processes

MDPI

Article

Industrial Part Faults Prediction for Nonlinearity and Implied Temporal Sequences

Shuyu Zhang [1,†], Mengyi Zhang [1,*,†], Cuimei Bo [1,†] and Cunsong Wang [2,*]

[1] College of Electrical Engineering and Control Science, Nanjing Tech University, Nanjing 211800, China; 18351969932@163.com (S.Z.); bcm@njtech.edu.cn (C.B.)
[2] Institute of Intelligent Manufacturing, Nanjing Tech University, Nanjing 210009, China
* Correspondence: myzhang@njtech.edu.cn (M.Z.); wangcunsong@njtech.edu.cn (C.W.)
† These authors contributed equally to this work.

Abstract: The ability to preemptively identify potential failures in industrial parts is crucial for minimizing downtime, reducing maintenance costs and ensuring system reliability and safety. However, challenges such as data nonlinearity, temporal dependencies, and imbalanced datasets complicate accurate fault prediction. In this study, we propose a novel combined approach that integrates the Logistic Model Tree Forest (LMT) with Stacked Long Short-Term Memory (LSTM) networks, addressing these challenges effectively. This hybrid method leverages the decision-making capability of the LMT and the temporal sequence learning ability of Stacked LSTM to improve fault prediction accuracy. Additionally, to tackle the issues posed by imbalanced datasets and noise, we employ the ENN-SMOTE (Edited Nearest Neighbors-Synthetic Minority Over-sampling Technique), a technique for data preprocessing, which enhances data balance and quality. Experimental results show that our approach significantly outperforms traditional methods, achieving a fault prediction accuracy of up to 98.2%. This improvement not only demonstrates the effectiveness of the combined model but also highlights its potential for real-world industrial applications, where high accuracy and reliability are paramount.

Keywords: fault prediction; logistic model tree; clarification; LSTM

check for updates

Academic Editors: Iqbal M. Mujtaba and Olympia Roeva

Received: 8 December 2024
Revised: 18 January 2025
Accepted: 29 January 2025
Published: 6 February 2025

Citation: Zhang, S.; Zhang, M.; Bo, C.; Wang, C. Industrial Part Faults Prediction for Nonlinearity and Implied Temporal Sequences. *Processes* **2025**, *13*, 436. https://doi.org/10.3390/pr13020436

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Predicting industrial part defects is of great significance in modern manufacturing, as it enables the early identification of potential issues, preventing defective products from advancing to subsequent production stages or reaching end users. Such predictions not only minimize rework and scrap rates but also reduce production costs and enhance quality control [1,2]. Intelligent defect prediction systems further facilitate real-time monitoring and adaptive decision-making, which are critical for ensuring consistent product quality and improving the flexibility and responsiveness of production lines [3]. Despite the rapid advancements in defect prediction technologies, achieving high prediction accuracy under complex data scenarios, such as imbalanced datasets and nonlinear dependencies, remains a challenging problem.

Machine learning (ML) has been extensively used in predictive analytics, leveraging historical data to identify patterns and predict future outcomes [4]. It enables the development of predictive models using a variety of algorithms and empowers systems to extract knowledge from diverse sources for intelligent predictive analysis [5–7]. The Logistic Model Tree (LMT), a decision tree-based algorithm, has been particularly valued for its straightforward interpretability and efficient pruning mechanism [8]. These capabilities enable it to

achieve optimal performance, as evidenced by numerous studies [9,10]. For instance, studies have shown its effectiveness in domains ranging from spectroradiometers and drone technologies to miRNA-disease association prediction, achieving high sensitivity and AUC metrics [11,12]. However, the LMT's inability to model temporal dependencies and handle complex nonlinear relationships limits its applicability in dynamic industrial environments.

Recent advancements in deep learning, particularly in the field of neural networks, have significantly enhanced the ability to model complex temporal dependencies and nonlinear feature interactions in industrial defect prediction. Long Short-Term Memory (LSTM) networks, on the other hand, excel in sequential data modeling and have been successfully applied in various domains, including time series forecasting and industrial fault prediction [13–15]. For example, Tang et al. [14] combined wavelet transforms and singular spectrum analysis with LSTM for financial time series forecasting, while Wu et al. [15] proposed a bidirectional LSTM with ensemble attention for variable selection. Li et al. [16] derive dynamic time lags to reconstruct multivariate datasets, enhancing attention-based LSTM models for industrial time series prediction. However, LSTM models are often constrained by their inability to efficiently capture complex feature interactions, which are prevalent in industrial datasets with strong nonlinear dependencies.

In recent years, newer approaches leveraging Transformer Networks, Graph Neural Networks (GNNs), and hybrid models that integrate CNNs and LSTMs have emerged as powerful tools for defect prediction. Transformer-based models, with their attention mechanisms, can better capture long-range dependencies and contextual information in time-series data [17]. GNNs, on the other hand, have shown promise in learning from graph-structured data, enabling better handling of spatial and temporal dependencies [18]. Additionally, hybrid models, such as the CNN-LSTM [19] and CNN-Transformer [20], combine the strengths of multiple architectures, addressing both temporal dependencies and nonlinear feature interactions more effectively. These models have demonstrated superior performance in various industrial applications, including predictive maintenance, fault diagnosis, and quality control, owing to their ability to learn complex patterns from high-dimensional and imbalanced datasets.

Steel plate fault prediction, a crucial aspect of materials science and industrial manufacturing, exemplifies the challenges of defect prediction in high-dimensional and nonlinear environments [21]. Traditional machine learning algorithms, such as Support Vector Machines (SVMs) [22,23], decision trees (DTs) [22,24–26], and Random Forests (RFs) [27–29], have been applied extensively but fail to address temporal dependencies effectively. Conversely, deep learning methods, including LSTM [30] and Convolutional Neural Networks (CNNs) [31], often overlook complex nonlinear relationships or require significant computational resources for training. Therefore, a hybrid approach that combines the strengths of multiple algorithms is needed to address these limitations.

Despite the significant progress in defect prediction technologies, current methods often face limitations in addressing both temporal dependencies and nonlinear feature interactions simultaneously. While the LMT provides interpretability and efficient decision-making, its inability to model temporal relationships constrains its effectiveness in sequential data. Similarly, LSTM excels in temporal modeling but struggles with capturing nonlinear feature interactions, which are critical in complex industrial datasets. Furthermore, few studies have explored the integration of these two methodologies to overcome their individual shortcomings. For visual display, a table summary is shown in Table 1.

Table 1. Summary of literature review.

Citation	Methodology	Key Findings	Limitations and Relation to This Study
[1,2]	Intelligent Defect Prediction Systems	Enabled early detection of defects, reducing rework, scrap rates, and costs while improving quality control and flexibility.	Did not address challenges of temporal dependencies or nonlinear relationships.
[4]	Machine Learning (ML)	Highlighted the role of ML in leveraging historical data to identify patterns and predict future outcomes.	Provided a general overview without addressing specific challenges like data imbalance or nonlinear dependencies.
[8,11,12]	Logistic Model Tree (LMT)	Demonstrated LMT's interpretability and efficient pruning mechanism; successfully applied in fields like spectroradiometers, drone technology, and miRNA-disease prediction.	LMT cannot model temporal dependencies or handle nonlinear relationships, which limits its application in dynamic environments.
[13–15]	Long Short-Term Memory (LSTM)	Showed LSTM's capability in time series forecasting, including applications with wavelet transforms and ensemble attention mechanisms.	LSTM struggles with modeling complex feature interactions and nonlinear relationships in industrial datasets.
[22–24,26,27,30]	SVM, DT, RF, CNN, and LSTM	Applied various ML and DL methods for defect prediction, demonstrating their strengths in classification and temporal modeling.	Traditional ML methods fail to address temporal dependencies; DL methods often overlook complex nonlinear relationships and require significant computational resources.
[21]	Logistic Regression	Highlighted the effectiveness of logistic regression in certain industrial applications.	Provided limited insights on handling high-dimensional and nonlinear dependencies in industrial data.
[17]	Transformer Networks	Demonstrated the effectiveness of Transformer models in capturing long-range dependencies and contextual information in time-series data.	May require large datasets and high computational resources for training.
[18]	Graph Neural Networks (GNNs)	Highlighted the potential of GNNs in learning from graph-structured data, which can capture both spatial and temporal dependencies.	GNNs may have limited scalability in extremely large industrial datasets.
[19]	Hybrid CNN-LSTM Models	Combined CNN and LSTM models to address both spatial and temporal dependencies in predictive maintenance.	Computationally intensive and may require extensive hyperparameter tuning.
[20]	CNN-Transformer Models	Integrated CNN and Transformer models to better capture complex nonlinear relationships in industrial fault diagnosis.	Requires substantial computational resources and careful tuning of model parameters.

To address these gaps, this study proposes a novel hybrid defect prediction framework that combines the Logistic Model Tree (LMT) with Stacked Long Short-Term Memory (LSTM). This approach leverages the LMT's strength in nonlinear decision-making and LSTM's temporal modeling capabilities, offering an innovative solution for steel plate fault prediction. By systematically integrating these two models, the proposed framework aims to enhance prediction accuracy, robustness, and interpretability, thus advancing the state of defect prediction in industrial manufacturing.

Specifically, we have emphasized the following unique aspects of our research:

1. The integration of the Logistic Model Tree (LMT) with Long Short-Term Memory (LSTM) networks to leverage both the interpretability of decision tree-based models and the ability of LSTM to capture temporal dependencies, which has not been explored in previous studies.
2. The use of ENN-SMOTE for data preprocessing to address the challenges of imbalanced and noisy datasets, a critical issue in real-world applications.
3. The application of this novel framework to the specific problem of steel plate defect classification, demonstrating its effectiveness in improving classification accuracy (98.2%) compared to existing methods.

The rest of this paper is structured as follows: Section 2 provides a detailed description of the preliminary work and algorithm principles. Section 3 presents the case study results, while Section 4 discusses the findings and comparisons. Finally, Section 5 summarizes the study and suggests directions for future research.

2. Preliminary

This part serves to provide the theoretical foundation and key methodologies that underpin the proposed hybrid LMT-LSTM model. By summarizing the principles of Logistic Model Trees (LMTs), Long Short-Term Memory (LSTM) networks, and ENN-SMOTE preprocessing techniques, this section aims to equip readers with the necessary context to understand the innovations presented in Section 3. While the techniques discussed here are well established, their integration and application in the proposed framework constitute a novel contribution, as detailed in subsequent sections.

2.1. ENN

The Edited Nearest Neighbors (ENNs) algorithm is primarily used to remove noisy samples from a dataset. Its fundamental concept involves using the nearest neighbor method to determine whether a sample is consistent with the classification of its neighbors. If a sample is found to be inconsistent with its neighbors, it is considered a noisy sample and subsequently removed.

For each sample x_i in the dataset, the algorithm identifies its k nearest neighbors. Typically, k is set to 3, meaning that the three nearest neighbors of each sample are considered. The definition of nearest neighbors is based on the Euclidean distance, which can be represented as follows:

$$d(x_i, x_j) = \sqrt{\sum_{m=1}^{M} (x_i^m - x_j^m)^2} \tag{1}$$

where x_i^m and x_j^m are the values of samples x_i and x_j in the m-th dimension, and M is the number of dimensions of the samples. For each sample x_i, find the classes of its k nearest neighbors $y_{i1}, y_{i2}, \ldots, y_{ik}$ and determine whether the classes of these nearest neighbors are consistent with the class y_i of sample x_i. If the majority of the neighbors' classes differ from

the class of the sample, the sample is considered a noisy sample. The following formula can be used to decide whether to delete sample x_i:

$$Remove\ x_i\ if\ \frac{1}{k}\sum_{j=1}^{k} I(y_{ij} \neq y_i) > 0.5 \tag{2}$$

where I is the indicator function, which equals 1 when $y_{ij} \neq y_i$ and 0 otherwise.

2.2. SMOTE

The Synthetic Minority Over-sampling Technique (SMOTE) increases the number of minority class samples by generating synthetic samples, thereby balancing the class distribution in the dataset. SMOTE creates new synthetic samples by interpolating between existing samples of the minority class, avoiding the issues associated with simply duplicating minority class samples.

Assume we have an imbalanced dataset where the minority class sample set is $S = \{x_1, x_2, \ldots, x_n\}$ and the majority class sample set is M. For each minority class sample x_i, its k nearest neighbors are identified, which can be computed using the Euclidean distance. For each minority class sample x_i, one sample x_{ij} is randomly selected from its k nearest neighbors. A new synthetic sample x_{new} is then generated using the following formula:

$$x_{new} = x_i + \delta \times (x_{ij} - x_i) \tag{3}$$

where δ is a random number in the range $[0, 1]$. By varying the value of δ, different synthetic samples can be generated between x_i and x_{ij}.

2.3. LMT

The Logistic Model Tree (LMT) combines logistic regression and decision trees to create a powerful predictive model. In this study, THE LMT (Logistic Model Tree) refers to a predictive model that uses a single decision tree. However, when we refer to an ensemble of such trees, we use the term LMT (Logistic Model Tree Forest). Thus, LMT in this paper may refer either to a single tree or a forest of trees, depending on the context. It builds a decision tree with linear logistic regression models at the leaves. The decision tree partitions the feature space into regions, and within each region, a logistic regression model is used to make predictions. This approach leverages the interpretability of decision trees and the predictive power of logistic regression, resulting in a model that can handle both linear and nonlinear relationships in the data. A significant advantage of the LMT is that it determines the number of LogitBoost iterations through validation techniques, thereby integrating logistic regression and classification to prevent overfitting.

The algorithm employs least squares fitting $L_c(x)$:

$$L_c(x) = \sum_{i=1}^{n} \beta_i x_i + \beta_0 \tag{4}$$

where β_i represents the coefficient corresponding to the i-th element in vector x.

The algorithm employs logistic regression methods to determine the posterior probabilities at the nodes of the tree:

$$p(c|x) = \frac{\exp(L_c(x))}{\sum_{c'=1}^{r} \exp(L_{c'}(x))} \tag{5}$$

where r is the number of classes. This theoretical expression can be easily applied to the parameterization of the prediction process in machine learning models.

2.4. LSTM

To address the vanishing gradient problem, LSTM employs gate functions in its state dynamics. Each LSTM unit contains a hidden vector h and a memory vector m. Its structural schematic is shown in Figure 1. At each time step, the memory vector modulates state updates and outputs, allowing the following calculations to be performed.

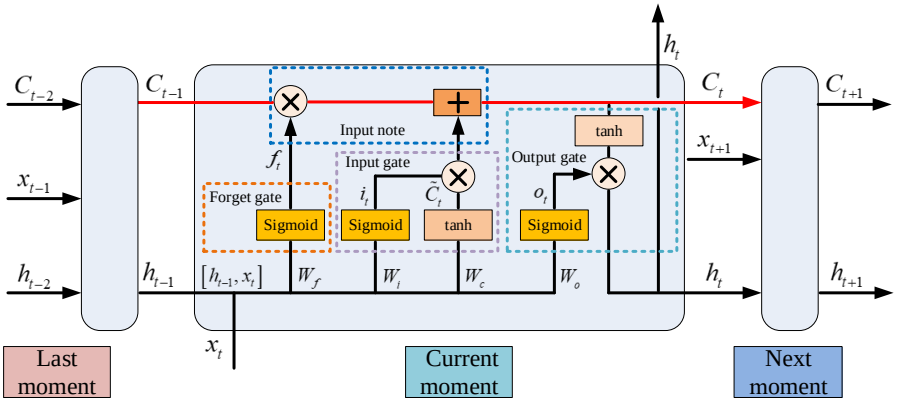


Figure 1. LSTM structure diagram.

The forget gate determines which information in the cell state needs to be forgotten. Its output is a vector between 0 and 1, representing the degree of forgetting for each element in the cell state from the previous time step:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{6}$$

where σ compresses the input to a range between 0 and 1. W_f is the weight matrix for the forget gate, containing weights connecting the previous hidden state h_{t-1} and the current input x_t.$[h_{t-1}, x_t]$ is the concatenated vector of the previous hidden state and the current input.b_f is the bias term for the forget gate.

The input gate determines which new information needs to be added to the cell state. The input gate, together with the candidate state, decides how the new information updates the cell state.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{7}$$

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{8}$$

where W_i is the weight matrix for the input gate. W_c i the weight matrix for the candidate state. tanh compresses the input to a range between -1 and 1, used to generate the candidate state $\tilde{C}_t$, representing the new potential information. b_i, b_c are the bias terms for the input gate and candidate state, respectively.

The cell state is updated through information from the forget gate and the input gate. The new cell state is updated by retaining some of the old information and adding some new information.

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{9}$$

where C_t is the cell state at the current time step. C_{t-1} is the cell state at the previous time step. $f_t \cdot C_{t-1}$ is element-wise multiplication used to forget part of the old cell state. $i_t \cdot \tilde{C}_t$ is element-wise multiplication used to add new information to the cell state.

The output gate determines which information is extracted from the cell state and used as the output at the current time step, while also generating a new hidden state.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{10}$$

$$h_t = o_t \cdot \tanh(C_t) \tag{11}$$

where W_o is the weight matrix for the output gate. $b_o$is the bias term for the output gate. o_t is the activation value of the output gate, indicating which information will be output. $\tanh(C_t)$ compresses the current cell state to a range between -1 and 1 using the tanh function. h_t is the hidden state and output at the current time step.

In summary, this section provides the foundational understanding of the core methodologies employed in this study. These techniques, while individually well established, are uniquely combined in the proposed framework to address the challenges of imbalanced datasets and temporal dependency in industrial fault detection. The details of this integration are presented in Section 3.

3. Materials and Methods

The primary objective of this study is to develop a novel steel plate defect classification and prediction model by integrating advanced machine learning and neural network techniques. Figure 2 illustrates the unique architecture of this model. Unlike traditional approaches, which often apply preprocessing, classification, and feature extraction independently, our method combines these steps into a unified workflow tailored to the challenges posed by imbalanced and noisy industrial datasets. Specifically, the preprocessing phase begins with the application of the ENN technique, which not only removes noisy and borderline samples but also ensures the retention of critical data points, thereby enhancing the overall data quality in a targeted manner. Following this, SMOTE is employed to generate high-quality synthetic samples for the minority class, addressing the class imbalance issue with improved fidelity compared to standard oversampling techniques.

What sets this approach apart is the sequential integration of ENN and SMOTE preprocessing with the LMT classifier, which is adept at capturing complex nonlinear relationships. During the LMT training process, a 10-fold cross-validation strategy is adopted to optimize model performance and generate intermediate features that capture intricate data interactions. These features, which are rarely explored in conventional workflows, are then fused with the original dataset to construct an enriched dataset with enhanced representational capacity. Finally, this enriched dataset is fed into the LSTM network, which is specifically designed to capture and leverage underlying time-series patterns in defect data—a critical requirement for accurate prediction in this domain. This stepwise integration of feature engineering, nonlinear classification, and time-series analysis constitutes a novel methodology that significantly improves classification accuracy while addressing the limitations of existing methods.

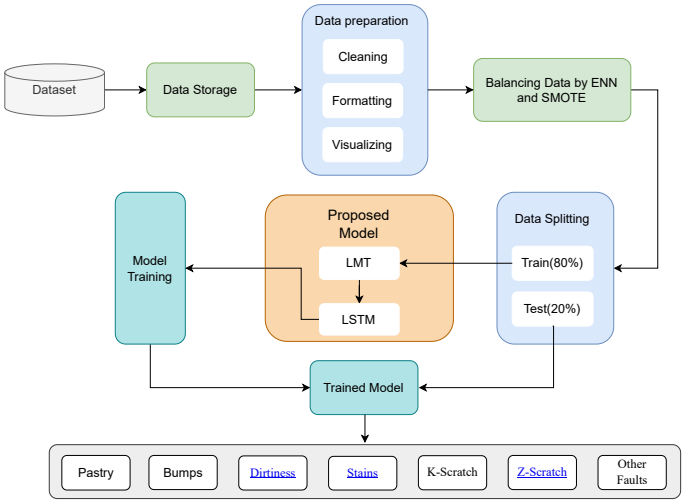


Figure 2. The architecture of the model.

3.1. Data Preprocessing Based on ENN-SMOTE

For the original dataset D, the ENN algorithm is applied to obtain the cleaned dataset $D_{cleaned}$. For the minority class samples in $D_{cleaned}$, the SMOTE algorithm is applied to generate new synthetic samples, resulting in the dataset D_{SMOTE}. The cleaned dataset $D_{cleaned}$ is then combined with the synthetic samples D_{SMOTE} to form the final enhanced dataset $D_{enhanced}$.

$$D_{enhanced} = D_{cleaned} \cup D_{SMOTE} \tag{12}$$

Figure 3 illustrates this technique. Supposing a sample is determined, the Euclidean distance is used to obtain the three closest samples, and the categories of these three samples ($k = 3$) are compared with their own categories. If they are consistent, they are retained, and if they are inconsistent, noisy samples are removed. For instance, sample x_1 belongs to the majority class, with its classification result (blue circle) differing from its original class (green triangle). Therefore, x_1 is removed to reduce the impact of noise on the experimental results. The same applies to samples x_2 and x_3. Since the imbalance in the number of classified samples can lead to inaccurate final classification results, increasing the number of minority samples is crucial. For sample x_4, the Euclidean distance is used to identify the three nearest neighbors, and a new synthetic sample (red square) is interpolated among these three samples.

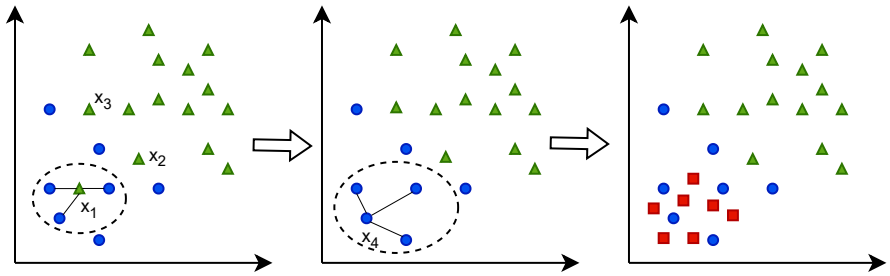


Figure 3. Sample simulation of the ENN-SMOTE method.

3.2. Proposed Model

The results processed by the LMT algorithm are fed into a multi-layer LSTM classification model, utilizing multiple iterations to enhance the model's classification performance. In each iteration, the input sequence first passes through an LSTM layer. This LSTM layer is responsible for learning and remembering long-term dependencies within the sequence, capturing the relationships between time steps, and outputting a hidden state representing the sequence.This hidden state is not directly used for classification but is passed to the next LSTM layer. In the subsequent iteration, the new LSTM layer continues to process this hidden state, further capturing complex patterns and high-level features within the sequence. By stacking multiple LSTM layers, each layer progressively learns different levels of abstract features, thereby enhancing the model's representation capability of the input sequence.

After processing through multiple LSTM layers, the output hidden state of the final layer is fed into a fully connected layer. Figure 4 illustrates the specific implementation of the target model. The role of this fully connected layer is to map the high-dimensional hidden state to a corresponding classification label, outputting the final classification result through an activation function such as the softmax function. This design of the multi-layer LSTM classification model effectively leverages deep feature learning, improving the classification accuracy and generalization ability for input sequences.

In a multi-layer LSTM, each layer's LSTM units pass their hidden state to the next layer.

First LSTM Layer**: For the input x_t at time step t, the hidden state $h_t^{(1)}$ and cell state $C_t^{(1)}$ are computed.

Second LSTM Layer**: The input to the second layer is the hidden state $h_t^{(1)}$ from the first layer, and it calculates the hidden state $h_t^{(2)}$ and cell state $C_t^{(2)}$ for the second layer.

$$h_t^{(2)} = LSTM^{(2)}(h_t^{(1)}) \tag{13}$$

Subsequent LSTM Layers**: For each layer l, the input is the hidden state $h_t^{(l-1)}$ from the previous layer $l-1$. The hidden state $h_t^{(l)}$ and cell state $C_t^{(l)}$ for the current layer are then computed.

$$h_t^{(l)} = LSTM^{(l)}(h_t^{(l-1)}) \tag{14}$$

The output of the last layer's hidden state $h_t^{(L)}$ is fed into a fully connected layer for classification. Assuming the final hidden state is $h_t^{(L)}$, the classification output is

$$y_t = softmax(W_y \cdot h_t^{(L)} + b_y) \tag{15}$$

where W_y and b_y are the weights and bias of the fully connected layer.

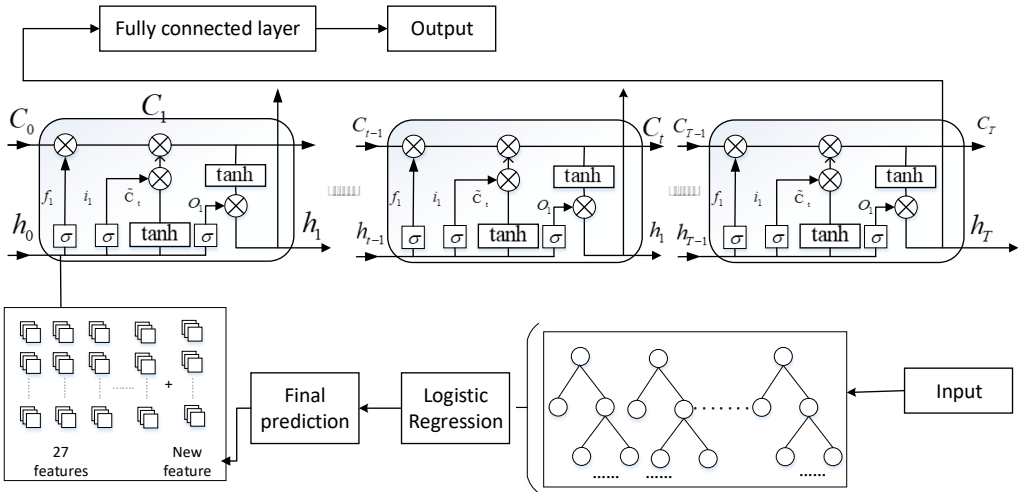


Figure 4. Specific implementation of the target model(LMT-LSTM).

3.3. Model Evaluation

Using the trained model, real-time classification of the existing data is performed. In classification tasks, evaluation metrics are a standard way to assess model performance and are commonly used to evaluate how well a model performs on a specific task. Precision (P), Recall (R), and F1 Score ($F1$) are frequently used as evaluation metrics for algorithms. Precision evaluates the model's performance from the perspective of true positive predictions, while Recall evaluates the model's performance from the perspective of capturing all relevant instances. Since Precision and Recall have a complementary relationship and influence each other, the F1 Score is used to provide a comprehensive measure of both Precision and Recall. The F1 Score ranges from 0 to 1, with values closer to 1 indicating better model performance. The calculation formulas for Precision, Recall, and F1 Score are as follows:

$$P = \frac{TP}{TP + FP} \tag{16}$$

$$R = \frac{TP}{TP + FN} \tag{17}$$

$$F1 = \frac{2PR}{P+R} \tag{18}$$

where TP is the number of true positives, FP is the number of false positives, and FN is the number of false negatives.

During the training process of the proposed LMT-LSTM model, both the loss function and accuracy metrics for the training and validation datasets were monitored. These metrics serve as indicators of the model's convergence and generalization ability, providing valuable insights into the optimization process.

4. Results

4.1. Data Description

In this study, the steel plate fault dataset was utilized to assess the effectiveness of the proposed method. This dataset is provided in the Data Availability Statement. Detailed information about the dataset used in this study is listed in Table 2. This dataset is a multivariate dataset designed to develop and test machine learning models for the automatic identification of fault patterns on steel plate surfaces. Since 2010, this dataset has been widely used in numerous studies to evaluate and compare various fault detection methods. The dataset contains 1941 records, each labeled to indicate different types of faults that may occur on the surface of steel plates. The dataset includes a variety of attribute types, encompassing both integer and real values.

Table 2. Summary of the dataset.

Tasks	Instance	Features	Field
Classification	1941	27	Physics and Chemistry

The dataset comprises 27 distinct features, as detailed in Table 3, including statistical attributes such as the minimum, mean, and maximum values for each feature. Each feature provides critical information about the condition of the steel plate surface, aiding in making accurate judgments during fault classification. These features encompass a wide range of mechanical properties, including strength, toughness, elongation, shape, dimensional accuracy, and appearance. Specifically, strength features may include yield strength and tensile strength, describing the steel plate's performance under stress. Toughness features reflect the material's ability to absorb energy before fracturing. Elongation features describe the extent to which the material stretches during tensile testing, which is crucial for understanding the material's ductility and plastic deformation behavior.

Table 3. Detailed introduction of features.

No	Variable Name	Min	Mean	Max
1	X Maximum	1	617.9645	1713
2	X Minimum	1	571.1360	1749
3	Y Maximum	6724	1650.6848681	12,987.661
4	Y Minimum	6712	1650.6848681	12,987.661
5	Pixels Area	2	4940.0	14,060
6	X Perimeter	1	111.8552	10,449
7	Y Perimeter	1	333.0	10,449
8	Sum of Luminosity	2504	206,312.1479	1,591,144
9	Maximum of Luminosity	234	234.0	234
10	Minimum of Luminosity	1	1.0	1

Table 3. *Cont.*

No	Variable Name	Min	Mean	Max
11	Lenght of Converyer	10,277	16,060	20,859
12	Type of Steel (A300)	0	0.0	0
13	Type of Steel (A400)	0	0.599794	1
14	Steel Plate Thickness	0.90	1.033493	1.10
15	Empty Index	0.414112	0.442010	0.943096
16	Square Index	0.00833	0.5708	3.0
17	Outside X Index	0.000	0.000	0.8759
18	Edges X Index	0.0144	0.1553	0.853
19	Edges Y Index	0.0144	0.1533	0.8539
20	Outside Global Index	0.000	0.014	0.8759
21	Log of Areas	0.3014	2.4924	5.1873
22	Log X Index	0.0	0.0000	0.803
23	Log Y Index	0.0	0.0	0.7039
24	Orientation Index	−0.9914	−0.0383	0.9914
25	Luminosity Index	0.99993	−10.413	10.413
26	Sigmoid of Areas	0.119	0.5854	1
27	Edges Index	0	0.3317	0.9952

Moreover, shape and dimensional accuracy features provide important information about the geometric shape and dimensional consistency of the steel plates. These features may include the thickness, width, and length of the plates, as well as the tolerances for these dimensions during the manufacturing process. Appearance features cover various indicators of surface quality, such as surface roughness, glossiness, and the number of defects.

Statistical attributes such as the minimum, mean, and maximum values are also crucial for understanding each feature. The minimum value provides the lower bound of the feature, indicating the worst-case performance of the steel plate; the maximum value shows the best-case performance; and the mean value offers an overall performance overview.

The rich features and detailed statistical attributes of this dataset provide a solid foundation for developing high-performance fault detection models. Through in-depth analysis of these data, models can effectively capture the complex patterns of faults on steel plate surfaces, enhancing the accuracy and robustness of identification and classification.

4.2. Results of Data Imbalance Processing

The steel production dataset comprises a total of 1941 records, including 158 records of Pastry, 190 records of Z-Scratch, 391 records of K-Scratch, 72 records of Stains, 55 records of Dirtiness, 402 records of Bumps, and 637 records of Other Faults. Upon analyzing the dataset, it is evident that there is a class imbalance problem. Class imbalance means that the number of samples in some categories far exceeds that in others, which can negatively impact the model's training effectiveness, leading to poor learning performance and reduced overall accuracy. In such cases, the model tends to predict the majority classes, neglecting the minority class samples, thereby failing to accurately identify and classify the fault patterns in these minority classes.

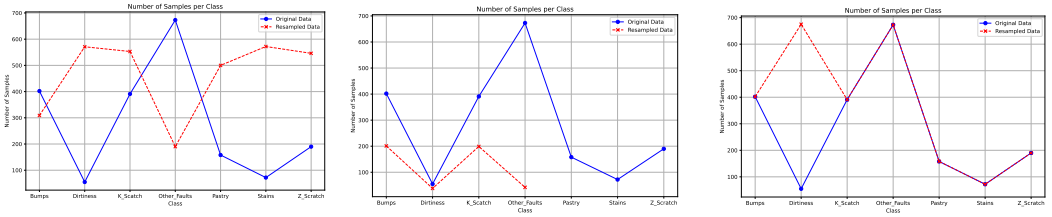
To address the class imbalance issue in the dataset, this study employs a combined sampling method using ENN and the SMOTE to reconstruct the dataset. The processed fault types and the number of associated instances are shown in Table 4. After processing, there were 500 records of Pastry, 546 records of Z-Scratch, 554 records of K-Scratch, 572 records of Stains, 571 records of Dirtiness, 309 records of Bumps, and 190 records of Other Faults. This results in a more balanced dataset compared to the original. Figure 5 illustrates the

data fluctuation before and after processing, showing that the class distribution curve is much smoother after applying the ENN-SMOTE treatment than it was without it.

In Figure 5a, the ENN-SMOTE treatment results in a much smoother class distribution, with the curve indicating that the number of samples across the classes has become more uniform. In (b) and (c), the comparison with other methods like KNN and ADASYN further demonstrates that ENN-SMOTE offers a more stable distribution, minimizing the class imbalance issue while preserving the integrity of the data. This improved balance allows the model to more effectively capture and classify patterns across all fault types, leading to better generalization and more accurate fault predictions.

Table 4. Fault types and number of related instances processed by ENN-SMOTE.

No	Fault Type	Before	After
1	Pastry	158	500
2	Z-Scratch	190	546
3	K-Scratch	391	554
4	Stains	72	572
5	Dirtiness	55	571
6	Bumps	402	309
7	Other Faults	367	190
Total Number of Samples		1635	3242



(**a**) ENN-SMOTE (**b**) KNN (**c**) ADASYN

Figure 5. Data fluctuation before and after processing.

In terms of computational cost, it is important to note that while ENN-SMOTE offers a powerful method for balancing the dataset, it does incur additional computational overhead. Specifically, the process of identifying nearest neighbors and removing borderline instances can be computationally expensive, particularly for large datasets. This additional cost is primarily due to the K-nearest neighbor search and the need to calculate distances between samples. To quantify the computational cost, we measured the time taken for the resampling process compared to simpler methods such as Random Oversampling and Tomek Links. We found that ENN-SMOTE requires more processing time and memory, particularly for datasets with a large number of features. However, the performance improvements achieved in terms of accuracy and generalization justify the additional computational expense.

4.3. Classification Prediction Results

An important input parameter for the LMT is the number of trees. To achieve the highest accuracy of this method, an increment of 10 is achieved by using 1 to 100 trees, in steps of 10, to achieve the highest accuracy for this method. The results are presented in Table 5. These outputs were obtained by averaging the results from a 10-fold cross-validation. The evaluation showed that both 50 and 60 trees achieved the same highest accuracy of 93.14%. While 60 trees were highlighted, 50 trees might also serve as an optimal choice considering computational efficiency and model complexity. Beyond this peak (60 trees), the accuracy slightly declined. The evaluation results indicate that increasing

the number of trees beyond 50 does not necessarily lead to higher accuracy, as both 50 and 60 trees achieved the same maximum accuracy of 93.14%. While increasing the number of trees might enhance model stability in some cases, it also results in greater computational costs. Therefore, selecting an intermediate number of trees, such as 50, provides a balance between performance and computational efficiency.

When the ensemble includes a small number of trees, the likelihood of misclassification increases due to the instability of individual decision trees. With a small ensemble size, each classifier has a significant impact on the final prediction. If the quality of the individual members is poor, the overall performance is consequently affected. Therefore, increasing the number of trees is necessary to mitigate the influence of low-quality members.

Table 5. Ten-fold cross validation results.

Number of Trees	1	10	20	30	40	50	60	70	80	90	100
Accuracy (%)	88.38	92.48	93.09	93.09	93.12	93.14	93.14	93.09	92.82	92.82	92.94

LSTM excels at capturing temporal dependencies in sequential data but may not be as effective in learning static features. In contrast, the LMT, as an ensemble learning method, can combine the results of multiple weak classifiers to better capture complex features and handle nonlinear relationships in the data. When the LMT is used to generate new features, these features are often more discriminative than the original ones. By combining these two approaches, we leverage the LMT's feature extraction capability and LSTM's ability to learn temporal dependencies, thereby enhancing the overall model performance.

Given that 50 trees achieved the highest accuracy, the number of trees was set to 50. Based on this, the classification results were fed into a multi-layer LSTM network. The classification results obtained are presented in Table 6. The classification accuracy for each category exceeded 93.6%, with the specific accuracy values provided in the table, or higher, and the overall model accuracy was as high as 98.1%.

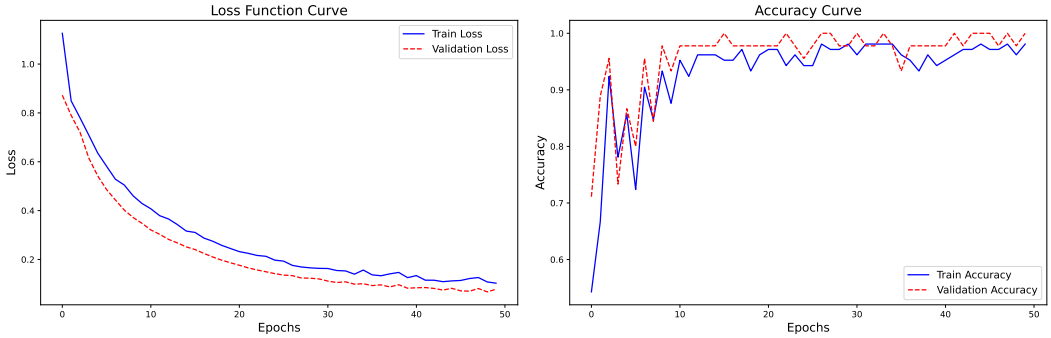
Table 6. LMT-LSTM classification results.

Class	Precision	Recall	F1 Score
Bumps	0.936	0.936	0.936
Dirtiness	0.985	1.000	0.992
K-Scratch	1.000	0.991	0.996
Other Faults	0.967	0.879	0.921
Pastry	0.979	0.904	0.940
Stains	0.990	1.000	0.995
Z-Scratch	0.990	1.000	0.995
Accuracy			0.981
Weighted Avg	**0.981**	**0.982**	**0.981**

4.4. Training Process Analysis

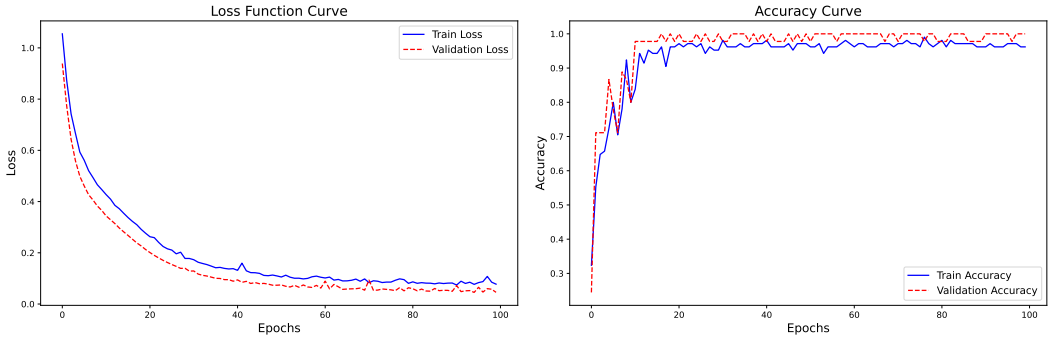
Figure 6 depicts the training and validation loss and accuracy curves over 50 epochs, providing insights into the model's early-stage training dynamics. In subfigure (a), the training and validation loss decrease steadily, showing proper convergence and indicating that the model effectively minimizes the error on both datasets. Subfigure (b) illustrates the training and validation accuracy, with a rapid improvement observed in the initial epochs, followed by stabilization. The alignment of the validation accuracy with the training accuracy suggests that the model achieves good generalization within 50 epochs, with minimal overfitting.

Figure 7, on the other hand, extends the training period to 100 epochs, offering a deeper view into the model's performance over a longer duration. In subfigure (a), the training and validation loss continue to decrease and stabilize as the training progresses, with the validation loss remaining close to the training loss. This further confirms the model's stability and its ability to avoid overfitting over an extended training period. Subfigure (b) presents the accuracy curves, which improve gradually and stabilize near the maximum accuracy. The validation accuracy closely follows the training accuracy throughout the 100 epochs, highlighting the model's robustness and ability to generalize effectively to unseen data.



(**a**) (**b**)

Figure 6. Loss function and accuracy curve for epoch 50. (**a**) Training and validation loss (epochs = 50). (**b**) Raining and validation accuracy (epochs = 50).



(**a**) (**b**)

Figure 7. Loss function and accuracy curve for epoch 100. (**a**) Training and validation loss (epochs = 100). (**b**) Raining and validation accuracy (epochs = 100).

Based on the analysis of the loss and accuracy curves for both 50 and 100 epochs, it is evident that the model achieves satisfactory convergence within 50 epochs, as the loss stabilizes and the training and validation accuracy curves align closely, indicating minimal overfitting. Extending the training to 100 epochs shows only marginal improvements in the loss and accuracy, with a slight risk of overfitting as the gap between training and validation loss begins to widen slightly. Additionally, the computational cost of training for 100 epochs outweighs the negligible performance gain observed. Therefore, 50 epochs is a more efficient and optimal choice, providing a balance between computational efficiency and model performance.

4.5. Comparative Analysis of Results

In our study, we will conduct a comparative analysis using data processed with ENN-SMOTE and unprocessed data with the same model. Additionally, we will explicitly

compare the performance of traditional machine learning methods, such as decision tree (DT) and Random Forest (RF), with the proposed machine learning and neural network-based steel plate defect prediction model. The comparison highlights the relative strengths and weaknesses of each approach in terms of metrics such as Accuracy, Precision, Recall, and F1 Score. Specifically, the original data and the data processed with ENN-SMOTE will be compared across the models LMT, LSTM, and LMT-LSTM. By comparing the performance of each model on metrics such as Accuracy, Precision, Recall, and F1 Score, we validate the superiority of the proposed model in steel plate defect prediction.

The comparative results for the original dataset are shown in Table 7, while the results for the ENN-SMOTE processed data are shown in Table 8. The confusion matrix comparison for the original data is illustrated in Figure 8, and the confusion matrix comparison for the ENN-SMOTE processed data is shown in Figure 9. Additionally, the bar chart comparison of scores for the ENN-SMOTE processed data is presented in Figure 9.

As shown in Table 7, the traditional methods, DT and RF, achieved lower performance on the original dataset, with accuracy values of 73.1% and 73.8%, respectively. These results highlight their limited ability to handle the challenges of unprocessed data, such as imbalances and noise. On the other hand, as shown in Table 8, after applying ENN-SMOTE preprocessing, the performance of DT and RF significantly improved to 86.1% and 91.8%, respectively. However, their results are still lower than the proposed LMT-LSTM model, which achieved an accuracy of 98.2%, demonstrating the advantages of the hybrid approach in both raw and preprocessed data scenarios.

Table 7. Comparison of traditional methods (DT, RF) and advanced methods (LMT, LSTM, LMT-LSTM) on the original dataset.

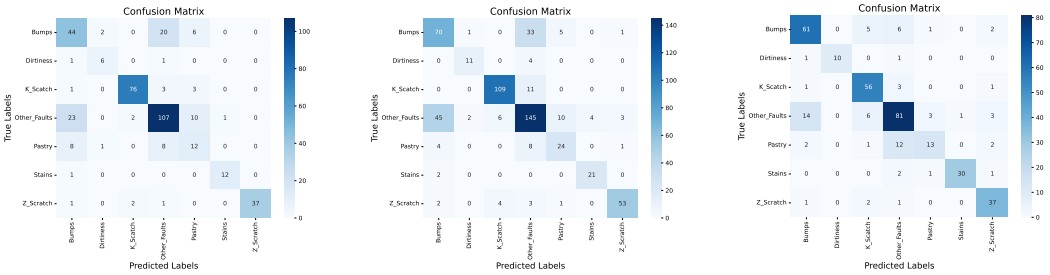
	Accuracy	Precision	Recall	F1 Score
DT	0.731	0.734	0.731	0.732
RF	0.738	0.736	0.738	0.739
LMT	0.748	0.761	0.748	0.753
LSTM	0.742	0.748	0.742	0.744
LMT-LSTM (ours)	0.814	0.818	0.814	0.810

Table 8. Comparison of traditional methods (DT, RF) and advanced methods (LMT, LSTM, LMT-LSTM) on ENN-SMOTE processed data.

	Accuracy	Precision	Recall	F1 Score
DT	0.861	0.864	0.861	0.862
RF	0.918	0.916	0.918	0.919
LMT	0.931	0.934	0.931	0.932
LSTM	0.952	0.951	0.952	0.951
LMT-LSTM (ours)	0.982	0.981	0.982	0.981

The results without data preprocessing clearly show that the model proposed in this article has the best performance of 81.4%, outperforming traditional methods like DT (73.1%) and RF (73.8%). Similarly, after data preprocessing with ENN-SMOTE, the proposed LMT-LSTM model achieves a significant improvement with an accuracy of 98.2%, compared to DT (86.1%) and RF (91.8%).

Figure 9 illustrates the comparison of confusion matrices and performance metrics (Precision, Recall, and F1 Score) across three different models, the LMT, LSTM, and the proposed LMT-LSTM, using both processed and unprocessed datasets. Panels (a), (b), and (c) correspond to the performance of the LMT and LSTM models, while panels (d), (e), and (f) show the results for the LMT-LSTM model.



(**a**) (**b**) (**c**)

Figure 8. The confusion matrix comparison for the original data. (**a**) Original data using LMT (73%). (**b**) Original data using LSTM (74%). (**c**) Original data using LMT-LSTM (81%).

The results indicate that the proposed LMT-LSTM model achieved the best overall performance, with a classification accuracy of 98.2%, which significantly outperforms the LMT model (93.1%) and the LSTM model (95.2%). This is evident from the confusion matrix in panel (e), where most of the diagonal elements demonstrate high true positive rates for all defect classes, indicating superior classification accuracy and minimal misclassification errors compared to panels (a) and (c).

Additionally, the bar charts in panels (b), (d), and (f) provide a detailed comparison of the Precision, Recall, and F1 Score for each defect category. For all defect classes, the LMT-LSTM model consistently achieved higher metric values than the LMT and LSTM models.

Moreover, the figures highlight the benefits of data preprocessing. By comparing the results for processed and unprocessed datasets, it is evident that the processed dataset contributes significantly to the performance improvement across all the models. This is reflected in the higher metric values and reduced misclassification errors when using the processed dataset.

Table 9 summarizes previous studies and methods applied to the same dataset, highlighting a variety of traditional and advanced approaches used in the field. It includes algorithms such as DT, SVM, K-nearest neighbors (KNNs), and more sophisticated techniques like the LSTM networks and PCA-based decision tree forests (PDTDFs). The accuracy achieved by these methods varies, with some incorporating optimization techniques like I-PDTDF or advanced instance selection strategies such as ENN-MQRWA. By presenting these results, the table emphasizes the state-of-the-art methods' limitations in achieving higher accuracy. In contrast, our method achieves the highest accuracy, demonstrating its clear advantage and superior performance. This comparison underscores the effectiveness and innovation of our approach in surpassing prior methods and advancing the field.

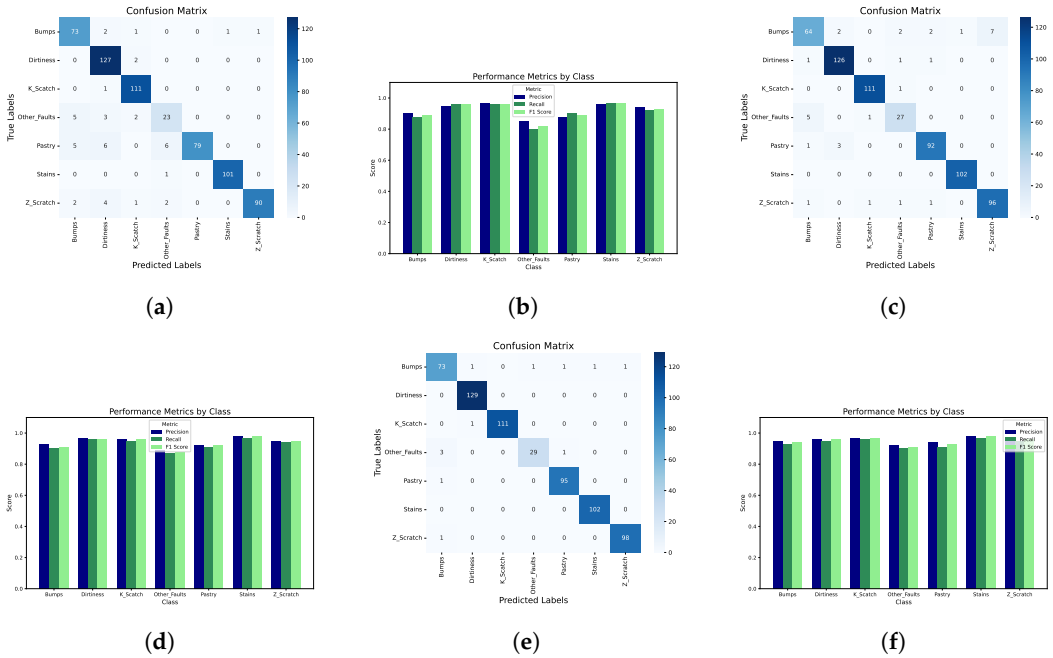
Table 9. The comparison of the state-of-the-art methods on the same dataset.

Reference	Year	Method	Accuracy (%)
Shu et al. [22]	2023	Extended decision label annotation (ELA) for support vector machines	77.53
	2023	C4.5 algorithm with enhanced decision label annotation	75.42
Agrawal and Adane [30]	2022	LSTM network applied to sequential data	75.21
	2022	PCA-based decision tree forest (PDTDF)	76.19
	2022	Improved PDTDF incorporating optimization (I-PDTDF)	76.09

Table 9. *Cont.*

Reference	Year	Method	Accuracy (%)
Ju et al. [32]	2022	RBF-SVM for feature-based classification	62.80
	2022	Decision trees using CART methodology	66.29
	2022	Neighborhood classifier (NEC) approach	65.68
Zhang et al. [33]	2022	Bi-selection method using fuzzy rough sets (BSFRS)	69.18
	2022	Instance selection method (CDIS-MQRWA) based on central density	71.14
	2022	Edited Nearest Neighbor instance selection (ENN-MQRWA)	73.72
Mohamed and Samsudin [24]	2021	Naive Bayes for probabilistic classification	69.20
	2021	K-nearest neighbors (KNNs) for instance-based learning	75.10
	2021	Classification using decision tree (DT) algorithm	75.10

In summary, the proposed LMT-LSTM model, combined with effective data preprocessing techniques, not only achieves the highest accuracy but also demonstrates consistent improvements in Precision, Recall, and F1 Score across all defect categories, making it a highly reliable approach for defect detection in steel plates.



(**a**) (**b**) (**c**)

(**d**) (**e**) (**f**)

Figure 9. The confusion matrix comparison and performance metrics (Precision, Recall, F1 Score) for each defect class for the processed data. (**a**) Processed data using LMT. (**b**) Processed data using LMT. (**c**) Processed data using LSTM. (**d**) Processed data using LSTM. (**e**) Processed data using LMT-LSTM. (**f**) Processed data using LMT-LSTM.

5. Conclusions and Future Work

The experimental results demonstrate that preprocessing imbalanced data with ENN-SMOTE, followed by classification and prediction using the LMT-LSTM model, significantly improves classification performance. Specifically, the proposed method achieved an overall classification accuracy of 98.2%, surpassing other methods such as the LMT (93.1%) and LSTM (95.2%). Additionally, category-specific F1 scores ranged from 92.1% to 99.5%, showcasing the robustness of the model across various fault types. The training process, reflected in the loss and accuracy curves, illustrates the LMT-LSTM model's ability to converge effectively and generalize well, confirming its robustness and reliability for fault classification tasks. These findings validate the theoretical and practical effectiveness of the proposed approach, providing strong support for intelligent manufacturing and quality control in industrial production.

We acknowledge the importance of validating the model in real industrial scenarios, and we have indeed performed such validation. Our model was tested in collaboration with industry partners, where data from actual production lines were used. This real-world validation was conducted to assess the model's performance under practical conditions, where the data are noisy, incomplete, and subject to varying operational conditions. The results demonstrated that our model is effective in detecting faults in the industrial environment, even with the challenges posed by real-world data.

However, despite these promising results, there are certain limitations that need to be addressed. The model's applicability is primarily based on the dataset used in this study, limiting its generalizability to other industrial environments. Additionally, the high computational complexity of the stacked LSTM could pose challenges in real-time applications or large-scale deployments, where low-latency and resource efficiency are critical. While the training time and resource consumption are manageable in controlled settings, they may require optimization for real-time, large-scale applications.

Future work will focus on expanding the model's validation across diverse industrial datasets to assess its robustness and generalizability in a wider range of operational conditions. This will include testing in real-time production environments with varying fault types, operational states, and environmental factors. Furthermore, we plan to explore techniques for improving scalability and real-time performance, such as model pruning, quantization, and hardware acceleration (e.g., GPUs and TPUs), to enhance the model's suitability for large-scale applications.

Moreover, we aim to investigate the integration of state-of-the-art deep learning techniques, such as transfer learning and multi-task learning, to improve the model's adaptability to new, unseen industrial conditions. Finally, we will explore cross-domain validation and long-term deployment scenarios to evaluate the model's long-term stability and reliability in industrial settings. These efforts will contribute to the development of a more robust, scalable, and efficient model that can be deployed in diverse industrial environments, ultimately advancing the level of automation and intelligence in manufacturing processes.

Author Contributions: The contributions of the authors are as follows: S.Z. was responsible for realizing most of the prototype and conducting the experiments. M.Z. assisted in comparative studies experiments to analyze the model structure. C.W. was involved in finishing the writing, and gave instructive suggestions on the assessment methodology and experiments. C.B. provided valuable suggestions on the model design, as well as the manuscript writing. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Key Research and Development Program of China under Grant 2021YFB3301300; in part by the National Natural Science Foundation of China

under Grant 62203213; and in part by the Natural Science Foundation of Jiangsu Province under Grant BK20220332.

Data Availability Statement: The data presented in this study is available on: https://archive.ics.uci.edu/dataset/198/steel+plates+faults, accessed on 20 January 2023.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AUC	Area Under the Curve
CNN	Convolutional Neural Networks
DT	Decision Tree
ENN	Edited Nearest Neighbor
ENN-SMOTE	Edited Nearest Neighbors-Synthetic Minority Over-sampling Technique
KNN	K-Nearest Neighbor
LMT	Logistic Model Tree (Forest)
LSTM	Long Short-Term Memory
ML	Machine Learning
NB	Naive Bayes
NN	Neural Network
PDTDF	PCA-based Decision Tree Forests
RF	Random Forest
SMOTE	Synthetic Minority Oversampling Technique
SVM	Support Vector Machine
XGBoost	Extreme Gradient Boosting

Nomenclature

Symbol	**Definition**
x	Input feature vector
y	Output label
$\hat{y}$	Predicted output
w	Weight parameters of the neural network
b	Bias term in the neural network
σ	Activation function (e.g., sigmoid or ReLU)
L	Loss function (e.g., cross-entropy)
t	Time step in sequential data
h_t	Hidden state at time t in LSTM
c_t	Cell state at time t in LSTM
f_t	Forget gate activation in LSTM
i_t	Input gate activation in LSTM
o_t	Output gate activation in LSTM
$\tilde{C}_t$	Candidate cell state in LSTM
W	Weight matrix in LSTM gates
$d(x_i, x_j)$	Distance between samples x_i and x_j
δ	Random number for synthetic sample generation in SMOTE
α	Learning rate of the optimization algorithm
∇	Gradient operator for updating weights
X	Input data matrix (feature set)
Y	Output labels for the dataset
N	Total number of samples

$\mathcal{L}$	Total loss value in the training process
ϵ	Convergence threshold for the optimization algorithm
AUC	Area Under the Curve

References

1. Chongwatpol, J. Prognostic analysis of defects in manufacturing. *Ind. Manag. Data Syst.* **2015**, *115*, 64–87. [CrossRef]
2. Wang, C.; Lu, N.; Cheng, Y.; Jiang, B. A Data-Driven Aero-Engine Degradation Prognostic Strategy. *IEEE Trans. Cybern.* **2021**, *51*, 1531–1541. [CrossRef]
3. Yang, J.; Li, S. Using Deep Learning to Detect Defects in Manufacturing: A Comprehensive Survey and Current Challenges. *Materials* **2020**, *13*, 5755. [CrossRef]
4. Mahesh, B. Machine learning algorithms-a review. *Int. J. Sci. Res.* **2020**, *9*, 381–386. [CrossRef]
5. Abdollahi, A.; Pradhan, B. Explainable artificial intelligence (XAI) for interpreting the contributing factors feed into the wildfire susceptibility prediction model. *Sci. Total Environ.* **2023**, *879*, 0048–9697. [CrossRef] [PubMed]
6. Runge, J.; Saloux, E. A comparison of prediction and forecasting artificial intelligence models to estimate the future energy demand in a district heating system. *Energy* **2023**, *269*, 0360–5442. [CrossRef]
7. Sawhney, R.; Malik, A.; Sharma, S.; Narayan, V. A comparative assessment of artificial intelligence models used for early prediction and evaluation of chronic kidney disease. *Decis. Anal. J.* **2023**, *6*, 2772–6622. [CrossRef]
8. Landwehr, N.; Hall, M.; Frank, E. Logistic model trees. *Mach. Learn.* **2005**, *59*, 161–205. [CrossRef]
9. Nhu, V.H.; Mohammadi, A.; Shahabi, H.; Ahmad, B.B.; Al-Ansari, N.; Shirzadi, A.; Geertsema, M.; RKress, V.; Karimzadeh, S.; Valizadeh Kamran, K.; et al. Landslide Detection and Susceptibility Modeling on Cameron Highlands (Malaysia): A Comparison between Random Forest, Logistic Regression and Logistic Model Tree Algorithms. *Forests* **2020**, *11*, 830. [CrossRef]
10. Pham, B.T.; Phong, T.V.; Nguyen, H.D.; Qi, C. A Comparative Study of Kernel Logistic Regression, Radial Basis Function Classifier, Multinomial Naïve Bayes, and Logistic Model Tree for Flash Flood Susceptibility Mapping. *Water* **2020**, *12*, 239. [CrossRef]
11. Amirruddin, A.D.; Muharam, F.M.; Ismail, M.H.; Tan, N.P.; Ismail, M.F. Synthetic Minority Over-sampling TEchnique (SMOTE) and Logistic Model Tree (LMT)-Adaptive Boosting algorithms for classifying imbalanced datasets of nutrient and chlorophyll sufficiency levels of oil palm (Elaeis guineensis) using spectroradiometers and unmanned aerial vehicles. *Comput. Electron. Agric.* **2022**, *193*, 0168–1699.
12. Zhou, S.; Wang, S.; Wu, Q.; Azim, R.; Li, W. Predicting potential miRNA-disease associations by combining gradient boosting decision tree with logistic regression. *Comput. Biol. Chem.* **2020**, *85*, 107200. [CrossRef] [PubMed]
13. Lindemann, B.; Maschler, B.; Sahlab, N.; Weyrich, M. A survey on anomaly detection for technical systems using LSTM networks. *Comput. Ind.* **2021**, *131*, 103498. [CrossRef]
14. Tang, Q.; Shi, R.; Fan, T. Prediction of financial time series based on LSTM using wavelet transform and singular spectrum analysis. *Math. Probl. Eng.* **2021**, *2021*, 9942410. [CrossRef]
15. Wu, K.; Wu, J.; Feng, L. An attention-based CNN-LSTM-BiLSTM model for short-term electric load forecasting in integrated energy system. *Int. Trans. Electr. Energy Syst.* **2021**, *31*, e12637. [CrossRef]
16. Li, J.; Yang, B.; Li, H.; Wang, Y. DTDR–ALSTM: Extracting dynamic time-delays to reconstruct multivariate data for improving attention-based LSTM industrial time series prediction models. *Knowl.-Based Syst.* **2021**, *211*, 106508. [CrossRef]
17. Wen, Q.; Zhou, T.; Zhang, C. Transformers in time series: A survey. *arXiv* **2022**, arXiv:2202.07125.
18. Zhao, H. Graph Neural Networks for Predictive Maintenance and Fault Diagnosis. *IEEE Trans. Ind. Informatics.* **2022**, *18*, 2421–2430.
19. Li, J. Hybrid CNN-LSTM Networks for Predictive Maintenance in Industrial Systems. *Sensors* **2023**, *23*, 912.
20. Zhu, Y. CNN-Transformer Hybrid Models for Fault Diagnosis in Complex Industrial Data. *Neural Netw.* **2024**, *147*, 80–91.
21. Ghasemkhani, B.; Yilmaz, R.; Birant, D.; Kut, R.A. Logistic Model Tree Forest for Steel Plates Faults Prediction. *Machines* **2023**, *11*, 679. [CrossRef]
22. Shu, W.; Yan, Z.; Yu, J.; Qian, W. Information gain-based semi-supervised feature selection for hybrid data. *Appl. Intell.* **2023**, *53*, 7310–7325. [CrossRef]
23. Zhang, X.; Mei, C.; Chen, D.; Yang, Y. A fuzzy rough set-based feature selection method using representative instances. *Knowl. Based Syst.* **2018**, *151*, 216–229. [CrossRef]
24. Mohamed, R. An Optimized Discretization Approach using k-Means Bat Algorithm. *Turk. J. Comput. Math. Educ. (TURCOMAT)* **2021**, *12*, 1842–1851.
25. Mohamed, R.; Yusof, M.M.; Wahid, N. Bat algorithm and k-means techniques for classification performance improvement. *Indones J. Electr. Eng. Comput. Sci.* **2019**, *15*, 1411–1418. [CrossRef]
26. Taşar, B. Comparison Analysis of Machine Learning Algorithms for Steel Plate Fault Detection. *Düzce Üniversitesi Bilim Ve Teknol. Derg.* **2022**, *10*, 1578–1588. [CrossRef]

27. Nkonyana, T.; Sun, Y.; Twala, B.; Dogo, E. Performance Evaluation of Data Mining Techniques in Steel Manufacturing Industry. *Procedia Manuf.* **2019**, *35*, 623–628. [CrossRef]
28. Mary, D. Constructing optimized Neural Networks using Genetic Algorithms and Distinctiveness. In Proceedings of the 1st ANU Bio-Inspired Computing Conference (ABCs 2018), Canberra, Australia, 20 July 2018; pp. 1–8.
29. Thirukovalluru, R.; Dixit, S. Generating feature sets for fault diagnosis using denoising stacked auto-encoder. In Proceedings of the 2016 IEEE International Conference on Prognostics and Health Management (ICPHM), Ottawa, ON, Canada, 20–22 June 2016; pp. 1–7.
30. Agrawal, L.; Adane, D. Ensembled Approach to Heterogeneous Data Streams. *Int. J. Next-Gener. Comput.* **2022**, *13*, 1014–1020.
31. Elanangai, V.; Vasanth, K. An automated steel plates fault diagnosis system using adaptive faster region convolutional neural network. *J. Intell. Fuzzy Syst.* **2022**, *43*, 7067–7079. [CrossRef]
32. Ju, H.; Ding, W.; Shi, Z.; Huang, J.; Yang, J.; Yang, X. Attribute Reduction with Personalized Information Granularity of Nearest Mutual Neighbors. *Int. J. Next Gener. Comput.* **2022**, *613*, 114–138. [CrossRef]
33. Zhang, X.; Mei, C.; Li, J.; Yang, Y.; Qian, T. Instance and Feature Selection Using Fuzzy Rough Sets: A Bi-Selection Approach for Data Reduction. *IEEE Trans. Fuzzy Syst.* **2022**, *31*, 1–15. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.