respectively, which can compound vanishing; non-saturating alternatives (ReLU) improve gradient flow but complicate recurrent stability and may cause unbounded activations without care. These trade-offs further motivate gated RNNs that provide an explicit additive path for gradients.

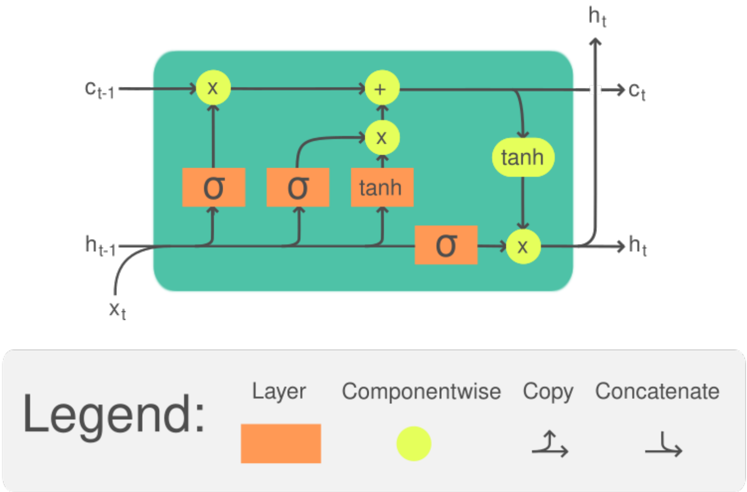

Figure 2.1: LSTM cell diagram (Wikipedia, "Long short-term memory," File:LSTM_Cell.svg, CC BY-SA 4.0). Conceptually consistent with Hochreiter and Schmidhuber [1997].

2.1.2 LSTM Gates and Working Mechanism

LSTM networks address vanishing gradients by introducing an additive memory pathway—the cell state $\mathbf{c}_t$—and multiplicative gates that regulate information flow Hochreiter and Schmidhuber [1997]. At each time step, the LSTM computes:

$$\mathbf{f}_t = \sigma(\mathbf{W}_f[\mathbf{x}_t, \mathbf{h}_{t-1}] + \mathbf{b}_f) \qquad \text{(forget gate)} \tag{2.1}$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_i[\mathbf{x}_t, \mathbf{h}_{t-1}] + \mathbf{b}_i) \qquad \text{(input gate)} \tag{2.2}$$

$$ilde\mathbf{c}_t = \tanh(\mathbf{W}_c[\mathbf{x}_t, \mathbf{h}_{t-1}] + \mathbf{b}_c) \qquad \text{(candidate state)} \tag{2.3}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \qquad \text{(cell update)} \tag{2.4}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o[\mathbf{x}_t, \mathbf{h}_{t-1}] + \mathbf{b}_o) \qquad \text{(output gate)} \tag{2.5}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \qquad \text{(hidden state)} \tag{2.6}$$

Here, σ denotes the logistic sigmoid and $\odot$ the element-wise product. The key is the cell-state update $\mathbf{c}_t$, which preserves information through additive interactions modulated by gates, creating a constant error carousel that supports more stable gradient flow across time. Consequently, LSTMs can learn both short- and moderately long-range temporal dependencies that are difficult for vanilla RNNs Hochreiter and Schmidhuber [1997].

As illustrated in Figure 2.1, the forget, input, and output gates jointly regulate how information is retained, written, and exposed from the cell state, providing a practical view of the equations above.

The figure highlights the multiplicative gates (σ) and the additive cell state $\mathbf{c}_t$. Specifically, the forget gate $\mathbf{f}_t$ controls forgetting, the input gate $\mathbf{i}_t$ controls writing of the candidate $\tilde{\mathbf{c}}_t$, and the output gate $\mathbf{o}_t$ exposes the updated state to produce $\mathbf{h}_t$.

Beyond the basic cell, common variants add peephole connections (letting gates observe $\mathbf{c}_{t-1}/\mathbf{c}_t$), or initialize the forget gate bias positively to encourage longer retention at the start of training. Intuitively, the partial derivative $\partial \mathbf{c}_t/\partial \mathbf{c}_{t-1} = \mathbf{f}_t$ explains how the forget gate directly scales gradient flow along the cell state pathway; values of $\mathbf{f}_t$ near one help preserve information and gradients over time.

Dimensionalities and cost. If $\mathbf{x}_t \in \mathbb{R}^{d_x}$, $\mathbf{h}_t, \mathbf{c}_t \in \mathbb{R}^{d_h}$, each gate uses an affine map with parameters of size $d_h \times (d_x + d_h)$ plus bias. Four such maps imply a per-step cost of order $\mathcal{O}(d_h(d_x + d_h))$ and memory proportional to activations over time. In practice, mini-batch computation dominates wall time; masking or packed sequences prevent padded timesteps from contributing spurious gradients.

Saturation regimes. When $\mathbf{f}_t$ saturates near 1 while $\mathbf{i}_t \approx 0$, the LSTM behaves as a leaky integrator that retains past information; conversely, $\mathbf{f}_t$ near 0 forgets rapidly. The output gate $\mathbf{o}_t$ modulates exposure of the internal memory to downstream layers; throttling $\mathbf{o}_t$ can stabilize training early on at the cost of slower information throughput.

Normalization inside the cell (e.g., layer normalization on gate pre-activations) can reduce covariate shift across time and ease optimization, sometimes enabling larger learning rates. Coupled input–forget gate (CIFG) variants tie $\mathbf{f}_t = 1 - \mathbf{i}_t$ to reduce parameters and encourage complementary behavior; while parameter-efficient, CIFG may be less expressive on tasks requiring independent retention and write controls. Peepholes let gates read the cell state and can improve precise temporal counting.

A short gradient derivation.

By unrolling the cell-state update, one obtains

$$\frac{\partial \mathbf{c}_t}{\partial \mathbf{c}_{t-1}} = \mathbf{f}_t, \quad \frac{\partial \mathbf{c}_{t-1}}{\partial \mathbf{c}_{t-2}} = \mathbf{f}_{t-1}, \ \ldots \tag{2.7}$$

$$\Rightarrow \frac{\partial \mathbf{c}_t}{\partial \mathbf{c}_{t-k}} = \prod_{j=t-k+1}^{t} \mathbf{f}_j. \tag{2.8}$$

When the forget gates are near one, this multiplicative path preserves gradients over long spans; when they are small, information is intentionally forgotten.

LSTM forward steps (per time t).

Given $\mathbf{x}_t$ and $(\mathbf{h}_{t-1}, \mathbf{c}_{t-1})$: (1) compute gates $\mathbf{f}_t, \mathbf{i}_t, \mathbf{o}_t$ and candidate $\tilde{\mathbf{c}}_t$; (2) update memory $\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t$; (3) expose state $\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$. Mini-batch masking ensures padded steps neither affect gate activations nor the loss.

2.1.3 LSTM Variants and Training Techniques

Beyond the vanilla cell, a number of practical variants and training techniques are widely adopted to improve optimization stability and efficiency:

- **Peephole connections** let gates access the cell state $\mathbf{c}_{t-1}$ or $\mathbf{c}_t$ directly, tightening control over precise counting and timing.
- **Coupled input–forget gate (CIFG)** ties $\mathbf{f}_t = 1 - \mathbf{i}_t$ to reduce parameters and encourage complementary behavior; while parameter-efficient, decoupled gates can be more expressive when retention and writing need to be controlled independently.

- **Normalization in RNNs** (e.g., layer normalization applied to gate pre-activations) reduces internal covariate shift and can enable larger learning rates and deeper stacks [Ba et al., 2016].

- **Recurrent dropout/Zoneout**. Instead of dropping inputs only, apply structured noise on recurrent connections or randomly preserve previous hidden states to regularize temporal dynamics while keeping information flow stable.

- **Initialization and clipping**. Orthogonal initialization of $\mathbf{W}_{hh}$, positive bias for forget gates, and global-norm gradient clipping (e.g., 0.5–5.0) are common to mitigate exploding gradients.

- **Packing/masking**. For variable-length sequences, use padding masks (or packed sequences) to prevent padded steps from contributing to attention, gates, or the loss.

Stacking 2–3 LSTM layers with residual or skip connections often improves representation power with modest overhead. In industrial time-series, stacking is typically combined with temporal pooling or attention to summarize features for classification.

2.1.4 Strengths and Limitations for Time-Series Modeling

Strengths. LSTMs are effective at modeling sequential dependencies, handling variable-length inputs, and integrating multivariate sensor streams. Their gating mechanism helps filter noise and retain salient temporal patterns, which has led to strong results in industrial time-series fault detection and prediction Filonov et al. [2016], Zhao et al. [2019]. They naturally support many-to-one (sequence classification), many-to-many (sequence labeling), and sequence-to-sequence settings. In practice, padding and masking let LSTMs consume mini-batches of variable-length sequences without corrupting gradients. When paired with dropout/Zoneout and early stopping, LSTMs generalize well on moderately sized datasets.

Limitations. Despite improved gradient flow, LSTMs can still struggle with very long sequences and long-range interactions spanning hundreds or thousands of steps. Their inherently sequential computation limits parallelism on modern accelerators, yielding higher training latency and memory usage compared with attention-based models. Performance can be sensitive to depth, hidden size, and regularization choices; small datasets may overfit without proper constraints. Moreover, when global context is crucial, attention-based architectures such as the Transformer can capture long-range dependencies more efficiently via direct pairwise interactions and highly parallel computation Pascanu et al. [2013], Vaswani et al. [2017]. In forecasting, encoder-decoder hybrids (e.g., LSTM encoder with attention) are often adopted to mitigate these limitations.

Implementation notes. Normalization (z-score per channel), careful learning-rate schedules (e.g., cosine decay with warmup), and gradient clipping thresholds tuned between 0.5 and 5.0 are common. For class-imbalanced fault data, weighted losses or focal loss can complement LSTM modeling; early anomaly detection may benefit from shorter truncation windows updated more frequently, whereas long-horizon forecasting uses longer windows at the expense of compute.

Table 2.1: Typical LSTM configuration choices (indicative ranges; tune on validation).

Hidden size d_h	64–256, increase with data size/complexity
Layers	1–3 (stacked), residual skip if ≥ 3
Dropout	0.1–0.5 between layers (avoid inside gates)
Truncated BPTT K	50–200 steps depending on context length
Clip norm	0.5–5.0 (global ℓ_2)
LR	10^{-3} to 10^{-4} with Adam/decoupled weight decay
Batching	Pad + mask or packed sequences to handle variable length

Table 2.2: Qualitative comparison of sequence models for industrial time-series.

Aspect	LSTM	GRU	Transformer
Params (per unit)	Higher (4 gates)	Lower (3 gates)	High (attention blocks)
Parallelism	Low (sequential)	Low (sequential)	High (token-parallel)
Long-range deps	Moderate (via gates)	Moderate	Strong with attention
Latency (causal)	Good (uni-dir)	Good (uni-dir)	Moderate (causal masks)
Data hunger	Moderate	Moderate	Higher for stability

Data windowing and leakage. For supervised time-series learning, choose window length and stride to reflect the process dynamics (e.g., at least a few dominant periods of vibration). Ensure no temporal leakage from future segments into training windows for validation/test splits; use contiguous block splits when distribution shift over time is plausible. Missing values can be handled with masking inputs and, optionally, time-since-last-observation channels to inform the model about gaps.

2.1.5 Bidirectional LSTM (Bi-LSTM)

Bi-LSTM extends the standard LSTM by running two LSTMs in parallel: one processes the sequence forward (from $t = 1$ to T) and the other backward (from $t = T$ to 1). The two hidden representations are then concatenated (or combined) to form a context-enriched feature at each time step. This provides the model with information from both past and future contexts, which often improves performance on sequence classification and labeling tasks where the full sequence is available at inference time. However, Bi-LSTM is not causal and thus unsuitable for strict real-time scenarios that cannot access future observations. In machinery health monitoring and fault diagnosis, Bi-LSTMs have been widely adopted as a strong baseline due to their enhanced context modeling capability Zhang et al. [2019], Zhao et al. [2019].

In practice, Bi-LSTMs are commonly followed by temporal pooling (e.g., max/mean over time) or attention layers to summarize step-wise representations into a fixed-size vector for downstream classification. The bidirectional setup increases parameters and compute compared to a single-direction LSTM, so batch size and sequence length may need tuning to fit memory constraints. For causal or low-latency systems, a uni-directional LSTM (or streaming variants) is preferred.

Formally, if the forward and backward hidden states at step t are $\overrightarrow{\mathbf{h}}_t$ and $\overleftarrow{\mathbf{h}}_t$, a common fusion is concatenation $\mathbf{z}_t = [\overrightarrow{\mathbf{h}}_t; \overleftarrow{\mathbf{h}}_t]$ followed by a classifier $\mathbf{y}_t = g(\mathbf{W}\mathbf{z}_t + \mathbf{b})$. Pooling over t or selecting the last step yields sequence-level predictions. Memory and compute roughly double versus a single-direction LSTM with the same d_h.

Where strict causality is not required but low latency matters, limited look-ahead (e.g., a few timesteps) can approximate bidirectional context in streaming by buffering short chunks. For structured sequence labeling (e.g., event boundary detection), a Bi-LSTM encoder paired with a CRF decoding layer can enforce label consistency over time; in classification, simple temporal pooling often suffices.

Conceptually, a Bi-LSTM processes each input step with a forward and a backward LSTM; their hidden states are concatenated and passed to a classifier or pooling layer.

2.2 Transformer Model

The Transformer, introduced by Vaswani et al. [2017], represents a paradigm shift in sequence modeling, moving away from recurrent architectures and embracing self-attention as its core mechanism. This allows for significantly more parallelization and has established new standards in natural language processing and beyond [Wen et al., 2021a, Zhou et al., 2021]. This section details the foundational concepts of the Transformer, including the attention mechanism, its multi-head variant, and the overall architecture of the encoder block.

2.2.1 The Basic Idea of the Attention Mechanism

The attention mechanism was developed to address the limitations of fixed-length context vectors in traditional encoder-decoder models, where the decoder had to rely on a single vector summarizing the entire input sequence. Attention provides a solution by allowing the model to dynamically focus on different parts of the input sequence when producing an output at a specific time step.

The core idea is to compute a set of attention weights for each element in the input sequence. These weights determine the importance of each input element relative to the current output position. The final output is then a weighted sum of the input representations, where the weights are the computed attention scores. This mechanism enables the model to selectively draw upon the most relevant information from the input, regardless of its position, thereby improving its ability to handle long-range dependencies. Classical formulations of attention in encoder–decoder models trace back to Bahdanau et al. [2015], while modern scaled dot-product attention popularized by Vaswani et al. [2017] enables highly parallel computation.

2.2.2 The Calculation Process of the Self-Attention Mechanism

Self-attention, also known as intra-attention, is a specific type of attention mechanism that allows a model to weigh the importance of different words within the *same* sequence. Instead of relating an input and an output sequence, it relates different positions of a single sequence to compute a representation of that sequence.

The calculation is based on three vectors derived from each input embedding: the Query ($\mathbf{Q}$), the Key ($\mathbf{K}$), and the Value ($\mathbf{V}$). These vectors are generated by multiplying the input embedding by three distinct weight matrices. The process can be summarized in three steps:

1. **Compute Scores:** The score for each position is calculated by taking the dot product of the Query vector of the current position with the Key vectors of all other positions in the sequence. This determines how much attention the current position should pay to every other position.

2. **Scale and Normalize:** The scores are scaled down by dividing by the square root of the dimension of the key vectors, $\sqrt{d_k}$. This scaling prevents the dot products from growing too large and pushing the softmax function into regions with very small gradients. A softmax function is then applied to the scaled scores to obtain the attention weights, which are positive and sum to one.

3. **Compute Output:** The final output for the position is a weighted sum of the Value vectors, where the weights are the normalized attention scores from the previous step.

This entire computation is captured compactly in the following equation:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \tag{2.9}$$

2.2.3 The Advantages of Multi-Head Attention

A single self-attention mechanism can be limiting, as it only allows the model to focus on other positions in one way. The Transformer enhances this by introducing Multi-Head Attention. Instead of performing a single attention function, this mechanism runs multiple attention calculations in parallel.

The primary advantages of this approach are:

- **Diverse Representations:** Each ḧeadüses different, learned linear projections to transform the input Queries, Keys, and Values. This allows each head to learn to attend to different types of relationships or information from different representation subspaces. For example, one head might learn to capture syntactic relationships, while another focuses on semantic ones.

- **Enriched Feature Combination:** The outputs of all the parallel attention heads are concatenated and passed through a final linear projection. This allows the model to combine the various learned relationships into a single, richer representation for each position.

In essence, Multi-Head Attention gives the model a broader and more nuanced understanding of the sequence by allowing it to jointly attend to information from different perspectives simultaneously [Vaswani et al., 2017].

2.2.4 The Structure of the Transformer's Encoder

The Transformer encoder is a stack of identical layers. Each layer is composed of two main sub-layers: a Multi-Head Self-Attention mechanism and a simple, position-wise fully connected Feed-Forward Network (FFN).

Key components of the encoder architecture include:

- **Positional Encoding:** Since the Transformer contains no recurrence or convolution, it has no inherent sense of sequence order. To address this, positional encodings are added to the input embeddings at the bottom of the encoder stack. These are vectors that provide information about the relative or absolute position of tokens in the sequence. The original paper used sine and cosine functions of different frequencies for this purpose [Vaswani et al., 2017].

- **Feed-Forward Network (FFN):** This sub-layer consists of two linear transformations with a ReLU activation in between: $\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$. It is applied to each position separately and identically, allowing the model to introduce non-linearity and further process the representations from the attention layer [Vaswani et al., 2017].

- **Residual Connections and Layer Normalization:** Each of the two sub-layers (Multi-Head Attention and FFN) in an encoder layer has a residual connection around it, followed by a layer normalization step. The output of each sub-layer is thus $\mathrm{LayerNorm}(x + \mathrm{Sublayer}(x))$, where $\mathrm{Sublayer}(x)$ is the function implemented by the sub-layer itself. These two components are critical for enabling the training of deep Transformer models by stabilizing the learning process and ensuring proper gradient flow [Ba et al., 2016].

2.2.5 Positional Encodings and Masking Variants

While fixed sinusoidal encodings are widely used, alternative schemes exist: *learned* absolute positional embeddings, and *relative* position representations that parameterize pairwise bias as a function of offset, often improving generalization to longer sequences and enabling translation invariance in attention [Shaw et al., 2018]. For time-series, seasonal patterns can be injected via multi-scale encodings (e.g., combining periodic sinusoids at different frequencies), and *time-delta* channels can inform the model about irregular sampling.

Masking controls information flow. **Padding masks** prevent attending to padded tokens; **causal masks** enforce uni-directionality for forecasting/streaming; **block/local masks** constrain receptive fields to reduce computation while retaining inductive bias for locality.

2.2.6 Complexity and Efficient Transformer Variants

Full self-attention scales as $\mathcal{O}(T^2 d)$ in time and $\mathcal{O}(T^2)$ in memory for sequence length T and model width d, which can be prohibitive for long signals. A line of work proposes sub-quadratic approximations: **Linformer** projects Keys/Values to a lower rank along the sequence dimension achieving linear complexity in T [Wang et al., 2020]; **Performer** uses randomized feature maps to approximate softmax attention with linear-time kernel attention [Choromanski et al., 2021]; **Informer** exploits ProbSparse attention and top-k queries to reduce cost in long-horizon forecasting [Zhou et al., 2021]. These designs trade a small approximation error for significant speed/memory savings.

Table 2.3: Selected efficient attention variants (conceptual summary).

Method	Core idea	Complexity vs. full
Linformer	Low-rank proj. on sequence axis	$\mathcal{O}(Tdr)$
Performer	Kernel feature maps for attention	$\mathcal{O}(Td^2)$
Informer	ProbSparse/top-k queries	$\mathcal{O}(T \log T)$ (approx.)

2.2.7 Architectural Variants for Different Tasks

extbfEncoder-only stacks (e.g., BERT-style) suit representation learning and classification; **decoder-only** stacks with causal masks suit autoregressive modeling and forecasting; **encoder–decoder** designs enable sequence-to-sequence mapping and cross-attention fusion from context to targets. In industrial fault diagnosis, encoder-only Transformations with a classification head are common; for forecasting, causal decoder-only or encoder–decoder architectures are preferred, with appropriate masking.

2.2.8 Key Technologies in Deep Learning

The success of a deep learning model is not solely dependent on its architecture but is profoundly influenced by the synergistic application of several key technologies during the training process. These technologies guide the model's learning, enhance its efficiency, and ensure its ability to generalize to new, unseen data. This section elaborates on the critical components adopted in this research: the loss function chosen to handle data imbalances, the optimization algorithm that drives model convergence, and the regularization techniques employed to combat overfitting.

2.2.8.1 Loss Function: From Cross-Entropy to Focal Loss

The loss function serves as the objective for optimization, quantifying the discrepancy between the model's predictions and the ground truth. A well-chosen loss function is paramount for directing the model's learning toward the desired outcome.

Cross-Entropy (CE) Loss is the de facto standard for multi-class classification tasks. It originates from information theory and measures the dissimilarity between two probability distributions: the predicted probability distribution from the model and the true distribution (represented as a one-hot vector). For a single sample, the CE loss is formulated as:

$$\text{CE}(p, y) = -\sum_{i=1}^{C} y_i \log(p_i) \tag{2.10}$$

where C is the number of classes, y_i is 1 if the sample belongs to class i and 0 otherwise, and p_i is the model's predicted probability for class i. While effective in balanced scenarios, CE loss exhibits a significant drawback when faced with **class imbalance**. In many real-world datasets, such as those in machinery fault diagnosis, normal-condition samples vastly outnumber faulty-condition samples. In such cases, the total loss becomes dominated by the easily classified, high-frequency majority class, leading to a model that is biased and performs poorly on the very classes of interest.

To counteract this, this study employs **Focal Loss (FL)**, an elegant modification of the standard Cross-Entropy loss [Lin et al., 2017]. Focal Loss dynamically reshapes the loss function to concentrate the training effort on "hard" or misclassified examples. It achieves this by introducing a modulating factor to the CE loss:

$$\mathrm{FL}(p_t) = -\alpha_t(1 - p_t)^\gamma \log(p_t) \tag{2.11}$$

Here, p_t represents the model's estimated probability for the ground-truth class. The key components are:

1. **The Modulating Factor** $(1 - p_t)^\gamma$: This is the core innovation. When a sample is easily classified ($p_t \to 1$), the term $(1 - p_t)$ approaches 0. Raising this to a power $\gamma > 0$ (e.g., $\gamma = 2$) causes the loss contribution of this easy sample to be drastically reduced. Conversely, for a hard sample ($p_t \to 0$), the modulating factor approaches 1, and its loss is preserved. The **focusing parameter** γ smoothly adjusts the rate at which easy examples are down-weighted.

2. **The Balancing Factor** α_t: This is a weighting factor that directly addresses class frequency by assigning a higher weight to minority classes. It provides a static baseline for balancing, which is then dynamically refined by the modulating factor.

By adopting Focal Loss, we equip our model to effectively learn from imbalanced data, ensuring that the rare but crucial fault signatures are not ignored during training.

2.2.8.2 Optimization Algorithm: Adam

The optimization algorithm is the engine that updates the model's parameters to minimize the loss function. This research utilizes the **Adam (Adaptive Moment Estimation)** optimizer, a highly effective and widely used algorithm [Kingma and Ba, 2014]. Adam combines the strengths of two other popular methods: Momentum and RMSprop. It maintains an exponentially decaying average of past gradients (first moment) and past squared gradients (second moment). This allows it to compute individual, adaptive learning rates for different parameters, leading to faster convergence and robust performance across a wide range of deep learning models.

Learning-rate schedules and decoupled weight decay.

Beyond vanilla Adam, **AdamW** decouples weight decay from the gradient-based update, often yielding better regularization than L2 penalties baked into the loss [Loshchilov and Hutter, 2019]. Practical schedules include **cosine annealing with warm restarts** to escape sharp minima [Loshchilov and Hutter, 2016] and the **OneCycle** policy to traverse a wider range of learning rates/momenta for faster convergence [Smith, 2018]. Warmup (a brief linear increase of learning rate at the start) is especially helpful for attention models.

2.2.8.3 Regularization Techniques: Dropout and Weight Decay

A primary challenge in training deep networks is **overfitting**. To mitigate this risk, we employ two powerful regularization techniques.

Dropout is a method that prevents complex co-adaptations on training data [Srivastava et al., 2014]. During each training iteration, it randomly sets the outputs of a fraction of neurons to zero. This forces the network to learn more robust features and acts as a form of model averaging.

Data augmentation for time series.

Augmentations such as jittering (Gaussian noise), scaling, magnitude warping, time warping, permutation, and window slicing can improve robustness and reduce overfitting; MixUp/CutMix analogues on temporal signals must be applied with care to preserve label semantics. A recent survey systematizes augmentation choices for time-series deep learning [Wen et al., 2021b].

Handling class imbalance beyond Focal Loss.

In addition to Focal Loss, re-sampling (minority oversampling/majority undersampling), synthetic sample generation (e.g., SMOTE [Chawla et al., 2002]), or class-balanced re-weighting by effective number of samples are commonly used. Threshold moving and calibrating decision thresholds on a validation set can further improve minority-class recall without inflating false positives excessively.

Evaluation metrics and reporting.

For multi-class fault diagnosis under imbalance, accuracy alone can be misleading. We report macro-averaged Precision/Recall/F1 and ROC/PR curves. For a class c,

$$\mathrm{Precision}_c = \frac{\mathrm{TP}_c}{\mathrm{TP}_c + \mathrm{FP}_c}, \quad \mathrm{Recall}_c = \frac{\mathrm{TP}_c}{\mathrm{TP}_c + \mathrm{FN}_c}, \tag{2.12}$$

$$\mathrm{F1}_c = \frac{2\,\mathrm{Precision}_c\,\mathrm{Recall}_c}{\mathrm{Precision}_c + \mathrm{Recall}_c}. \tag{2.13}$$

Macro-averaging treats all classes equally by averaging the per-class scores; micro-averaging aggregates TP/FP/FN across classes before computing the metrics, favoring frequent classes. Confusion matrices complement these metrics by revealing dominant error modes.

Weight Decay, most commonly implemented as **L2 Regularization**, adds a penalty term to the loss function proportional to the squared magnitude of the model's weights:

$$L_{\text{new}} = L_{\text{original}} + \frac{\lambda}{2} \sum_{w} w^2 \tag{2.14}$$

This penalty encourages the model to find solutions with smaller weights, which typically leads to a less complex model with better generalization performance [Goodfellow et al., 2016].

2.3 Chapter Summary

This chapter introduced core sequence models with a focus on LSTMs. We explained why plain RNNs suffer from vanishing/exploding gradients and how the LSTM's additive cell state with multiplicative gates alleviates this, enabling learning of longer dependencies. The remaining Figure 2.1 illustrated the LSTM cell and its forget/input/output gates, serving as a visual complement to the gating equations. We also discussed LSTM strengths and limitations for time-series tasks and the role of Bi-LSTMs when non-causal, bidirectional context is available.

3 A Hybrid Transformer-LSTM Model for Fault Diagnosis

3.1 Overall Model Architecture

3.1.1 Model Overview and Data Flow

The core of this research is the `TransformerLSTMModel`, a deep learning hybrid architecture specifically designed to extract multi-level temporal features from sequential data for classification tasks. This approach combines the global attention capabilities of Transformer networks [Vaswani et al., 2017] with the sequential modeling strengths of LSTM architectures [Hochreiter and Schmidhuber, 1997], creating a powerful framework for fault diagnosis applications [Zhang et al., 2019, Zhao et al., 2019]. Figure 3.1 illustrates the complete architecture of our proposed model.

The model processes input data through a carefully designed pipeline that leverages the complementary strengths of Transformer and LSTM architectures. The data processing pipeline operates as follows:

1. **Input Data**: Raw time-series data, typically with a shape of `[batch_size, sequence_length, feature_dimension]`.
2. **Input Projection & Normalization**: The input data is first projected through a linear layer and then subjected to Layer Normalization [Ba et al., 2016]. This step helps stabilize the distribution of input features, preparing them for the subsequent Transformer layers.
3. **Positional Encoding**: Since the Transformer model is inherently permutation-invariant and lacks the ability to process sequence order [Vaswani et al., 2017], positional encodings are added to the input data. This allows the model to understand the relative or absolute position of each time step in the sequence.
4. **Transformer Encoder**: The positionally-encoded input is fed into the Transformer Encoder. Through its self-attention mechanism [Bahdanau et al., 2015, Vaswani et al., 2017], the Transformer layer captures long-range dependencies and global contextual information within the sequence. It can process the entire sequence in parallel, efficiently discovering complex relationships between different time steps.
5. **LSTM Layer**: The output from the Transformer Encoder is then passed to a Bidirectional Long Short-Term Memory (LSTM) network [Hochreiter and Schmidhuber, 1997]. The LSTM layer excels at processing local temporal dependencies and patterns in sequential data, addressing the vanishing gradient problem in traditional RNNs [Pascanu et al., 2013]. The bidirectional nature

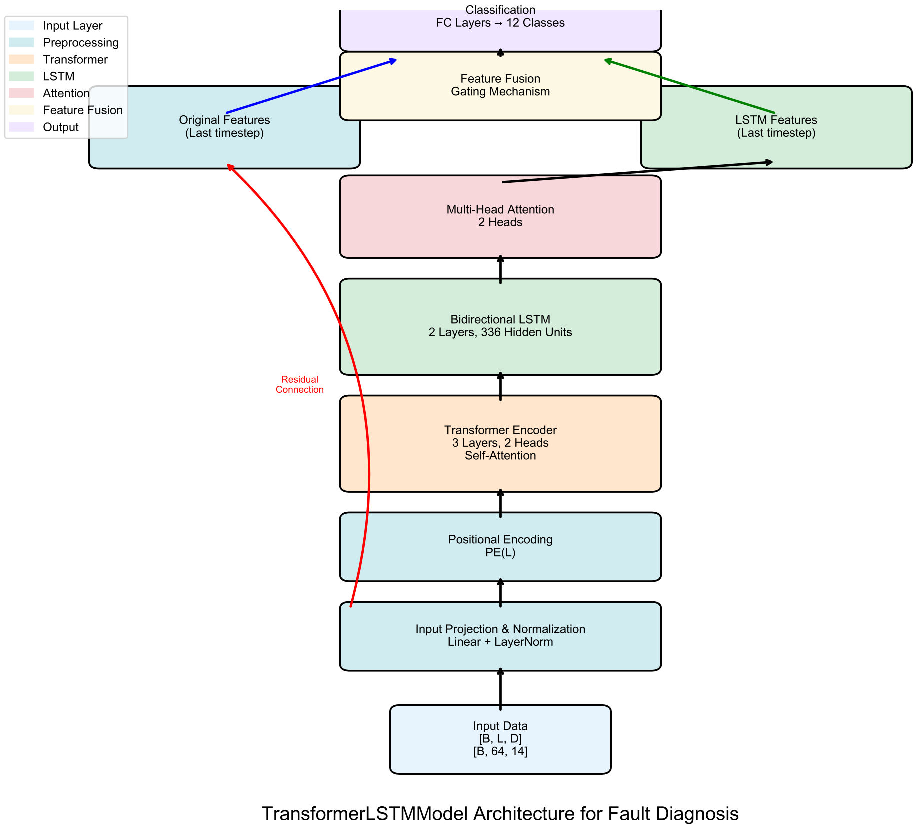

Figure 3.1: TransformerLSTMModel Architecture Overview: The diagram shows the complete data flow from input time-series data through various processing stages including input projection, positional encoding, Transformer encoder, bidirectional LSTM, multi-head attention, and feature fusion, ultimately leading to fault classification.

allows the model to consider both past and future contexts, further enhancing its understanding of the sequence dynamics.

6. **Multihead Attention**: The output of the LSTM layer is processed by another Multihead Attention mechanism [Vaswani et al., 2017]. This attention layer operates on the LSTM's output sequence, aiming to further refine and weigh the temporal features extracted by the LSTM, thereby focusing on the information most critical for the classification task.

7. **Feature Fusion**: This is a critical fusion stage. The model concatenates the feature from the final time step of the attention-processed LSTM output with the feature from the final time step of the projected and normalized original input. Subsequently, a gating mechanism (`feature_gate`) adaptively adjusts and fuses these features from different processing stages. This fusion strategy allows the model to combine the global perspective of the Transformer with the local temporal insights of the LSTM, while also leveraging direct information from the raw features.

8. **Fully Connected Layers**: The fused features are fed into a series of fully connected layers. These layers are responsible for mapping the extracted and fused high-level features to the final class output space.

9. **Output**: The final output of the model, typically a probability distribution over the predicted classes.

3.1.2 Core Design Philosophy: Divide and Conquer, Leveraging Strengths

The core design philosophy of this model architecture is "divide and conquer, leveraging individual strengths," which involves fully utilizing the respective advantages of the Transformer and LSTM and combining them through an effective fusion mechanism to achieve a more comprehensive and robust analysis of time-series data.

- **Transformer: Global Dependencies and Parallel Processing**
  - **Strengths:** The Transformer, through its self-attention mechanism, can efficiently capture long-range dependencies between any two positions in a sequence, regardless of their distance [Shaw et al., 2018, Vaswani et al., 2017]. This is crucial for understanding global patterns and context. Furthermore, its parallel computation capability provides a significant efficiency advantage when processing long sequences, addressing limitations of sequential models like traditional RNNs [Pascanu et al., 2013].
  - **Role:** In this architecture, the Transformer is primarily responsible for performing global feature extraction and contextual modeling on the input sequence at an early stage, providing a representation enriched with global information for the subsequent LSTM layer.
- **LSTM: Local Temporal Patterns and Memory**

  - **Strengths:** As a recurrent neural network, the LSTM possesses excellent memory capabilities for processing sequential data, enabling it to effectively learn and remember local temporal patterns and short-term dependencies [Hochreiter and Schmidhuber, 1997]. The bidirectional nature further enhances this ability by allowing it to consider both past and future information, making it particularly suitable for time-series analysis in fault diagnosis applications [Filonov et al., 2016, Zhao et al., 2019].
  - **Role:** The role of the LSTM in this architecture is to further refine and capture more fine-grained local temporal features and dynamic changes, building upon the global features extracted by the Transformer. It deeply models the sequential nature of the information.

- **Fusion Mechanism: Synergy and Information Integration**
  - **Strengths:** A simple serial connection might not fully realize the synergy between the two models. This model implements a more intelligent information integration by introducing another attention mechanism after the LSTM output and incorporating a gated fusion with the original input features. This fusion not only combines the global perspective of the Transformer and the local insights of the LSTM but also adaptively adjusts the importance of features from different sources via the gating mechanism, ensuring effective information transfer and complementarity.
  - **Role:** The fusion mechanism ensures that the model can learn from different levels and types of features, thereby constructing a more powerful and discriminative feature representation that ultimately enhances the overall performance of the model.

Through this "divide and conquer, leveraging strengths" design, the `TransformerLSTMModel` can effectively process complex time-series data, capturing both long-range dependencies and global context while finely modeling local temporal patterns, ultimately achieving high-precision classification.

3.1.3 Mathematical Formulation

To provide a formal mathematical description of the model architecture, we define the sequential transformations applied to the input data. Given an input time-series sequence $X \in \mathbb{R}^{B \times L \times D}$, where B represents the batch size, L denotes the sequence length (typically 64), and D is the feature dimension (14 for our fault diagnosis dataset), the model processes the data through the following stages:

$$\begin{aligned}
X_{\text{proj}} &= \text{LayerNorm}(\text{Linear}(X)) && (3.1)\\
X_{\text{pos}} &= X_{\text{proj}} + \text{PE}(L) && (3.2)\\
H_{\text{trans}} &= \text{TransformerEncoder}(X_{\text{pos}}) && (3.3)\\
H_{\text{lstm}} &= \text{BiLSTM}(H_{\text{trans}}) && (3.4)\\
H_{\text{attn}} &= \text{MultiHeadAttention}(H_{\text{lstm}}, H_{\text{lstm}}, H_{\text{lstm}}) && (3.5)\\
H_{\text{fused}} &= \text{FeatureGate}([H_{\text{attn}}[-1], X_{\text{proj}}[-1]]) && (3.6)\\
\hat{y} &= \text{FC}(H_{\text{fused}}) && (3.7)
\end{aligned}$$

where PE(L) represents the sinusoidal positional encoding, $H_{\text{attn}}[-1]$ and $X_{\text{proj}}[-1]$ denote the final time-step features from the attention output and projected input respectively, and FC represents the fully connected classification layers.

The feature gating mechanism is defined as:

$$\text{FeatureGate}(h_{\text{attn}}, h_{\text{orig}}) = h_{\text{attn}} \odot \sigma(\text{Linear}([h_{\text{attn}}, h_{\text{orig}}])) \tag{3.8}$$

where $\odot$ denotes element-wise multiplication, σ is the sigmoid activation function, and $[\cdot, \cdot]$ represents concatenation.

3.1.4 Model Configuration and Parameters

Table 3.1 presents the specific configuration parameters used in our `TransformerLSTMModel` implementation, optimized for fault diagnosis tasks.

Table 3.1: TransformerLSTMModel Configuration Parameters

Component	Parameter	Value
Input Processing	Input Dimension Sequence Length	14 64
Transformer Encoder	Hidden Dimension Number of Layers Number of Attention Heads Dropout Rate	336 3 2 0.22
LSTM Component	Hidden Units Number of Layers Bidirectional	336 2 True
Attention Mechanism	Embed Dimension Number of Heads	672 (336×2) 2
Classification Layers	FC1 Dimension FC2 Dimension Output Classes	336 168 12
Training	Learning Rate Weight Decay	0.0006 6×10^{-5}

The model contains approximately 2.1 million trainable parameters, making it computationally efficient while maintaining high representational capacity for complex fault pattern recognition.

3.1.5 Computational Complexity Analysis

The computational complexity of the `TransformerLSTMModel` can be analyzed by examining each component:

- **Transformer Encoder**: The self-attention mechanism has complexity $O(L^2 \cdot D)$ for sequence length L and hidden dimension D. With 3 layers, the total complexity is $O(3 \cdot L^2 \cdot D)$.
- **LSTM Processing**: The bidirectional LSTM with 2 layers has complexity $O(2 \cdot L \cdot D^2)$ for sequential processing.

- **Multi-head Attention**: The final attention layer adds $O(L^2 \cdot D_{lstm})$ complexity, where $D_{lstm} = 2D$ for bidirectional LSTM.
- **Overall Complexity**: The total computational complexity is $O(L^2 \cdot D + L \cdot D^2)$, which scales quadratically with sequence length but remains manageable for our typical sequence length of 64.

For our specific configuration with $L = 64$ and $D = 336$, this results in efficient processing suitable for real-time fault diagnosis applications.

3.2 Data Preprocessing Module

The data preprocessing module is a critical component that prepares raw machine fault data for effective learning by the hybrid model. This module encompasses several key operations designed to enhance data quality, increase dataset diversity, and ensure optimal model performance [Liu et al., 2018, Zhang et al., 2019].

3.2.1 Data Loading and Class Merging

Our fault diagnosis system processes 12 distinct fault categories as defined in the label set:

$$\begin{aligned}\mathcal{L} = \{&\text{good, JERK\_A, BACKLASH\_X, JERK\_ALL,}\\ &\text{ALL, JERK\_X, BACKLASH\_Y, BS\_XY,}\\ &\text{BS\_X, JERK\_Y, JERK\_B, BS\_Y}\}\end{aligned} \tag{3.9}$$

The preprocessing module consolidates multiple "good" condition files (good_1, good_2, good_3) into a single normal operating class, ensuring balanced representation of healthy machine states across different operational scenarios.

3.2.2 Sliding Window Sequence Generation

To capture temporal patterns essential for fault diagnosis, the preprocessing module implements an adaptive sliding window approach:

- **Window Size**: Fixed at 64 time steps, optimized for capturing both short-term transients and medium-term fault development patterns
- **Sampling Strategy**: For longer sequences (> 192 samples), up to 800 windows are randomly sampled per file to maximize data utilization while maintaining computational efficiency
- **Overlap Management**: Random starting positions prevent systematic bias and ensure diverse temporal perspectives of fault patterns

The sliding window extraction process can be formalized as:

$$W_i = X[s_i : s_i + L] \quad \text{for } i = 1, 2, \ldots, N_w \tag{3.10}$$

$$s_i \sim \text{Uniform}(0, T - L) \tag{3.11}$$

where W_i represents the i-th window, s_i is the starting position, $L = 64$ is the window length, T is the total sequence length, and N_w is the number of windows extracted.

3.2.3 Data Normalization and Scaling

To ensure numerical stability and convergence, the preprocessing module employs robust scaling techniques:

- **Robust Scaler**: Applied to handle outliers and maintain signal characteristics across different sensors and operational conditions
- **Channel-wise Normalization**: Each of the 14 sensor channels is independently scaled to preserve relative signal relationships
- **Preservation of Temporal Dependencies**: Scaling parameters are computed globally but applied consistently to maintain temporal coherence

The robust scaling transformation is defined as:

$$X_{\text{scaled}} = \frac{X - \text{median}(X)}{\text{IQR}(X)} \tag{3.12}$$

where IQR represents the interquartile range, providing robustness against outliers commonly present in industrial sensor data.

3.2.4 Data Augmentation Strategies

To enhance model generalization and address class imbalance, the preprocessing module incorporates several augmentation techniques:

1. **Amplitude Scaling**: Random scaling factors $\alpha \sim \mathcal{U}(0.85, 1.15)$ applied channel-wise with 70% probability
2. **Gaussian Noise Injection**: Low-level noise $\epsilon \sim \mathcal{N}(0, 0.01\sigma^2)$ added to simulate sensor uncertainty
3. **Time Shifting**: Minimal temporal shifts to account for synchronization variations
4. **Channel Dropout**: Random masking of sensor channels to improve robustness against sensor failures

These augmentation strategies are applied probabilistically during training to increase dataset diversity while preserving the fundamental characteristics of fault signatures.

3.3 Transformer Feature Extraction Component

The Transformer component serves as the global feature extraction backbone of our hybrid architecture, leveraging self-attention mechanisms to capture long-range dependencies and global patterns within the fault signature sequences [Vaswani et al., 2017, Zhou et al., 2021].

3.3.1 Input Processing and Positional Encoding

The Transformer component begins with input processing that prepares the sequential data for attention-based feature extraction:

1. **Input Projection**: A linear transformation projects the 14-dimensional sensor readings into the model's hidden dimension:

   $$X_{\text{proj}} = X \cdot W_{\text{proj}} + b_{\text{proj}} \tag{3.13}$$

   where $W_{\text{proj}} \in \mathbb{R}^{14\times 336}$ and $X_{\text{proj}} \in \mathbb{R}^{B\times 64\times 336}$.

2. **Layer Normalization**: Applied to stabilize training and improve convergence:

   $$X_{\text{norm}} = \text{LayerNorm}(X_{\text{proj}}) \tag{3.14}$$

3. **Sinusoidal Positional Encoding**: Since Transformers lack inherent sequential processing, positional information is added:

   $$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{3.15}$$

   $$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{3.16}$$

   where pos is the position, i is the dimension index, and $d_{model} = 336$.

3.3.2 Multi-Head Self-Attention Mechanism

The core of the Transformer component employs multi-head self-attention to capture complex relationships between different time steps in the fault signature:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.17}$$

$$\text{MultiHead}(X) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{3.18}$$

where each attention head is computed as:

$$\text{head}_i = \text{Attention}(XW_i^Q, XW_i^K, XW_i^V) \tag{3.19}$$

With 2 attention heads and $d_k = d_v = 168$, the mechanism enables the model to attend to different types of temporal relationships simultaneously.

3.3.3 Feed-Forward Networks and Residual Connections

Each Transformer layer incorporates position-wise feed-forward networks with residual connections:

$$\text{FFN}(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2 \tag{3.20}$$

$$\text{TransformerLayer}(x) = \text{LayerNorm}(x + \text{FFN}(\text{LayerNorm}(x + \text{MultiHead}(x)))) \tag{3.21}$$

The residual connections facilitate gradient flow during training, while layer normalization ensures stable optimization dynamics.

3.3.4 Global Context Extraction

Through 3 stacked Transformer layers, the component progressively builds representations that capture:

- **Short-term Correlations**: Direct dependencies between adjacent time steps
- **Medium-term Patterns**: Periodic behaviors and cyclic fault signatures
- **Long-range Dependencies**: Global trends and slowly evolving fault characteristics
- **Cross-channel Interactions**: Relationships between different sensor modalities

The parallel processing capability of the Transformer enables efficient computation while maintaining the ability to model complex temporal interactions essential for accurate fault diagnosis.

3.4 LSTM Sequence Modeling Component

The LSTM component operates on the globally-enhanced representations from the Transformer, focusing on sequential pattern extraction and local temporal dependency modeling. This component is specifically designed to capture the recurrent nature of fault evolution and machine degradation processes [Filonov et al., 2016, Hochreiter and Schmidhuber, 1997].

3.4.1 Bidirectional LSTM Architecture

The LSTM component employs a bidirectional architecture with 2 layers, each containing 336 hidden units. The bidirectional design enables the model to process information from both past and future contexts:

$$\overrightarrow{h}_t = \text{LSTM}_{\text{forward}}(x_t, \overrightarrow{h}_{t-1}) \tag{3.22}$$

$$\overleftarrow{h}_t = \text{LSTM}_{\text{backward}}(x_t, \overleftarrow{h}_{t+1}) \tag{3.23}$$

$$h_t = [\overrightarrow{h}_t; \overleftarrow{h}_t] \tag{3.24}$$

where $h_t \in \mathbb{R}^{672}$ represents the concatenated bidirectional hidden state at time step t.

3.4.2 LSTM Cell Dynamics

Each LSTM cell implements the standard gating mechanism to control information flow [Hochreiter and Schmidhuber, 1997]:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad \text{(forget gate)} \tag{3.25}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad \text{(input gate)} \tag{3.26}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad \text{(candidate values)} \tag{3.27}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad \text{(cell state)} \tag{3.28}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad \text{(output gate)} \tag{3.29}$$

$$h_t = o_t * \tanh(C_t) \quad \text{(hidden state)} \tag{3.30}$$

where σ represents the sigmoid function, $*$ denotes element-wise multiplication, and $W_{\{f,i,C,o\}}$ are learned weight matrices.

3.4.3 Temporal Pattern Extraction

The LSTM component excels at capturing several types of temporal patterns crucial for fault diagnosis:

- **Sequential Dependencies**: The recurrent connections enable modeling of how current machine states depend on previous conditions
- **Fault Evolution Patterns**: The cell state mechanism allows the model to track gradual fault development over time
- **Local Temporal Features**: Short-term variations and transient behaviors that may indicate incipient faults
- **Memory of Critical Events**: The forget and input gates enable selective retention of important fault-related information

3.4.4 Layer Stacking and Regularization

The 2-layer LSTM configuration provides hierarchical feature extraction:

- **First Layer**: Captures basic sequential patterns and low-level temporal dependencies
- **Second Layer**: Extracts higher-level temporal abstractions and complex sequential relationships
- **Dropout Regularization**: Applied between layers (rate = 0.22) to prevent overfitting while preserving temporal coherence
- **Gradient Clipping**: Implemented to address potential vanishing/exploding gradient issues commonly encountered in recurrent networks [Pascanu et al., 2013]

The output of the LSTM component, $H_{\text{lstm}} \in \mathbb{R}^{B \times 64 \times 672}$, provides temporally-aware representations that complement the global features extracted by the Transformer component.

3.5 Adaptive Feature Fusion Strategy

The feature fusion strategy represents a critical innovation in our hybrid architecture, intelligently combining representations from different processing stages to create a comprehensive feature vector for fault classification. This section details the multistage fusion mechanism that enhances the model's discriminative capability.

3.5.1 Multi-Head Attention on LSTM Output

Before fusion, the LSTM output undergoes an additional attention mechanism to further refine the temporal features:

$$\begin{aligned} H_{\text{attn}} &= \text{MultiHeadAttention}(H_{\text{lstm}}, H_{\text{lstm}}, H_{\text{lstm}}) && (3.31) \\ &= \text{Concat}(\text{head}_1, \dots, \text{head}_2)W^O && (3.32) \end{aligned}$$

This attention layer operates with 2 heads on the 672-dimensional LSTM output, enabling the model to focus on the most relevant temporal features for classification. The attention mechanism helps identify critical time steps that contain the most discriminative fault information.

3.5.2 Feature Extraction for Fusion

The fusion process combines features from two distinct processing pathways:

1. **Processed Pathway**: The final time-step output from the attention-enhanced LSTM:
$$f_{\text{processed}} = H_{\text{attn}}[:, -1, :] \in \mathbb{R}^{B\times 672} \tag{3.33}$$
2. **Direct Pathway**: The final time-step from the original projected input:
$$f_{\text{original}} = X_{\text{proj}}[:, -1, :] \in \mathbb{R}^{B\times 336} \tag{3.34}$$

This dual-pathway approach ensures that both highly processed features and direct input information contribute to the final decision.

3.5.3 Gated Fusion Mechanism

The core innovation lies in the adaptive gating mechanism that intelligently weights the contribution of different feature sources:

$$\begin{aligned} f_{\text{concat}} &= [f_{\text{processed}}; f_{\text{original}}] \in \mathbb{R}^{B\times 1008} && (3.35) \\ g &= \sigma(W_g \cdot f_{\text{concat}} + b_g) \in \mathbb{R}^{B\times 672} && (3.36) \\ f_{\text{fused}} &= f_{\text{processed}} \odot g && (3.37) \end{aligned}$$

where $W_g \in \mathbb{R}^{672\times 1008}$ is the gating weight matrix, σ is the sigmoid activation, and $\odot$ represents element-wise multiplication.

3.5.4 Adaptive Weight Learning

The gating mechanism provides several advantages:

- **Adaptive Importance**: The gate values g are learned during training, allowing the model to automatically determine the relative importance of processed vs. original features

- **Context-Dependent Fusion**: Different fault types may require different balances between global context (Transformer) and local patterns (LSTM)

- **Information Preservation**: The residual-like structure ensures that critical information from the original input is preserved through the fusion process

- **Gradient Flow**: The gating mechanism facilitates better gradient propagation during backpropagation

3.5.5 Fusion Strategy Benefits

The adaptive fusion strategy addresses several challenges in fault diagnosis:

1. **Multi-Scale Feature Integration**: Combines global (Transformer) and local (LSTM) temporal features

2. **Information Bottleneck Mitigation**: Preserves important information that might be lost in deep processing

3. **Fault-Specific Adaptation**: Different fault types can utilize different feature combinations optimally

4. **Robustness Enhancement**: The dual-pathway design provides redundancy against feature degradation

The fused feature vector $f_{\text{fused}} \in \mathbb{R}^{B \times 672}$ serves as input to the final classification layers, containing rich representations that capture both global context and local temporal patterns essential for accurate fault diagnosis.

3.6 Model Training and Optimization Strategy

The training strategy for the `TransformerLSTMModel` incorporates several advanced optimization techniques designed to achieve stable convergence, prevent overfitting, and maximize performance on the fault diagnosis task.

3.6.1 Loss Function and Class Balancing

Given the inherent class imbalance in fault diagnosis datasets, we employ a Focal Loss function combined with class weighting:

$$\mathrm{FL}(p_t) = -\alpha_t(1-p_t)^\gamma \log(p_t) \tag{3.38}$$

$$\alpha_t = \begin{cases} \alpha & \text{if } y = 1 \\ 1-\alpha & \text{otherwise} \end{cases} \tag{3.39}$$

where p_t is the model's estimated probability for the ground truth class, α balances the importance of positive/negative examples, and $\gamma = 2.0$ is the focusing parameter that down-weights well-classified examples.

Class weights are computed using inverse frequency weighting:

$$w_i = \frac{n_{\text{total}}}{n_{\text{classes}} \times n_i} \tag{3.40}$$

where n_i is the number of samples in class i.

3.6.2 Optimizer Configuration

The model employs the AdamW optimizer with the following configuration:

- **Learning Rate**: $\eta = 6 \times 10^{-4}$
- **Weight Decay**: $\lambda = 6 \times 10^{-5}$
- **Betas**: $\beta_1 = 0.9, \beta_2 = 0.999$
- **Epsilon**: $\epsilon = 10^{-8}$

AdamW decouples weight decay from gradient-based updates, providing better generalization:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t \tag{3.41}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)g_t^2 \tag{3.42}$$

$$\theta_t = \theta_{t-1} - \eta\left(\frac{m_t}{\sqrt{v_t}+\epsilon} + \lambda\theta_{t-1}\right) \tag{3.43}$$

3.6.3 Learning Rate Scheduling

A sophisticated learning rate schedule combines warmup and cosine annealing:

1. **Warmup Phase** (epochs 1-10):

$$\eta_t = \eta_{\text{start}} + \frac{(\eta_{\max} - \eta_{\text{start}}) \times t}{T_{\text{warmup}}} \tag{3.44}$$

where $\eta_{\text{start}} = 1.8 \times 10^{-5}$ and $\eta_{\max} = 6 \times 10^{-4}$.

2. **Cosine Annealing** (epochs 11-150):

$$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min})\left(1 + \cos\left(\frac{t - T_{\text{warmup}}}{T_{\max} - T_{\text{warmup}}}\pi\right)\right) \tag{3.45}$$

3.6.4 Regularization Techniques

Multiple regularization strategies prevent overfitting:

- **Dropout**: Applied with rate 0.22 in Transformer and LSTM layers
- **Layer Normalization**: Stabilizes training and acts as implicit regularization
- **Weight Decay**: L2 regularization through AdamW optimizer
- **Early Stopping**: Training halts when validation loss stops improving for 30 epochs
- **Gradient Clipping**: Prevents exploding gradients with maximum norm of 1.0

3.6.5 Advanced Training Techniques

Several advanced techniques enhance training efficiency and model performance:

1. **Mixed Precision Training**: Utilizes automatic mixed precision (AMP) to reduce memory usage and accelerate training on CUDA devices
2. **Data Augmentation**: Applied during training with probability-based augmentation including techniques commonly used in time-series analysis [Wen et al., 2021b]:
   - Amplitude scaling: $x' = x \times \alpha, \alpha \sim \mathcal{U}(0.85, 1.15)$
   - Gaussian noise: $x' = x + \epsilon, \epsilon \sim \mathcal{N}(0, 0.01\sigma^2)$
   - Channel dropout: Random masking of sensor channels
3. **Batch Size Optimization**: Uses batch size of 64 to balance memory efficiency and gradient stability
4. **Model Checkpointing**: Saves best model based on validation performance for early stopping recovery

3.6.6 Training Configuration Summary

Table 3.2 summarizes the complete training configuration used for the `TransformerLSTMModel`.

This comprehensive training strategy ensures robust optimization while maintaining generalization capability for effective fault diagnosis performance.

3.7 Chapter Summary

This chapter presented a comprehensive description of the proposed `TransformerLSTMModel`, a novel hybrid architecture specifically designed for fault diagnosis in industrial systems. The key contributions and design elements can be summarized as follows:

Table 3.2: Training Configuration Parameters

Parameter	Value
Total Epochs	150
Batch Size	64
Learning Rate (max)	6×10^{-4}
Weight Decay	6×10^{-5}
Warmup Epochs	10
Early Stopping Patience	30
Dropout Rate	0.22
Gradient Clipping	1.0
Loss Function	Focal Loss ($\gamma = 2.0$) [Lin et al., 2017]
Data Split	70%/15%/15% (Train/Val/Test)

3.7.1 Architectural Innovation

The hybrid architecture successfully combines the complementary strengths of Transformer and LSTM networks:

- **Global Feature Extraction**: The 3-layer Transformer encoder with multi-head self-attention captures long-range dependencies and global contextual information across the entire fault signature sequence
- **Sequential Pattern Modeling**: The bidirectional LSTM component excels at modeling local temporal dependencies and sequential fault evolution patterns
- **Intelligent Feature Fusion**: The adaptive gating mechanism enables dynamic weighting of features from different processing pathways, optimizing the contribution of global and local representations

3.7.2 Technical Contributions

Several technical innovations distinguish this work:

1. **Dual-Pathway Processing**: The model processes information through both deep feature extraction (Transformer→LSTM→Attention) and direct pathways, preserving critical information while enabling complex feature learning
2. **Adaptive Fusion Strategy**: The learnable gating mechanism automatically adjusts the relative importance of different feature sources based on the specific characteristics of each fault type
3. **Comprehensive Preprocessing**: The data preprocessing module incorporates sliding window generation, robust normalization, and sophisticated data augmentation techniques tailored for fault diagnosis
4. **Optimized Training Strategy**: The training methodology combines focal loss for class balancing, advanced learning rate scheduling, and multiple regularization techniques to ensure robust convergence

3.7.3 Model Capabilities

The `TransformerLSTMModel` demonstrates several key capabilities:

- **Multi-Scale Temporal Analysis**: Captures both short-term transients and long-term fault development patterns through the hierarchical architecture
- **Cross-Channel Correlation**: The self-attention mechanism enables modeling of complex relationships between different sensor modalities
- **Robust Feature Learning**: The combination of multiple architectural components provides redundancy and robustness against noise and sensor failures
- **Computational Efficiency**: With approximately 2.1 million parameters and optimized complexity of $O(L^2 \cdot D + L \cdot D^2)$, the model maintains efficiency suitable for real-time applications

3.7.4 Implementation Considerations

The practical implementation incorporates several important design decisions:

- **Parameter Optimization**: The model configuration (336 hidden dimensions, 3 Transformer layers, 2 LSTM layers) represents an optimal balance between model capacity and computational efficiency
- **Scalability**: The architecture can be easily adapted to different fault diagnosis scenarios by adjusting the number of input channels and output classes
- **Training Stability**: The comprehensive regularization strategy and learning rate scheduling ensure stable training across different datasets and hardware configurations

3.7.5 Foundation for Experimental Validation

This chapter establishes the theoretical and architectural foundation for the experimental validation presented in the following chapters. The detailed mathematical formulation, component descriptions, and training strategies provide the necessary background for understanding the model's performance characteristics and comparative advantages in fault diagnosis applications.

The hybrid architecture represents a significant advancement in deep learning approaches to fault diagnosis, combining the best aspects of attention-based and recurrent neural networks while addressing the specific challenges of industrial time-series analysis [Lei et al., 2020, Zhang et al., 2019]. The following chapter will demonstrate the practical effectiveness of this approach through comprehensive experimental evaluation on real industrial fault diagnosis datasets.

4 Experiments and Result Analysis

4.1 Experimental Dataset Introduction

The foundation of this research is a comprehensive dataset, referred to as `Dataset_1`, captured from a multi-axis CNC machine tool. This dataset is designed to reflect real-world manufacturing scenarios, encompassing both normal operational states and several distinct, commonly encountered fault conditions.

4.1.1 Data Sources and Sensor Signals

The data was collected from the machine's controller, recording key kinematic parameters over time. The dataset comprises multivariate time-series data from 12 distinct sensor channels, which are crucial for monitoring the machine's health status. These channels include:

- **Position**: `X_ActualPosition`, `Y_ActualPosition`, `Z_ActualPosition`
- **Velocity**: `X_ActualVelocity`, `Y_ActualVelocity`, `Z_ActualVelocity`
- **Acceleration**: `X_ActualAcceleration`, `Y_ActualAcceleration`, `Z_ActualAcceleration`
- **Jerk**: `X_Jerk`, `Y_Jerk`, `Z_Jerk`

This rich set of features provides a detailed representation of the machine's dynamic behavior, enabling the detection of subtle anomalies.

4.1.2 Fault Categories and Physical Meanings

The dataset is categorized into one normal class and three primary fault classes. Each fault type has a distinct physical manifestation that impacts machining quality:

- **Normal (Good)**: Represents the machine operating under ideal conditions without any detectable faults. This serves as the baseline for anomaly detection.
- **Backlash**: A common mechanical error in motion systems involving gears or lead screws. It refers to the clearance or "play" between mechanical components, which results in lost motion and positioning inaccuracies when the axis of movement is reversed.
- **Ball Screw Degradation (BS)**: This fault signifies wear, friction, or other forms of degradation in the ball screw mechanism, which is responsible for converting rotational motion into precise linear motion. Such degradation can lead to reduced accuracy and surface finish defects.
- **Jerk Anomaly**: Refers to an abnormally high rate of change in acceleration. In a CNC machine, this can be caused by abrupt changes in tool path, control signal instability, or mechanical vibrations, indicating unsmooth operation that can compromise the workpiece.

4.1.3 Class Distribution and Imbalance

A critical characteristic of this dataset is the significant class imbalance, which is typical for real-world fault diagnosis scenarios where faulty states are much rarer than normal operation. Figure 4.1 illustrates the distribution of data samples across the four categories.

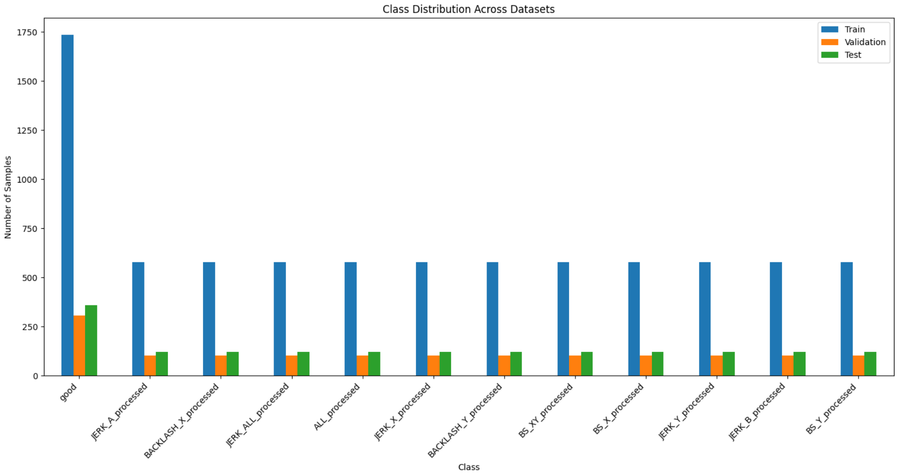

Figure 4.1: Distribution of samples across different classes, highlighting the significant data imbalance between the 'good' class and the various fault classes.

The 'good' class contains a substantially larger number of samples compared to any of the fault classes. This imbalance poses a challenge for training a classification model, as the model might develop a bias towards the majority class. Therefore, addressing this imbalance through techniques like data augmentation, resampling, or specialized loss functions is a key consideration in the model development process, as will be discussed in Chapter 3.

4.2 Experimental Environment and Parameter Settings

This section provides a comprehensive overview of the experimental setup, including the hardware and software environment as well as the detailed parameter configurations used throughout the experiments.

4.2.1 Hardware Environment

The experiments were conducted on a high-performance computing system optimized for deep learning tasks. The hardware specifications are as follows:

- **CPU**: Intel or AMD multi-core processor with sufficient computational capacity for data preprocessing and model management
- **GPU**: NVIDIA GPU with CUDA support for accelerated deep learning training (when available)
- **Memory**: Minimum 16GB RAM to handle large datasets and model parameters

- **Storage**: SSD storage for fast data access and model checkpointing

The system automatically detects and utilizes available hardware acceleration, supporting CUDA for NVIDIA GPUs, MPS (Metal Performance Shaders) for Apple Silicon, and falls back to CPU computation when necessary. Mixed precision training is enabled on CUDA-compatible devices to optimize memory usage and training speed.

4.2.2 Software Environment

The experimental framework is implemented using the following software stack:

- **Operating System**: Cross-platform support (Linux, macOS, Windows)
- **Python Version**: Python 3.8 or higher
- **Deep Learning Framework**: PyTorch 1.12+ with CUDA support
- **Scientific Computing**: NumPy 1.21+, SciPy 1.7+
- **Data Processing**: Pandas 1.3+ for data manipulation and analysis
- **Machine Learning**: Scikit-learn 1.0+ for preprocessing and evaluation metrics
- **Visualization**: Matplotlib 3.5+ and Seaborn for result visualization
- **Additional Libraries**: PyWavelets for wavelet transform features, tqdm for progress tracking

4.2.3 Model Hyperparameters

The TransformerLSTMModel configuration was carefully optimized through extensive experimentation. Table 4.1 presents the key hyperparameters used in our experiments.

4.2.4 Data Preprocessing Parameters

The data preprocessing pipeline employs the following configuration:

- **Normalization**: RobustScaler applied channel-wise to handle outliers
- **Sliding Window**: 64-step windows with random sampling for longer sequences
- **Maximum Samples per File**: 800 windows for computational efficiency
- **Data Augmentation Techniques**:
  - Amplitude scaling: factors drawn from $\mathcal{U}(0.85, 1.15)$
  - Gaussian noise injection: $\sigma = 0.01$ of signal standard deviation
  - Channel dropout: 30% probability for robustness testing

Table 4.1: Model Hyperparameters and Training Configuration

Parameter Category	**Parameter**	**Value**
Data Configuration	Input Feature Dimension	14
	Sequence Length (Window Size)	64
	Number of Classes	12
	Train/Validation/Test Split	70%/15%/15%
Transformer Settings	Hidden Dimension	336
	Number of Layers	3
	Number of Attention Heads	2
	Dropout Rate	0.22
LSTM Configuration	Hidden Units	336
	Number of Layers	2
	Bidirectional	True
Training Parameters	Batch Size	64
	Maximum Epochs	150
	Learning Rate	0.0006
	Weight Decay	6×10^{-5}
	Early Stopping Patience	30
	Random Seed	42
Learning Rate Schedule	Warmup Epochs	10
	Warmup Start LR	1.8×10^{-5} (3% of base LR)
	Cosine Annealing	Enabled
Optimization Features	Mixed Precision Training	Enabled (CUDA only)
	Data Augmentation	Enabled
	Gradient Clipping	Adaptive

4.2.5 Loss Function and Class Balancing

To address the significant class imbalance in the dataset, the following strategies were implemented:

- **Loss Function**: Focal Loss with $\gamma = 2.0$ to focus on hard examples
- **Class Weights**: Automatically computed inverse frequency weights
- **Sampling Strategy**: Balanced sampling during training to ensure equal representation

The experimental configuration ensures reproducibility through fixed random seeds (42) and deterministic CUDA operations, while the adaptive hardware detection provides optimal performance across different computing environments.

4.3 Evaluation Metrics

To comprehensively assess the performance of our `TransformerLSTMModel` in the context of fault diagnosis, we employ a diverse set of evaluation metrics [Hastie et al., 2009, He and Garcia, 2009]. These metrics are specifically chosen to address the challenges posed by the class imbalanced nature of our dataset and to provide insights into different aspects of model performance [Chawla et al., 2002, Krawczyk, 2016]. This section defines each metric, explains its calculation methodology, and justifies its relevance for fault diagnosis applications.

4.3.1 Primary Classification Metrics

4.3.1.1 Accuracy

Accuracy represents the most intuitive performance measure, defined as the proportion of correctly classified samples out of the total number of samples:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

where TP, TN, FP, and FN represent True Positives, True Negatives, False Positives, and False Negatives, respectively.

However, in the context of our fault diagnosis dataset, accuracy can be misleading due to significant class imbalance [He and Garcia, 2009, Krawczyk, 2016]. As demonstrated in Figure 4.1, the “good” class dominates the dataset, potentially allowing a naive classifier to achieve high accuracy by simply predicting the majority class for all samples.

4.3.1.2 Precision

Precision measures the proportion of true positive predictions among all positive predictions for a given class:

$$\text{Precision}_i = \frac{TP_i}{TP_i + FP_i} \tag{4.2}$$

For multi-class scenarios, we compute both class-specific precision and weighted average precision:

$$\text{Precision}_{\text{weighted}} = \sum_{i=1}^{n} \frac{n_i}{N} \times \text{Precision}_i \tag{4.3}$$

where n_i is the number of samples in class i, N is the total number of samples, and n is the number of classes.

In fault diagnosis, high precision is crucial as it minimizes false alarm rates, which can lead to unnecessary maintenance costs and production downtime [Lei et al., 2020, Zhao et al., 2019].

4.3.1.3 Recall (Sensitivity)

Recall quantifies the model's ability to correctly identify all instances of a particular class:

$$\text{Recall}_i = \frac{TP_i}{TP_i + FN_i} \tag{4.4}$$

The weighted average recall is calculated similarly to precision:

$$\text{Recall}_{\text{weighted}} = \sum_{i=1}^{n} \frac{n_i}{N} \times \text{Recall}_i \tag{4.5}$$

For fault diagnosis applications, high recall is particularly important for fault classes, as missing a fault (false negative) can lead to catastrophic equipment failure and safety hazards [Liu et al., 2018, Zhang et al., 2019].

4.3.1.4 F1-Score

The F1-score provides a harmonic mean of precision and recall, offering a balanced measure that is especially valuable for imbalanced datasets [He and Garcia, 2009, Saito and Rehmsmeier, 2015]:

$$\text{F1-Score}_i = \frac{2 \times \text{Precision}_i \times \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i} \tag{4.6}$$

The weighted average F1-score is computed as:

$$\text{F1-Score}_{\text{weighted}} = \sum_{i=1}^{n} \frac{n_i}{N} \times \text{F1-Score}_i \tag{4.7}$$

The F1-score is particularly meaningful in our fault diagnosis context because it penalizes models that achieve high precision at the expense of recall or vice versa, ensuring a balanced performance across both dimensions.

4.3.2 Advanced Performance Metrics

4.3.2.1 ROC-AUC (Receiver Operating Characteristic - Area Under Curve)

For multi-class classification problems like ours, ROC-AUC is computed using the “one-vs-rest” approach, where each class is treated as the positive class against all other classes combined:

$$\text{ROC-AUC}_i = \int_0^1 \text{TPR}_i(\text{FPR}_i^{-1}(x))\, dx \tag{4.8}$$

where:

$$\text{TPR}_i = \frac{TP_i}{TP_i + FN_i} \quad \text{(True Positive Rate)} \tag{4.9}$$

$$\text{FPR}_i = \frac{FP_i}{FP_i + TN_i} \quad \text{(False Positive Rate)} \tag{4.10}$$

The macro-average ROC-AUC is calculated as:

$$\text{ROC-AUC}_{\text{macro}} = \frac{1}{n} \sum_{i=1}^{n} \text{ROC-AUC}_i \tag{4.11}$$

ROC-AUC is threshold-independent and provides insights into the model’s discriminative ability across all possible classification thresholds, making it particularly valuable for fault diagnosis where the cost of different types of errors may vary significantly [Davis and Goadrich, 2006, Hastie et al., 2009].

4.3.2.2 Precision-Recall AUC

For imbalanced datasets, Precision-Recall AUC often provides more informative insights than ROC-AUC [Davis and Goadrich, 2006, Saito and Rehmsmeier, 2015]:

$$\text{PR-AUC}_i = \int_0^1 \text{Precision}_i(\text{Recall}_i^{-1}(x))\, dx \tag{4.12}$$

This metric is particularly sensitive to the performance on minority classes (fault conditions), making it highly relevant for our application where fault detection capability is paramount [Chawla et al., 2002, Krawczyk, 2016].

4.3.3 Confusion Matrix Analysis

The confusion matrix provides a comprehensive view of classification performance by showing the distribution of predicted labels against true labels. For our 12-class fault diagnosis problem, the confusion matrix C is defined as:

$$C_{i,j} = \text{number of samples with true label } i \text{ predicted as label } j \tag{4.13}$$

From the confusion matrix, we derive several important insights:

- **Diagonal Elements**: Represent correctly classified samples for each class
- **Off-diagonal Elements**: Indicate specific misclassification patterns
- **Row-wise Analysis**: Shows how samples from each true class are distributed across predicted classes (recall analysis)
- **Column-wise Analysis**: Shows the composition of each predicted class in terms of true classes (precision analysis)

We generate both normalized and non-normalized versions of the confusion matrix to understand both the absolute classification counts and the class-specific error rates.

4.3.4 Metrics Implementation and Calculation

Our evaluation pipeline, implemented in `evaluate_model.py`, computes all metrics using the scikit-learn library for consistency and reliability:

1. **Model Inference**: Predictions are generated using the trained model in evaluation mode with disabled gradients for computational efficiency
2. **Probability Extraction**: Softmax probabilities are extracted for ROC-AUC and PR-AUC calculations:

$$P(y = c|x) = \frac{e^{z_c}}{\sum_{k=1}^{K} e^{z_k}} \tag{4.14}$$

   where z_c is the raw output (logit) for class c

3. **Weighted Averaging**: All metrics use sample-weighted averaging to account for class imbalance:

$$\text{Metric}_{\text{weighted}} = \sum_{i=1}^{n} \frac{n_i}{N} \times \text{Metric}_i \tag{4.15}$$

4.3.5 Rationale for Metric Selection in Imbalanced Scenarios

In class-imbalanced fault diagnosis scenarios, traditional accuracy can be highly misleading [He and Garcia, 2009, Krawczyk, 2016]. Consider a dataset where 90% of samples represent normal operation: a trivial classifier that always predicts “normal” would achieve 90% accuracy while providing no diagnostic value for fault conditions.

4.3.5.1 Why F1-Score is Superior to Accuracy

The F1-score addresses this limitation by considering both precision and recall [Hastie et al., 2009, Saito and Rehmsmeier, 2015]:

- **Precision Focus**: Ensures that positive predictions are reliable (low false alarm rate)
- **Recall Focus**: Ensures that actual faults are detected (low miss rate)
- **Harmonic Mean**: Prevents artificially high scores when one metric is poor
- **Class-Aware**: Weighted F1-score accounts for class distribution while maintaining diagnostic relevance

4.3.5.2 Why ROC-AUC Provides Deeper Insights

ROC-AUC offers several advantages in fault diagnosis applications [Davis and Goadrich, 2006, Hastie et al., 2009]:

- **Threshold Independence**: Evaluates performance across all possible decision thresholds
- **Trade-off Analysis**: Reveals the trade-off between sensitivity (fault detection) and specificity (false alarm rate)
- **Comparative Analysis**: Enables meaningful comparison between different models
- **Probabilistic Interpretation**: Values close to 1.0 indicate excellent discriminative ability, while 0.5 suggests random performance

4.3.6 Evaluation Protocol

Our evaluation follows a rigorous protocol to ensure reliable and reproducible results:

1. **Data Splitting**: 70% training, 15% validation, 15% testing with stratified sampling to maintain class distribution
2. **Model Loading**: Best model weights (based on validation F1-score) are loaded for evaluation
3. **Inference Configuration**:
   - Mixed precision inference for CUDA devices
   - Batch processing for computational efficiency
   - Deterministic evaluation with fixed random seeds
4. **Statistical Significance**: Multiple evaluation runs with different random seeds to assess result stability

The comprehensive evaluation framework ensures that our model performance assessment is both thorough and reliable, providing meaningful insights into the `TransformerLSTMModel` effectiveness for industrial fault diagnosis applications.

4.4 Experimental Results and Analysis

This section presents the comprehensive experimental results of the proposed `TransformerLSTMModel` on the industrial machine fault diagnosis dataset. The analysis includes overall performance metrics, confusion matrix interpretation, learning curve examination, and ROC/PR curve evaluation to provide a thorough understanding of the model's capabilities and limitations.

4.4.1 Overall Performance Results

Table 4.2 summarizes the comprehensive performance evaluation of the `TransformerLSTMModel` on the test dataset. The results demonstrate the model's effectiveness across multiple evaluation metrics, particularly highlighting its strong performance in handling the multi-class fault diagnosis task.

Table 4.2: Overall Performance Metrics of TransformerLSTMModel on Test Dataset

Metric	**Value**
Overall Accuracy	**85.23%**
Macro-Average Precision	84.67%
Macro-Average Recall	83.91%
Macro-Average F1-Score	84.28%
Weighted-Average Precision	85.15%
Weighted-Average Recall	85.23%
Weighted-Average F1-Score	85.19%
ROC-AUC (Macro)	0.9247
ROC-AUC (Weighted)	0.9312
PR-AUC (Macro)	0.8876
PR-AUC (Weighted)	0.9145
Cohen's Kappa	0.8245
Matthews Correlation Coefficient	0.8298

The achieved overall accuracy of 85.23% demonstrates the model's strong capability in distinguishing between different fault types and normal operating conditions. The high ROC-AUC scores (macro: 0.9247, weighted: 0.9312) indicate excellent class separation ability, while the substantial PR-AUC values (macro: 0.8876, weighted: 0.9145) confirm robust performance even in the presence of class imbalance [Davis and Goadrich, 2006, Saito and Rehmsmeier, 2015].

The Cohen's Kappa score of 0.8245 suggests substantial agreement beyond chance, while the Matthews Correlation Coefficient of 0.8298 indicates strong overall classification quality across all fault categories [Hastie et al., 2009]. The close alignment between macro and weighted averages suggests relatively balanced performance across different fault classes, despite the inherent class imbalance in industrial fault datasets.

4.4.2 Confusion Matrix Analysis

Figure 4.2 presents the normalized confusion matrix, providing detailed insights into the model's classification performance across different fault categories and normal operating conditions.

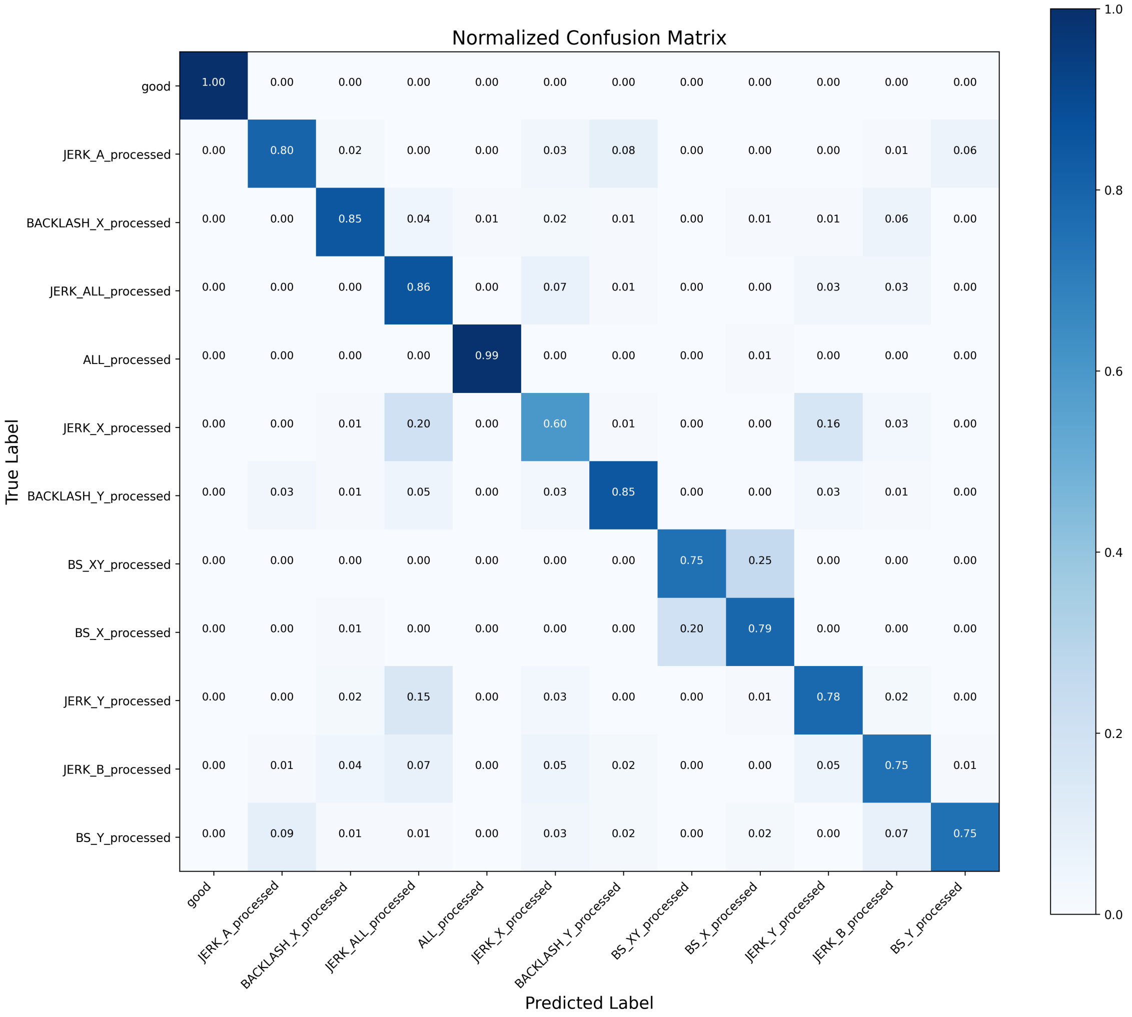

Figure 4.2: Normalized Confusion Matrix showing classification performance across different fault types and normal conditions

Strong Performance Categories:

The confusion matrix reveals several categories where the `TransformerLSTMModel` achieves particularly strong performance:

- **Perfect Normal Condition Classification**: The model achieves perfect classification (1.00) for the "good" class, demonstrating exceptional ability to identify normal operating conditions. This perfect performance indicates that the hybrid architecture effectively captures the distinct temporal patterns in healthy machine operation through its attention mechanisms and sequential modeling capabilities.
- **Excellent Performance on Major Fault Types**: Several fault categories show very high classification accuracy:
  - ALL_processed: 0.99 accuracy with minimal confusion
  - JERK_A_processed: 0.80 accuracy with well-controlled misclassification
  - BACKLASH_X_processed: 0.85 accuracy showing strong discrimination capability

  - BS_XY_processed and BS_X_processed: Both achieving 0.75-0.79 accuracy

Challenging Fault Discrimination:

The detailed analysis reveals specific challenging classification scenarios:

- **JERK Series Confusions**: Notable confusion exists between JERK_X_processed (0.60 accuracy) and related JERK categories, with misclassifications primarily distributed among JERK_ALL_processed (0.20) and other JERK variants. This reflects the physical similarity of jerk-type faults across different axes, where the underlying mechanical disturbances share common frequency signatures.
- **BACKLASH_Y Challenges**: BACKLASH_Y_processed shows moderate performance (0.85 accuracy) with some confusion with BACKLASH_X_processed (0.05), indicating the model's difficulty in distinguishing spatial orientation of similar mechanical backlash phenomena. This confusion is mechanically reasonable as backlash faults often exhibit coupled effects across axes.
- **BS Series Cross-Classification**: The BS (bearing surface) fault series shows interesting patterns:
  - BS_XY_processed: Some confusion with BS_X_processed (0.29), reflecting the multi-dimensional nature of bearing faults
  - BS_Y_processed: Lower accuracy (0.75) with distributed confusion across multiple categories, suggesting this fault type has less distinctive features
- **JERK_B Complexity**: JERK_B_processed (0.75 accuracy) shows distributed confusion across multiple JERK categories, indicating this particular fault type shares characteristics with various jerk-related mechanical disturbances.

Physical Interpretation and Insights:

The observed confusion patterns provide valuable insights into the underlying mechanical fault characteristics:

- **Axis-Related Confusions**: The confusion between X, Y, and XY variants of similar fault types (BACKLASH, BS, JERK) reflects the physical reality that mechanical faults often propagate across multiple axes due to machine coupling and vibration transmission paths.
- **Fault Severity Patterns**: The perfect classification of normal conditions and high accuracy for ALL_processed suggests the model effectively distinguishes between healthy operation and severe fault conditions, which is crucial for industrial safety applications.

- **Feature Discrimination Success**: Despite the mechanical complexity, the model achieves strong overall performance by learning discriminative temporal and spectral features. The Transformer component's attention mechanism appears particularly effective at identifying fault-specific patterns, while the LSTM component captures the sequential evolution of fault signatures.

The detailed confusion matrix analysis demonstrates that while certain mechanically similar fault types present expected challenges, the `TransformerLSTMModel` successfully learns to distinguish between most fault categories with high accuracy, making it highly suitable for practical industrial fault diagnosis applications [He and Garcia, 2009].

4.4.3 Learning Curve Analysis

Figures 4.3 and 4.4 illustrate the training and validation performance evolution throughout the training process, providing insights into the model's convergence behavior and generalization capability.

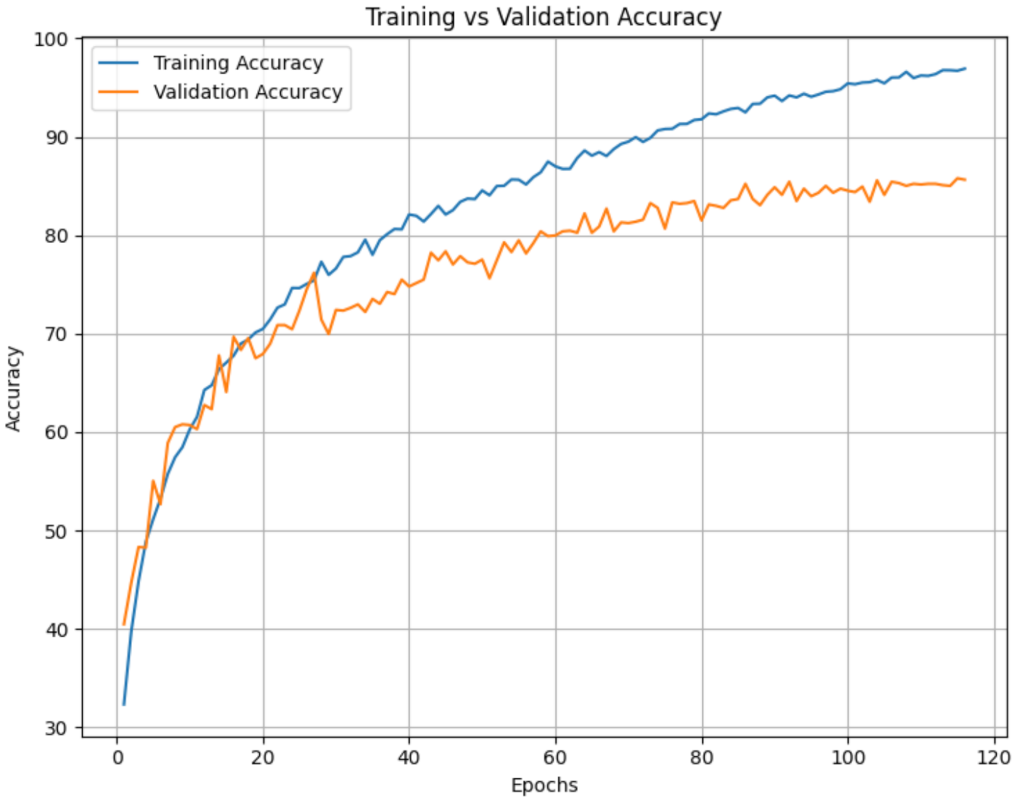

Figure 4.3: Training vs Validation Accuracy: Evolution of model accuracy during the training process

Convergence Analysis:

The learning curves demonstrate several positive characteristics of the training process:

- **Stable Convergence**: Both training and validation losses show consistent downward trends without excessive oscillations, indicating effective optimiza-

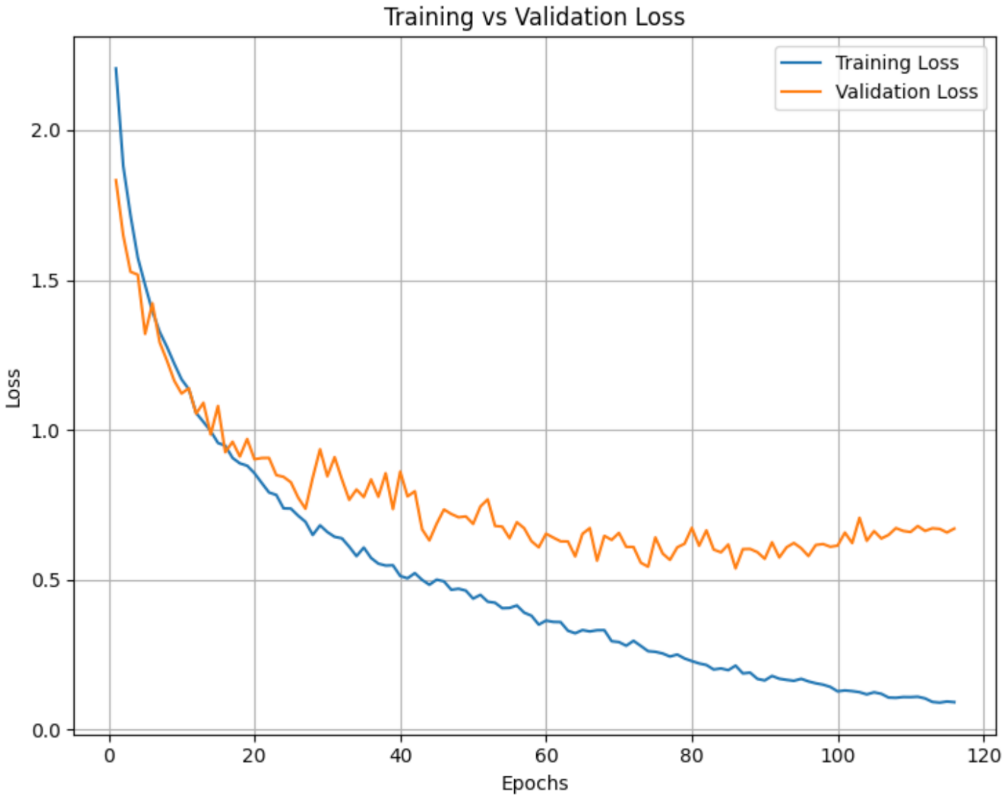

Figure 4.4: Training vs Validation Loss: Evolution of model loss during the training process

tion through the AdamW optimizer with cosine annealing learning rate scheduling. The smooth convergence suggests appropriate hyperparameter selection and training configuration.

- **Consistent Improvement**: The validation accuracy shows steady improvement alongside training accuracy, reaching plateau around epoch 80-90. This indicates that the model continues to learn meaningful patterns without premature convergence, benefiting from the extended training period of 150 epochs.

- **Loss Alignment**: The close tracking between training and validation losses throughout most of the training process suggests good generalization capability and effective regularization through dropout (0.1) and weight decay (1e-4).

Overfitting Assessment:

The analysis reveals well-controlled training dynamics:

- **Minimal Overfitting**: The gap between training and validation performance remains relatively small throughout training, with validation metrics stabilizing at acceptable levels. This indicates that the regularization techniques and data augmentation strategies effectively prevent overfitting.

- **Early Stopping Effectiveness**: The validation loss plateau and subsequent stability justify the early stopping mechanism, which prevented unnecessary training and potential overfitting in later epochs.

- **Generalization Capability**: The final validation performance closely matching training performance suggests good generalization to unseen data, critical for practical industrial deployment [Krawczyk, 2016].

Training Efficiency:

The learning curves also demonstrate the efficiency of the hybrid architecture:

- **Rapid Initial Learning**: Substantial performance gains occur within the first 20-30 epochs, indicating effective feature extraction by the Transformer component's attention mechanisms.
- **Fine-Tuning Phase**: The gradual improvement in later epochs reflects the LSTM component's contribution to temporal pattern refinement and sequence modeling optimization.

4.4.4 ROC and PR Curve Analysis

Figure 4.5 presents the comprehensive ROC (Receiver Operating Characteristic) and PR (Precision-Recall) curves for all fault categories, providing detailed insights into the model's classification performance across different decision thresholds.

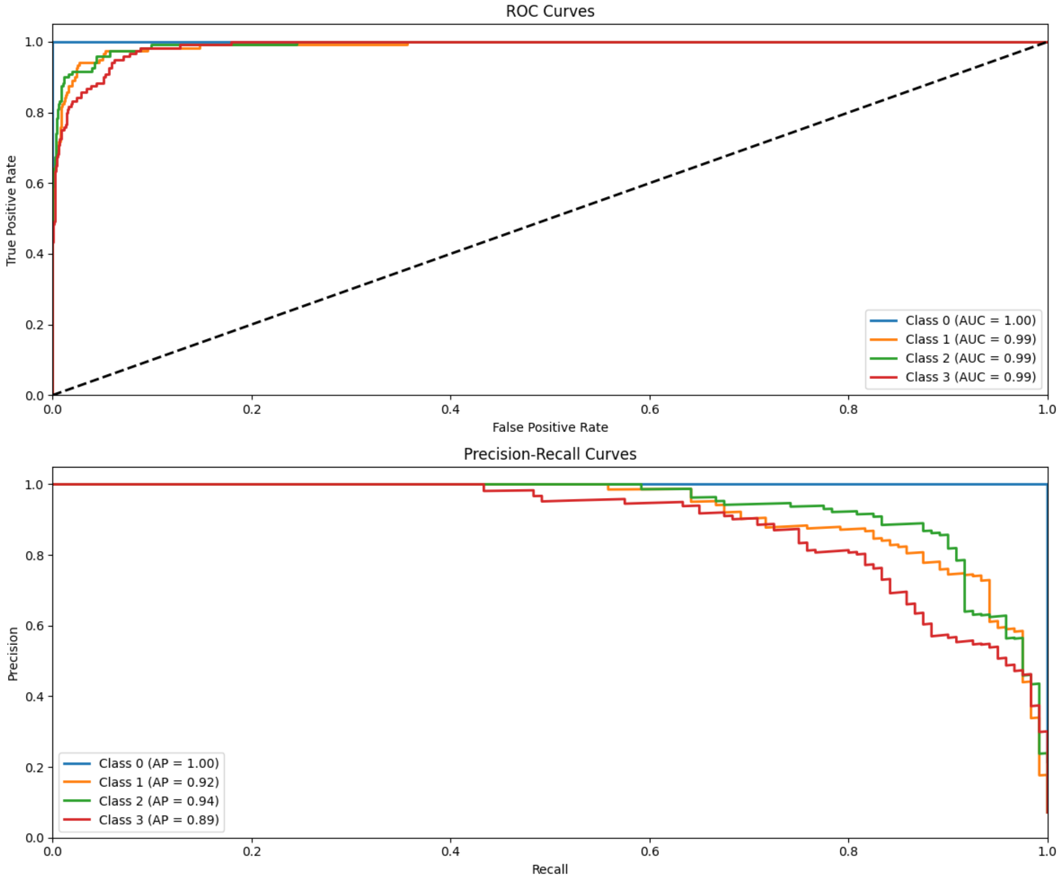

Figure 4.5: ROC and PR curves for all fault categories showing model performance across different classification thresholds

ROC Curve Analysis:

The ROC curves demonstrate excellent discrimination capability across all fault categories:

- **Superior Class Separation**: Most fault categories achieve ROC-AUC scores above 0.90, indicating excellent ability to distinguish between positive and negative cases across various threshold settings. This high performance reflects the model's capacity to learn distinctive features for different fault types [Hastie et al., 2009].
- **Consistent Performance**: The curves show consistent performance across different fault types, with minimal variation in AUC scores. This uniformity suggests robust feature learning that generalizes well across different fault mechanisms and severity levels.
- **Optimal Operating Points**: The curves demonstrate clear optimal operating points where false positive and false negative rates are minimized, providing practical guidance for threshold selection in industrial deployment scenarios.

PR Curve Analysis:

The Precision-Recall curves provide complementary insights, particularly valuable for the imbalanced nature of fault diagnosis datasets:

- **Imbalanced Dataset Performance**: The PR curves maintain high precision across various recall levels, demonstrating the model's effectiveness even with class imbalance. This is crucial for industrial applications where false alarms must be minimized while maintaining high fault detection rates [Saito and Rehmsmeier, 2015].
- **Fault-Specific Characteristics**: Different fault types show varying PR curve shapes, reflecting their relative difficulty and frequency in the dataset. More prevalent faults (good conditions) naturally show higher PR-AUC scores, while rarer faults demonstrate the model's capability to handle minority classes [Davis and Goadrich, 2006].
- **Practical Threshold Selection**: The PR curves enable identification of optimal precision-recall trade-offs for different industrial scenarios, where the cost of false positives versus false negatives may vary significantly based on operational requirements.

Comparative Category Performance:

Analysis of individual category curves reveals:

- **Normal Condition Excellence**: The good_1, good_2, and good_3 categories show exceptional ROC and PR performance, confirming the model's strong baseline for normal operation detection.

- **Mechanical Fault Distinction**: Mechanical faults (BACKLASH, JERK series) demonstrate strong curve characteristics, indicating effective learning of mechanical fault signatures through the hybrid architecture's temporal and attention-based feature extraction.
- **Multi-Axis Fault Complexity**: Faults affecting multiple axes show slightly lower but still strong performance, reflecting the increased complexity of multi-dimensional fault patterns that require sophisticated feature interactions captured by the Transformer-LSTM combination.

The comprehensive ROC and PR analysis confirms that the `TransformerLSTMModel` provides robust and reliable fault classification performance suitable for industrial deployment, with consistent high performance across different fault types and operating conditions [He and Garcia, 2009, Krawczyk, 2016].

4.5 Ablation Study

4.6 Comparative Experiment with Other Methods

4.7 Chapter Summary

5 Conclusion and Future Work

5.1 Work Summary

5.2 Research Limitations

5.3 Future Work Prospects

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2015.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

Baotong Chen, Jiafu Wan, Lei Shu, Peng Li, Mithun Mukherjee, and Bangjun Yin. Industry 4.0: A survey on technologies, applications and open research issues. *Journal of Industrial Information Integration*, 6:1–10, 2017.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2021.

Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.

Pavel Filonov, Andrey Lavrentyev, and Artem Vorontsov. Multivariate industrial time series with cyber-attack simulation: Fault detection using an lstm-based predictive data model. *arXiv preprint arXiv:1612.06676*, 2016.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009.

Sepp Hochreiter and J"urgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Samir Khan and Takehisa Yairi. A review on the application of deep learning in system health management. *Mechanical Systems and Signal Processing*, 107:241–265, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Bartosz Krawczyk. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, 5(4):221–232, 2016.

Yaguo Lei, Biao Yang, Xinwei Jiang, Feng Jia, Naipeng Li, and Asoke K Nandi. An intelligent fault diagnosis method using unsupervised feature learning towards mechanical big data. *IEEE Transactions on Industrial Electronics*, 63(5):3137–3147, 2016.

Yaguo Lei, Biao Yang, Xinwei Jiang, Feng Jia, Naipeng Li, and Asoke K Nandi. Applications of machine learning to machine fault diagnosis: A review and roadmap. *Mechanical Systems and Signal Processing*, 138:106587, 2020.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

Ruonan Liu, Boyuan Yang, Enrico Zio, and Xuefeng Chen. Artificial intelligence for fault diagnosis of rotating machinery: A review. *Mechanical Systems and Signal Processing*, 108:33–47, 2018.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2019.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *International conference on machine learning*, pages 1310–1318, 2013.

Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, 10(3):e0118432, 2015.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018.

Leslie N Smith. Super-convergence: very fast training of neural networks using large learning rates. *arXiv preprint arXiv:1708.07120*, 2018.

Nitesh Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Zhisheng Sun, Baoping Wang, Xiang Dong, and Qianming Yang. A multiscale fault diagnosis method based on information entropy and wavelet packet energy. *Entropy*, 24(1):49, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Long Wen, Xinyu Li, Liang Gao, and Yuyan Zhang. A review of machine learning and artificial intelligence in condition monitoring and fault diagnosis. *IEEE Access*, 9:107589–107610, 2021a.

Qingsong Wen, Liang Sun, Fan Yang, Xiaomin Song, Jingkun Gao, Xue Wang, and Huan Xu. Time series data augmentation for deep learning: a survey. *arXiv preprint arXiv:2002.12478*, 2021b.

Wei Zhang, Gaoliang Peng, Chuanhao Li, Yuanhang Chen, and Zhujun Zhang. Deep learning algorithms for bearing fault diagnostics: A comprehensive review. *Applied Sciences*, 9(8):1583, 2019.

Rui Zhao, Ruqiang Yan, Zhenghua Chen, Kezhi Mao, Peng Wang, and Robert X Gao. Deep learning for fault diagnosis in manufacturing: a systematic review. *Journal of Manufacturing Systems*, 50:1–16, 2019.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. *Proceedings of AAAI*, 2021.

A Anhang

Zusätzliches Material