A Novel Deep Parallel Time-Series Relation Network for Fault Diagnosis

Chun Yang, Jiyang Zhang, Yang Chang, Jianxiao Zou, Member, IEEE, Zhiliang Liu, Member, IEEE, and Shicai Fan

Abstract—Currently, deep learning-based methods are widely used in the fault diagnosis of time-series data for their high precision. However, the application of traditional deep learning fault diagnosis models is limited by their calculational efficiency and poor interpretation ability. To address the problems, a fault diagnosis model named the deep parallel time-series relation network (DPTRN) is proposed in this article. There are three main advantages of the DPTRN. First, our proposed time relationship unit can perform feature extraction on each time node of a time-series sample simultaneously; therefore, the DPTRN performs fault diagnosis in a parallel way and improves the computing efficiency significantly. Second, by improving the absolute position embedding, our novel decoupling position embedding unit can be directly applied for fault diagnosis and can learn contextual information. Third, our proposed DPTRN has an obvious advantage in feature interpretability compared with traditional deep learning-based models. Applying the DPTRN model on four datasets, we achieved higher diagnosis performance with much lower cost, which indicates the effectiveness, efficiency, and interpretability of the proposed DPTRN model.

Index Terms—Decoupling position embedding unit, parallel computation, time relationship unit, time-series fault diagnosis.

I. INTRODUCTION

With the increasing complexity of the modern industry, lots of faults might change continuously and slowly over time, which makes precise fault diagnosis more difficult. At present, a large number of studies have applied deep learning to implement fault diagnosis with time-series data [1]. Cabrera et al. [2] used long short-term memory (LSTM) to implement fault diagnosis on reciprocating compression machinery. Zhang et al. [3] applied the bidirectional recurrent neural network (BiRNN) to implement fault diagnosis on the TE dataset. Lavrova et al. [4] used the gated recurrent unit (GRU) to analyze time-series data and carry out anomaly detection. Besides, a large number of scholars have applied RNN [5], [6], [7], LSTM [8], [9], [10], [11], GRU [12], [13], [14], and convolutional neural networks (CNNs) [15], [16], [17], [18] to solve the problem of fault diagnosis for time-series data.

However, traditional network structures have two critical defects.

1) Low Computational Efficiency: In some industrial production processes, fast incremental training and inference are required for timely fault isolation or control strategy switch [19], [20], [21], [22], [23]. The traditional network structure used in the deep learning models is either a serial calculation method or requires a deeper network structure or larger convolution kernels to extract long-term evolution features, which makes them inefficient and unable to meet the increasing requirements for real-time fault diagnosis.

2) Poor Interpretation Ability: A fault diagnosis model with good interpretability can help the workers analyze the cause of faults and improve the understanding of fault mechanism [24], [25], [26], [27]. The traditional neural network structures use hidden vectors to represent the contextual information of time-series data, such as the memory unit in RNN and the output of each convolutional layer in CNN. These hidden vectors have no physical meaning, and their low interpretability makes it challenging to understand the data change process.

To address these problems, this article proposes a fault diagnosis method named the deep parallel time-series relation network (DPTRN) that is composed of three parts: a time relationship unit, a decoupling position embedding unit, and a classification layer.

For the proposed time relationship unit, it is constructed with an MLP structure with shared weights, which is inspired by the research in the field of computer vision (CV) and natural language processing (NLP) [28], [29], [30]. It can exploit all time nodes of a time-series sample in a parallel way and does
not require a deep structure or many parameters, which guarantees the feature extraction efficiency of the DPTRN. However, the time relationship unit cannot extract the contextual feature of the industrial process data directly because adding position embedding directly to the original features will change the physical meaning of the feature channels, which will degrade the fault diagnosis performance.

Therefore, for the second part, we propose a decoupling position embedding unit in the framework that considers the contextual information of the time-series data. In the decoupling position embedding unit, learnable parameters are used to optimize the positional embedding, and it is added to the output of the time relationship unit rather than directly to the raw data, avoiding adding noise to the original features.

The third part is a classification layer constructed with MLP. By combining the outputs of the time relationship unit and decoupling position embedding unit with the features of each time node, a comprehensive historical information vector that integrates the evolution information of the time-series data will be input into the classification layer together with the features of the current time node.

For our proposed DPTRN model, the proposed time relationship unit and the decoupling position embedding unit are two effective feature extractors, which promise diagnosis accuracy and good interpretability. Moreover, the simple structure and fewer parameters improve the efficiency by the way of parallel computing. The superiority of the proposed model is validated on the public TE dataset, the KDD-CUP99 dataset, and our PEMFC dataset, WHELL dataset.

The main contributions of this article are summarized as follows.

1) A time relationship unit was proposed in this article to extract the relationship between each historical time node and the current time node in a parallel way, which indicates the importance of each historical time node for fault diagnosis. While ensuring the feature extraction capability of the model, the computational efficiency of fault diagnosis was greatly improved. Furthermore, the number of parameters of the proposed model is greatly reduced, resulting in a significant reduction in the computational cost required for training and inference.

2) In order to fully use the contextual information of the time-series data, we proposed a novel unit, named the decoupling position embedding unit, in the diagnosis framework. The decoupling position embedding unit overcomes the defect that absolute position embedding cannot be directly applied to the field of fault diagnosis. It significantly improved the performance of fault diagnosis in the industrial process.

3) Our proposed DPTRN model provided a more convenient way to interpret the output compared with traditional deep learning networks. For the time relationship unit, its output explains the importance of each historical time node for fault diagnosis, and the output of the decoupling position embedding unit explains the importance of historical information over time.

The rest of this article is organized as follows. Section II details the pipeline used for fault diagnosis and the proposed DPTRN. In Section III, four datasets are introduced separately, and the fault diagnosis and fault detection cases of the four datasets verify the high efficiency, effectiveness, and interpretability of the proposed method. Finally, conclusions are drawn in Section IV.

II. FRAMEWORK OF THE DPTRN

This section introduces the DPTRN method for fault diagnosis of time-series data in detail. The framework and illustration of the proposed method are shown in Fig. 1. Specifically, the fault diagnosis procedure can be described as follows.

1) Establish time-series data training set \( X = (x_1, x_2, \ldots, x_n) \in R^{N \times T \times M} \), where \( N \), \( T \), and \( M \) are the number of samples, the length of the time series, and the number of feature dimensions, respectively. \( x_i \in R^{T \times M} \) is the \( i \)th sample of the training set.

2) For sample \( x_i \), the sample is divided into current time node \( D(T) \in R^M \) (the green vector in Fig. 1) and historical time node \( D(k) \in R^M, k \in 0, 1, \ldots, T - 1 \) (the blue vector in Fig. 1).

3) Each historical time node \( D(k) \) is inputted into the time relationship unit together with the current time node \( D(T) \) to obtain the preliminary relationship weight, which is a scalar. Details of the time relationship unit are introduced in Section II-A.

4) For each historical time node \( k \) and current time node \( T \), DPTRN first generates absolute position embeddings (the purple vector and brown vector in Fig. 1), then uses the position query matrix (the blue matrix in Fig. 1) and the position key matrix (the green matrix in Fig. 1) to perform multiple mappings, and finally, generates decoupling position embeddings, which are scalar as well. Details regarding the decoupling position embedding unit are introduced in Section II-B.

5) The output of the time relationship unit and decoupling position embedding unit is added to obtain the time relationship weight corresponding to each historical time node. A comprehensive historical information vector is obtained by weighting and sum-pooling the features of corresponding historical time nodes and time relationship weights.

6) Finally, the historical information vector contacts the features of the current time node and is input into the classification layer to obtain the fault diagnosis result. Details of the classification layer are explained in Section II-C.

A. Time Relationship Unit

The detailed structure of the time relationship unit is shown in the right part of Fig. 1. The purpose of designing the time relationship unit is to analyze the relationship between each historical time node and the current time node of the time-series data, that is, the importance of each historical time node for diagnosing the current sample. The impact of time nodes that are not important for diagnosis through the output of the time relationship unit is expected to be weakened, while, on the other hand, the impact of time nodes...
that are useful for diagnosis is expected to be enhanced. The specific implementation of the time relationship unit is a deep MLP network. Therefore, the unit can achieve long-term feature interaction in a parallel way, which greatly improves computational efficiency.

As shown in the figure, the features of the historical time node and the current time node will be linearly operated and contacted before being input into the MLP network. The vector after the linear operation and contact is $V_K$, as shown in the yellow vector in Fig. 1. The purpose of the linear operation is to improve the ability to discriminate between the features of the historical time node and the current time node, thereby enhancing the expressive ability of the time relationship unit. The DPTRN proposed in this article used addition and subtraction as linear operations because of their high efficiency and good performance. The vector $V_K$ that will be input into the MLP network in the time relationship unit is calculated as

$$V_K = [D(T), D(k), D(T) - D(k), D(T) + D(k)]$$

(1)

where the dimension of $V_K$ is $4M$. Then, the MLP outputs the preliminary relationship weight. In this article, the number of MLP layers in the time relationship unit is set to 3, which can ensure the model’s high computational efficiency while guaranteeing the model’s characterization capability. The output of the time relationship unit is

$$RW_{pre_K} = W_3\sigma(W_2(\sigma(W_1V_K + b_1)) + b_2) + b_3$$

(2)

where $RW_{pre_K}$ represents the preliminary relationship weight of the corresponding historical time node $K$, and matrices $W$ and $b$ are the corresponding network weight and bias of each layer, respectively. The activation function $\sigma(\cdot)$ that we applied is ReLU.

It is worth noting that, because the time relationship unit does not consider the context of each historical time node, $RW_{pre}$ is only the preliminary relationship weight. To significantly reduce the model size of the DPTRN and the difficulty of model training, the time relationship unit shares the weight for all historical time nodes as LSTM and CNN methods do. In addition, to highlight the importance of a specific historical time node and prevent important historical nodes from being smooth, the time relationship unit will not perform softmax normalization on the final $RW_{pre}$, which means that the sum of $RW_{pre}$ values at each historical time node is not 1. The time relation unit interacts with features of each time node of the time-series data in a novel structure, which significantly improves the diagnostic efficiency of the model.

B. Decoupling Position Embedding Unit

The detailed structure of the decoupling position embedding unit is shown in the green part of Fig. 1. The design of the decoupling position embedding unit was inspired by the introduction of absolute position embedding when the Transformer model in the NLP field processed the text sequence data [31]. The absolute position embedding was proposed in Transformer to introduce contextual information on time-series data for performance improvement. The Transformer uses the trainable position embedding, which is directly added to the original features, to introduce contextual features. However, absolute position embedding cannot be applied for fault diagnosis
directly because it will introduce noise to the raw data and change the physical meaning of the feature channels.

Inspired by the above research, we propose a decoupling position embedding unit in the framework that considers the contextual information of the time-series data. In the decoupling position embedding unit, learnable parameters are used to optimize the positional embedding, and it is added to the output of the time relationship unit rather than directly to the raw data, avoiding adding noise to the original features. Our proposed decoupling position embedding unit can consider the contextual information of time-series data without interfering with the raw features in a novel way. Because the decoupling position embedding unit is an auxiliary part of the time relationship unit, and we do not want to use too many parameters for this unit, we adopt the matrix mapping method to achieve it.

For each time node in the time-series data, an absolute position embedding will first be generated

\[
\begin{align*}
PE_{K}(pos, 2) &= \sin(pos/10000^{2/d_{model}}) \\
PE_{K}(pos, 2+1) &= \cos(pos/10000^{2/d_{model}})
\end{align*}
\]  

where pos is the position of time node \( K \) in time-series data and \( d_{model} \) is the constant \( T \), which represents the length of the time-series sample. Absolute position embedding performs sin or cos transformation according to the position of each time node in the time-series data sample.

In the Transformer, the absolute position embeddings were directly added to the raw features, and the model was expected to learn the context through those embeddings. Assuming that this article had adopted the same method of introducing absolute position embeddings as Transformer, that is, position embedding without decoupling, then the model structure shown in Fig. 2 will be obtained.

The features used in the Transformer were obtained through embedding layer mapping, which was trainable and could cooperate with absolute position embedding during the training process. However, in fault diagnosis, the features are generally collected in industrial processes that are not trainable. This means that directly adding the absolute position embedding to the raw data may change the physical meaning of each channel in the feature vector, thereby reducing the model’s reliability and effectiveness. To illustrate the negative effects of absolute position coding, we conducted an ablation experiment in Section III-E.

The decoupling position embedding unit in the DPTRN is designed to guide the model to learn contextual information of time-series data without introducing noise into the raw data. For any historical time node \( K \), its decoupling position embedding is calculated as follows:

\[
DPE_{K} = (PE_{K} \odot P_{-query}) \odot (PE_{T} \odot P_{-key})
\]  

where \( PE_{K} \) and \( PE_{T} \) indicate the absolute position embeddings corresponding to time node \( K \) and current time node \( T \), respectively. \( P_{-query} \) and \( P_{-key} \) are the mapping matrix corresponding to the historical time node and the current time node, respectively, that is, the blue and green matrices in Fig. 1. To reduce computational complexity, the mapping matrix is set to a square matrix of dimension \( M \times M \). \( DPE_{K} \) is a scalar that represents the positional relationship of the current time node \( K \) in the time-series data.

The decoupling position embedding \( DPE_{K} \) and the preliminary relationship weight \( RW_{pre} \), which is obtained from the time relationship unit, are added to obtain the final relationship weight corresponding to historical time node \( K \)

\[
RW_{K} = RW_{pre} + DPE_{K}
\]  

For all historical time nodes, a relationship weight vector can be obtained

\[
RW = \left[ RW_{1}, RW_{2}, \ldots, RW_{T-1} \right] / \sqrt{M}
\]  

where \( \sqrt{M} \) is the regularization coefficient, which is beneficial for model parameter optimization. The comprehensive historical information vector can be obtained by taking the outer product of the relationship weight vector and the original feature data of each historical node and summing them one by one

\[
HI = \text{sumpooling}(RW \otimes X_{T-1})
\]  

where \( X_{T-1} \) represents samples in the time-series data except the current time node sample. Except for the sum pooling, it can also use contacting, average pooling, or other pool methods to obtain the comprehensive historical information vector. In this article, the sum pooling layer is adopted considering computational efficiency, and other pooling operations are left for follow-up research.

Then, the historical information vector \( HI \) is contacted with the original features of the current time node to obtain the
vector that will be input to the classification layer
\[ I = [H, D(T)]. \] (8)

We believe that historical information and current node information should have the same importance so that the vector \( I \) not only contains the information of the historical node but also considers the information of the current node.

C. Classification Layer

Vector \( I \) will be input into the classification layer to obtain the final fault diagnosis result. The classification layer is also a three-layer MLP structure. The output of the classification layer is calculated as follows:
\[ \hat{y} = \text{softmax}(W_3'\sigma(W_2'(W_1'I + b'_1') + b'_2') + b_3') \] (9)
where matrices \( W' \) and \( b' \) represent the corresponding network weight and bias of each layer, respectively. The activation function \( \sigma(\bullet) \) is ReLU.

The fault diagnosis result can be obtained according to the output result of the classification layer.

D. Training Strategy

As the deep neural network is difficult to train due to the internal covariate shift, the method proposed in this article used the batch normalization layer. Batch normalization can perform standardization for each batch, allowing the model to use a higher learning rate in training and reducing the training steps and the training difficulty of the method [32]. For input \( X = (x_1, x_2, \ldots, x_n) \) with \( n \) dimensions, the batch normalization layer normalizes the data in the following way:
\[ \hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\text{Var}[x_i]}} \] (10)
where \( \mu_B \) and \( \text{Var}[x_i] \) are the mean value and standard deviation of the corresponding batch, respectively. For \( \hat{x}_i \), after simple normalization, batch normalization also introduced parameters \( \gamma \) and \( \beta \) to scale and move the normalized value so that the representation between each layer is not affected by the parameters of each layer
\[ f_i = \gamma_i\hat{x}_i + \beta_i \] (11)
where \( \gamma_i \) and \( \beta_i \) are learnable parameters. Due to space limitations, this article does not give more introduction to batch normalization. For more principles and training details of batch normalization, please refer to [32].

Here, we apply batch normalization immediately before the activation layers of MLP because it can accelerate the convergence of the DPTRN end-to-end training process and improve the robustness of the method. In addition, the activation function \( f(\bullet) \) that we applied is ReLU, and its combined utilization with batch normalization can further solve the problem of deep network gradient disappearance. Therefore, it is natural to apply batch normalization in the DPTRN model. In addition, to avoid overfitting the model, this article also applied L2 regularization and dropout strategies.

The DPTRN parameters approximate the optimal solution by calculating the loss of the cross-entropy function through supervised learning. Cross-entropy loss can minimize the difference between model output and expected output
\[ L = -\sum_{i=1}^{K} y_i \log(p_i) \] (12)
where \( K \) is the total number of categories, \( y_i \) is the corresponding label (when considering the \( i \)th category, \( y_i = 1 \); otherwise, it is 0), and \( p_i \) is the output of the neural network.

In online fault diagnosis, the detected data must also be arranged in the same time-series data format as the training dataset, and the data will be input into the model according to the same data pipeline as the training dataset to obtain the corresponding fault detection results.

III. Experiment

A. Datasets

Details of the four time-series datasets used for performance comparison are described as follows.

1) The Tennessee–Eastman (TE) chemical process, which simulated a real chemical production process. It has been widely used as a benchmark process to evaluate fault detection and fault diagnosis methods. This process includes one normal state and 20 fault states. Details of the TE process can be found in [33].

2) KDD-CUP99, which recorded seven million network traffic connection records within seven weeks. The network traffic data were labeled as normal or attacked data. Additional data details can be found in [34].

3) Proton exchange membrane fuel cell (PEMFC), which simulated a real fuel cell. As shown in Fig. 3, the PEMFC system is mainly composed of a fuel cell stack, an air supply subsystem, a hydrogen supply subsystem, a gas humidification subsystem, and a water management subsystem. By simulating fault conditions artificially, the dataset formed 8 possible faults that may occur during operation, as listed in Table I. The fuel cell of this experimental platform consists of 6 single cells, whose specifications under the normal condition are shown in Table II. The features of the PEMFC dataset are the voltages of the 6 cells in the cell stack and the sampling frequency is 10 Hz.

4) Wheelset Bearing dataset (WHELL), which simulated high-speed rail wheelset bearings. As shown in Fig. 4, the wheelset bearing test platform is mainly composed of a drive motor, a belt transmission system, a vertical loading set, a lateral loading set, two fan motors, and a control system. It simulates a real train using vertical and lateral loads. In this article, this dataset is divided into four states, as shown in Table III. To simulate various complex working conditions of wheelset bearings during their operation as much as possible, under each health condition, vertical loads of 56, 146, 236, and 272 kN are set, and two lateral loads (0 and 20 kN) are set. The sampling frequency of the dataset is 5120 Hz, and each sample contains two features, which are the data collected by the sensors [15].
The main information about the four datasets used in this article is shown in Table IV, including the feature dimensions, the length of the time series, and the numbers of samples in the training set, the validation set, and the test set.

To prevent information leakage when processing datasets, this article applies data noncrossover instead of sliding windows to collect time-series data. This means that each time node data point in the dataset would only belong to training samples or test samples. For a fair comparison, we employ the same data preprocessing approach for all methods.

B. Models for Comparison

The proposed method is compared with the pure MLP, LSTM, BiLSTM, and IDCNN models to verify its validity and efficiency. The network structures of the abovementioned method for comparison are shown in Table V. Since the core innovation of this article is the proposed time relationship unit and decoupling position embedding unit that can be regarded as a new feature extractor, for fairness, the same classification layer is applied when comparing the performances of the methods. The structure of the MLP in the time relationship unit in the D PTRN is (512, 128).

Furthermore, the D PTRN is compared with three advanced fault diagnosis methods, including FDGRU [35], MCNN-LSTM [36], and 1D CNN-V AF [37]. These three methods are also deep learning methods for fault diagnosis of time-series data. Details about their network structure can be found in the relevant literature. When implementing the above three methods, this article used the data preprocessing method and network parameters recommended by the authors.

C. Implementation Details

For fairness, all the compared models mentioned in this article are trained and inferred on the same experimental platform (RTX 2080ti with 12-GB memory). The batch size of each network is set to 32, the epoch is 60, and the learning rate is $6e^{-4}$. The L2 regularization coefficient is 0.0001.
D. Case Study

In this section, we want to answer two questions.

1) Can the proposed DPTRN be guaranteed to be effective?
2) Can the proposed DPTRN have obvious advantages in computational efficiency?

To answer the first question, the performances of the KDDCUP99 dataset (fault detection) and the TE dataset, the PEMFC dataset, and the WHELL dataset (fault diagnosis) are compared with those of the various baseline methods mentioned in Section III-D1. The detailed comparison results are shown in Tables VI and VII. The best parts of the comparison results are bolded, and suboptimal results are underlined. The evaluation indicators used in the comparative experiment include the recall rate, accuracy rate, and F1 value.

To answer the second question, in Section III-D2, we compared the model size and the floating-point operations (FLOPs) of the DPTRN with those of various baseline methods to verify the efficiency of the proposed method. Model size measures the training difficulty and the complexity of the model, which represents the number of trainable parameters of a model. When the model size increases, the model will have more parameters, a better ability to fit the data, and more difficulty to converge. FLOPs measure the number of floating-point calculations required by the model to feed a sample forward, which is only related to the structure and operation of the model, and is independent of hardware. FLOP is a direct indicator of the computational efficiency of the model. The computational efficiency comparison results are shown in Table IX.

1) Effectiveness of Different Methods: Tables VI and VII show that the experimental performance of the simple MLP model is the worst. It is reasonable that it has a simple structure and does not have the ability to process a large number of features of the time-series data. In contrast, LSTM+MLP, BiLSTM + MLP, and FDGRU achieve better fault detection or fault diagnosis results than simple MLP. This is because LSTM and GRU can use a recurring unit with shared weights to extract potential features and generate a hidden state vector to represent time evolution characteristics through serial calculation. The 1DCNN and 1DCNN-VAF also achieve satisfactory fault diagnosis performance because the 1DCNN uses different-sized convolution kernels to interact features between different time nodes in time-series data. In addition, MCNN-LSTM, which combines CNN and LSTM, also achieves good results.

Compared with various baseline methods, DPTRN achieves the best experimental results on the KDD-CUP99, TE, and PEMFC datasets and suboptimal experimental results on the WHELL dataset. This is because LSTM and GRU can use a recurring unit with shared weights to extract potential features and generate a hidden state vector to represent time evolution characteristics through serial calculation. The 1DCNN and 1DCNN-VAF also achieve satisfactory fault diagnosis performance because the 1DCNN uses different-sized convolution kernels to interact features between different time nodes in time-series data. In addition, MCNN-LSTM, which combines CNN and LSTM, also achieves good results.

To further illustrate the reliability of the comparison results, we statistically test the fault diagnosis results of DPTRN with other various models, as shown in Table VIII. We used the t-test and ANOVA test; as shown in the table, the confidence level is below 0.05 on all four experimental datasets. This result further shows that the fault diagnosis performance of DPTRN is significantly better than the baseline model.

Combining the above results on effectiveness, we can answer the first question mentioned earlier and state that...
the proposed DPTRN can perform fault detection or fault diagnosis effectively and has certain advantages in comparison with other baseline methods.

2) Efficiency of Different Methods: To investigate the model efficiency, the model sizes and FLOPs of different methods are listed in Table IX. Model size measures the training difficulty and the complexity of the model, which represents the number of trainable parameters of a model. FLOPs measure the number of floating-point calculations required by the model to feed a sample forward, which is only related to the structure and operation of the model and is independent of hardware.

Due to the simple structure and parallel computing way of the simple MLP, it has fewer FLOPs, and the fault diagnosis task can be completed with increased efficiency. However, at the same time, this method uses the classification layer immediately after flattening the time-series data, which makes the input vector dimension extremely large and greatly increases the number of parameters. Since the recurrent units in LSTM and GRU share weights, the model size is greatly reduced, which reduces the difficulty of training. However, for LSTM, it is limited by its complex serial calculation design; therefore, the FLOPs of both LSTM + MLP and BiLSTM + MLP are significantly increased, which reduces their calculation efficiency. For FDGRU, it has the most parameters due to its complex structure, so it has the most FLOPs, which makes it the least efficient. The 1DCNN requires different convolution kernel sizes or deep structures to realize long-term feature interaction, which increases the model size and training difficulty of the CNN. In addition, although the CNN is computed in parallel, the complex convolution operation of the CNN will increase its number of FLOPs, so the 1DCNN is also unable to meet the requirement of real-time fault detection and fault diagnosis.

Our proposed DPTRN can extract features in parallel, which makes it more computationally efficient. As shown in Table IX, the DPTRN has obvious advantages in model size, which makes training the DPTRN easier. In addition, because of its simple structure and the parallel computation method, the DPTRN has the fewest FLOPs. Compared with the five best-performing baseline models, DPTRN’s FLOPs are 20.21%, 18.76%, 2.7%, 30.11%, and 14.33% of the number of FLOPs of the BiLSTM, 1DCNN, FDGRU, 1DCNN-VAF, and MCNN-LSTM models, respectively, which means that the proposed DPTRN has a very clear advantage in computational efficiency compared to the baseline methods. Especially when the number of FLOPs and the model size of the proposed DPTRN are only 2.7% and 2.97% of those of the FDGRU, respectively, it surpasses the FDGRU on three datasets, and the results are very close on the WHELL dataset. Considering the results of each baseline method, the parallel computing method has fewer FLOPs and higher computational efficiency, which verifies the effectiveness of parallel computing emphasized in this article.
In order to more intuitively demonstrate the advantages of the proposed method in terms of computational efficiency, we compared the training efficiency of different diagnosis methods on the TE dataset, and the results are shown in Fig. 5. The figure shows the time required for each model to train a batch on the TE dataset. It can be seen from the figure that the training efficiency of the traditional model using the serial computing method is significantly lower than that of DPTRN, and the CNN method also requires a larger amount of computing to make the model converge. The best performing FDGRU in the baseline model takes 21 times longer to train a batch than DPTRN. This means that DPTRN completes fault diagnosis with a more efficient model structure.

Fault diagnosis models for practical applications generally require incremental learning to cope with changes in data distribution, so as to ensure their fault diagnosis performance. We simulated the process on the TE dataset and compared the training efficiency of different fault diagnosis models. Specifically, we divide the TE dataset into ten parts on average, feed these data into the models step by step, and test the total time consumption of the process. The detailed comparison results are shown in Fig. 6.

As shown in Fig. 6, DPTRN only needs about 200 min to complete the incremental learning of the TE dataset, while FDGRU needs 2374 min, which is about 12 times more than DPTRN. In general, traditional fault diagnosis models require 20–40 h to complete the incremental learning of TE datasets. These models are limited in their computational efficiency, so it is very difficult and expensive to update model parameters in real time.

Furthermore, DPTRN also has obvious advantages in inference speed. We compare the time required for different fault diagnosis models to infer one sample on the TE dataset, and the results are shown in Fig. 7. The Y-axis in the figure is obtained by taking the inference time of MLP as a standard unit and comparing it with the inference time of each model, where the time required for MLP to infer a sample is 3 ms.

As shown in the figure, due to its parallel computing characteristics, DPTRN shows obvious efficiency advantages, and its inference time for a sample is very close to that of MLP. Traditional fault diagnosis models require ten times or more inference time. Therefore, DPTRN can more easily implement real-time fault diagnosis of industrial processes on a simple computing platform, which can save a lot of computing costs.

Combining the above results on efficiency, we can answer the second question mentioned earlier that the proposed DPTRN has a significant advantage in computational efficiency compared to other baseline methods.

Based on the results obtained on the above four datasets, the DPTRN method proposed in this article has the advantage of high computational efficiency due to its parallel computing characteristics. In addition, it has fewer parameters and is easier to train the model because its time relationship unit shares weights, and the model can be summed as a six-layer MLP structure. Furthermore, the DPTRN model achieves the best fault diagnosis and fault detection results on three datasets and one suboptimal result on the WHELL dataset, which further proves the effectiveness, rationality, and robustness of the time relationship unit and decoupling position embedding.
TABLE X
ABLATION EXPERIMENT RESULTS OF THE DPTRN ON THE KDDCUP99 AND TE DATASETS

| Method | KDD-CUP99 dataset | TE dataset |
|--------|-------------------|------------|
|        | Recall (%) | Accuracy (%) | F1 (%) | Recall (%) | Accuracy (%) | F1 (%) |
| DPTRN<sub>a</sub> | 0.9785±0.0017 | 0.9807±0.0014 | 0.9791±0.0019 | 0.9795±0.0026 | 0.9839±0.0028 | 0.9805±0.0032 |
| DPTRN<sub>b</sub> | 0.9713±0.0049 | 0.9758±0.0037 | 0.9723±0.0029 | 0.9314±0.0039 | 0.9590±0.0035 | 0.9340±0.0039 |
| DPTRN | 0.9870±0.0030 | 0.9870±0.0030 | 0.9870±0.0028 | 0.9943±0.0028 | 0.9943±0.0028 | 0.9943±0.0028 |

DPTRN<sub>a</sub> does not use the position embedding unit but only uses the time relationship unit. DPTRN<sub>b</sub> is the model that uses the absolute position embeddings instead of using the decoupling position embedding unit while using the time relationship unit.

TABLE XI
ABLATION EXPERIMENT RESULTS OF THE DPTRN ON THE PEMFC AND WHELL DATASETS

| Method | PEMFC dataset | WHELL dataset |
|--------|---------------|---------------|
|        | Recall (%) | Accuracy (%) | F1 (%) | Recall (%) | Accuracy (%) | F1 (%) |
| DPTRN<sub>a</sub> | 0.9755±0.0020 | 0.9707±0.0015 | 0.9741±0.0016 | 0.9741±0.0031 | 0.9817±0.0035 | 0.9787±0.0027 |
| DPTRN<sub>b</sub> | 0.9681±0.0017 | 0.9668±0.0011 | 0.9673±0.0016 | 0.9659±0.0034 | 0.9728±0.0035 | 0.9710±0.0030 |
| DPTRN | 0.9811±0.0010 | 0.9764±0.0009 | 0.9810±0.0009 | 0.9912±0.0004 | 0.9961±0.0004 | 0.9939±0.0004 |

DPTRN<sub>a</sub> does not use the position embedding unit but only uses the time relationship unit. DPTRN<sub>b</sub> is the model that uses the absolute position embeddings instead of using the decoupling position embedding unit while using the time relationship unit.

To further verify the validity and rationality of the proposed time relationship unit and the decoupling position embedding unit separately, we conducted ablation experiments. For DPTRN<sub>a</sub>, it does not use the position embedding unit but only uses the time relationship unit (that is, it directly uses the preliminary relationship weight in formula (2) as the final relationship weight). For DPTRN<sub>b</sub>, the model in Fig. 2 only uses the absolute position embeddings instead of using the decoupling position embedding unit while using the time relationship unit. In order to distinguish, the model proposed in this article is called DPTRN in the comparison. The experimental results are shown in Tables X and XI.

All of the metrics of DPTRN<sub>a</sub> on the four datasets are more than 97% and surpass or are close to those of the simple MLP, LSTM, MCNN-LSTM, and 1DCNN-VAF models. This demonstrates the effectiveness of the time relationship unit, which is better able to extract the characteristics of time-series data.

The performance metrics of DPTRN<sub>b</sub> are not very satisfying and even worse than those of the simple MLP on the TE dataset. This is consistent with the hypothesis that the absolute position embeddings cannot be directly applied in fault diagnosis because it introduces noise to the raw data.

Our proposed DPTRN model is significantly better than DPTRN<sub>a</sub> and DPTRN<sub>b</sub>, which further verifies the advantages of the framework effectiveness of the decoupling position embedding unit proposed in this article. That is, decoupling position embedding units can help models introduce contextual information from time-series data, which is a capability that the time relationship unit does not have.

The results of the above ablation experiments show that the time relationship unit can effectively extract the evolution features of the time-series data, and the decoupling position embedding unit proposed in this article can further improve the performance of the model.

F. Interpretability of the DPTRN

The proposed framework combined with the time relationship unit and the decoupling position embedding unit can significantly improve the performance, so investigating how the two units affect the results is of interest. Therefore, we randomly selected two fault samples from the TE dataset and compared them with normal samples to check the output of the two units and analyze the internal working principles of these two units.

1) Interpretability of the Time Relationship Unit: For the time relationship unit, the outputs of the two fault samples mentioned above are shown in Fig. 8. The value of the heat map is the output value of the time relationship unit. In the heatmap of each subgraph, the darker red cubes indicate that the current time nodes have a higher preliminary relationship weight, and the darker blue cubes indicate the opposite. For the left subgraph, the outputs with larger weights are dark red. Integrating the left subgraph with the right subgraph shows that the red parts map the features that indicate the significant difference between the fault sample and the normal sample. The corresponding parts of the two subgraphs are marked with black boxes, and these marked time nodes contain long-term evolutionary information that is useful for correct fault diagnosis.

In addition, the time relationship unit outputs completely different preliminary relation weights for the two types of fault
samples. For each sample, historical time nodes that contain important information for fault diagnosis may be distributed anywhere in the length of the time series, which requires the time relationship unit to output the correct weight at each time node. By using the different weights to extract the evolution information in the time-series data, the subsequent classification layer can more easily focus on these historical nodes that are beneficial to fault diagnosis, which greatly improves the performance of fault diagnosis.

2) Interpretability of the Decoupling Position Embedding Unit: Since the input of the decoupling position embedding unit is the absolute position embedding determined by the timing length, the outputs of each sample are the same and shown in Fig. 9. The value of the heat map is the output value of the decoupling position embedding unit. The weight outputs of the decoupling position embedding unit increase as the historical time node approaches the current time node. Therefore, for the decoupling position embedding unit, the closer the distance to the current time node, the more valuable the time nodes are.

It is worth noting that the outputs of the decoupling position embedding unit are learnable, which means that it has learned results that are consistent with human prior knowledge, that is, the reference value of information from a long time ago will decrease. This result shows that the decoupling position embedding unit works as we expect, that is, it introduces the context information into the model without introducing noise to the raw data.

3) Interpretability of the Relationship Weight: For the two random time-series fault samples mentioned above, the final outputs obtained by combining the time relationship unit and the decoupling position embedding unit are shown in Fig. 10. As shown in the figure, the final relationship weights become smoother and have the ability to indicate both the importance of each historical time node and the relative importance of the times nodes among the time series for fault diagnosis.

Neural network structures, such as RNN, LSTM, and GRU, use a hidden state vector to represent the features of the evolution of the time-series data, which cannot explain the specific physical meaning. CNN applies convolution kernels to complete the feature interaction of time-series data, and it is also difficult to explain the specific physical meaning of the convolution kernel. For our proposed DPTRN, the time relationship unit explains the importance of each historical time node for fault diagnosis, and the decoupling position embedding unit explains the relationship of the importance of historical information over time.
Therefore, the proposed DPTRN is more interpretable than the traditional deep neural network.

IV. CONCLUSION

In this article, a DPTRN model for fault diagnosis of time-series data is proposed. Considering that traditional deep learning networks are limited to serial computing when analyzing time-series data, the DPTRN model proposed in this article uses parallel time relationship units to extract evolutionary features and improve computational efficiency. To introduce the contextual information of time-series data to the model, a decoupling position embedding unit is introduced to the DPTRN. Fault diagnosis and fault detection experiments were carried out on the TE, KDD-CUP99, PEMFC, and WHELL datasets. The experimental results show that the proposed DPTRN model has obvious superiority in both accuracy and efficiency compared with the traditional neural network fault diagnosis model. In addition, the output of the time relation unit and the decoupling position embedding unit is analyzed in detail, and the results show the reliability and stronger interpretability of the DPTRN.

It is worth discussing why our proposed DPTRN does not achieve the best results on the WHELL dataset. We think that there are two reasons. First, as seen from Table IX, the FDGRU has the most parameters, so it has a better data fitting ability than the DPTRN. Second, our proposed DPTRN relies on the feature interaction between different time nodes for feature extraction, and the feature dimension of WHELL data is small, with two dimensions, which reduces the feature extraction performance of the DPTRN. Nevertheless, our proposed DPTRN still achieves more than 99% on the three metrics, 2.77% higher than 1DCNN-VAF and 0.4% higher than MCNN-LSTM in the average recall.

When dealing with time-series data, this article refocuses attention on MLP and proves the feasibility and effectiveness of parallel computing. This provides a new idea for solving the problem of time-series data fault diagnosis.

In this article, only the original features of historical time nodes are used, while the label information of each historical node is ignored. Further research on how to fully use historical data will be an interesting direction. Furthermore, how to effectively interact features with smaller dimensions deserves further research.

REFERENCES

[1] D.-T. Hoang and H.-J. Kang, “A survey on deep learning based bearing fault diagnosis,” Neurocomputing, vol. 335, pp. 327–335, Mar. 2019.
[2] D. Cabrera et al., “Bayesian approach and time series dimensionality reduction to LSTM-based model-building for fault diagnosis of a reciprocating compressor,” Neurocomputing, vol. 380, pp. 51–66, Mar. 2020.
[3] S. Zhang, K. Bi, and T. Qiu, “Bidirectional recurrent neural network-based chemical process fault diagnosis,” Ind. Eng. Chem. Res., vol. 59, no. 2, pp. 824–834, Jan. 2020.
[4] D. Lavrova, D. Zegzhda, and A. Yarmak, “Using GRU neural network for cyber-attack detection in automated process control systems,” in Proc. IEEE Int. Black Sea Conf. Commun. Netw. (BlackSeaCom), Jun. 2019, pp. 1–3.
[5] A. Zollanvari, K. Kunanbayev, S. Akhavan Bitaghfsir, and M. Bagheri, “Transformer fault prognosis using deep recurrent neural network over vibration signals,” IEEE Trans. Instrum. Meas., vol. 70, pp. 1–11, 2021.
[6] M. Canizo, I. Triguero, A. Conde, and E. Onieva, “Multi-head CNN-RNN for multi-time series anomaly detection: An industrial case study,” Neurocomputing, vol. 363, pp. 246–260, Oct. 2019.
[7] J. Zhu, Q. Jiang, Y. Shen, C. Qian, F. Xu, and Q. Zhu, “Application of recurrent neural network to mechanical fault diagnosis: A review,” J. Mech. Sci. Technol., vol. 36, pp. 527–542, Feb. 2022.
[8] X. Gu, Z. Hou, and J. Cai, “Data-based flooding fault diagnosis of proton exchange membrane fuel cell systems using LSTM networks,” Energy AI, vol. 4, Jun. 2021, Art. no. 100086.
[9] H. Wang, W. Sun, L. He, and J. Zhou, “Intelligent fault diagnosis method for gear transmission systems based on improved multi-scale reverse dispersion entropy and swarm decomposition,” IEEE Trans. Instrum. Meas., vol. 71, pp. 1–13, 2022.
[10] T. Huang, Q. Zhang, X. Tang, S. Zhao, and X. Lu, “A novel fault diagnosis method based on CNN and LSTM and its application in fault diagnosis for complex systems,” Artif. Intell. Rev., vol. 55, no. 2, pp. 1289–1315, Feb. 2022.
[11] J. Shi, D. Peng, Z. Peng, Z. Zhang, K. Goebel, and D. Wu, “Planetary gearbox fault diagnosis using bidirectional-convolutional LSTM networks,” Mech. Syst. Signal Process., vol. 162, Jan. 2022, Art. no. 107996.
[12] Z. Chen, K. Liang, S. X. Ding, C. Yang, T. Peng, and X. Yuan, “A comparative study of deep neural network-aided canonical correlation analysis-based process monitoring and fault detection methods,” IEEE Trans. Neural Netw. Learn. Syst., vol. 33, no. 11, pp. 6158–6172, Nov. 2022.
[13] X. Nie and G. Xie, “A two-stage semi-supervised learning framework for fault diagnosis of rotating machinery,” IEEE Trans. Instrum. Meas., vol. 70, pp. 1–12, 2021.
[14] Y. Wang, D. Zhong, and R. Jia, “Fault diagnosis method for MMC-HVDC based on bi-GRU neural network,” Energies, vol. 15, no. 3, p. 994, Jan. 2022.
[15] H. Wang, Z. Liu, D. Peng, M. Yang, and Y. Qin, “Feature-level attention-guided multitask CNN for fault diagnosis and working conditions identification of rolling bearing,” IEEE Trans. Neural Netw. Learn. Syst., vol. 33, no. 9, pp. 4757–4769, Sep. 2022.
[16] G. Xu, M. Liu, Z. Jiang, W. Shen, and C. Huang, “Online fault diagnosis method based on transfer convolutional neural networks,” IEEE Trans. Instrum. Meas., vol. 69, no. 2, pp. 509–520, Feb. 2020.
[17] D. Huang, S. Li, N. Qin, and Y. Zhang, “Fault diagnosis of high-speed train bogie based on the improved-CCEEMDAN and 1-D CNN algorithms,” IEEE Trans. Instrum. Meas., vol. 70, pp. 1–11, 2021.

[18] X. Song, Y. Cong, Y. Song, Y. Chen, and P. Liang, “A bearing fault diagnosis model based on CNN with wide convolution kernels,” J. Ambient Intell. Humanized Comput., vol. 13, no. 8, pp. 4041–4056, Aug. 2022.

[19] J. Ni, C. Zhang, and S. X. Yang, “An adaptive approach based on KPCA and SVM for real-time fault diagnosis of HVCBs,” IEEE Trans. Power Del., vol. 26, no. 3, pp. 1960–1971, Jul. 2011.

[20] P. K. Wong, Z. Yang, C. M. Vong, and J. Zhong, “Real-time fault diagnosis for gas turbine generator systems using extreme learning machine,” Neurocomputing, vol. 128, pp. 249–257, Mar. 2014.

[21] H. Pourrahmani et al., “Progress in the proton exchange membrane fuel cells (PEMFCs) water/thermal management: From theory to the current challenges and real-time fault diagnosis methods,” Energy, vol. 1, no. 1, Sep. 2022, Art. no. 100002.

[22] X. Miao, S. Li, Y. Zhu, and Z. An, “A novel real-time fault diagnosis method for planetary gearbox using transferable hidden layer,” IEEE Sensors J., vol. 20, no. 15, pp. 8403–8412, Aug. 2020.

[23] S. Langarica, C. Ruffellmacher, and F. Nunez, “An industrial internet application for real-time fault diagnosis in industrial motors,” IEEE Trans. Autom. Sci. Eng., vol. 17, no. 1, pp. 284–295, Jan. 2020.

[24] D. Zhou, Q. Yao, H. Wu, S. Ma, and H. Zhang, “Fault diagnosis of gas turbine based on partly interpretable convolutional neural networks,” Energy, vol. 200, Jun. 2020, Art. no. 117467.

[25] A. Ragab, M. El-Koujok, B. Poulin, M. Amazouz, and S. Yacout, “Fault diagnosis in industrial chemical processes using interpretable patterns based on logical analysis of data,” Expert Syst. Appl., vol. 95, pp. 368–383, Apr. 2018.

[26] F. B. Abid, M. Sallenn, and A. Braham, “Robust interpretable deep learning for intelligent fault diagnosis of induction motors,” IEEE Trans. Instrum. Meas., vol. 69, no. 6, pp. 3506–3515, Jun. 2020.

[27] D. Zhang, Y. Chen, F. Guo, H. R. Karimi, H. Dong, and Q. Xuan, “A new interpretable learning method for fault diagnosis of rolling bearings,” IEEE Trans. Instrum. Meas., vol. 70, pp. 1–10, 2021.

[28] H. Liu, Z. Dai, D. R. So, and Q. V. Le, “Pay attention to MLPs,” 2021, arXiv:2105.08050.

[29] I. Tolstikhin et al., “MLP-mixer: An all-MLP architecture for vision,” 2021, arXiv:2105.01601.

[30] J. Lee-Thorp, J. Ainslie, I. Eckstein, and S. Ontanon, “FNet: Mixing tokens with Fourier transforms,” 2021, arXiv:2105.03824.

[31] A. Vaswani et al., “Attention is all you need,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 5998–6008.

[32] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in Proc. 32nd Int. Conf. Mach. Learn., 2015, pp. 448–456.

[33] J. J. Downs and E. F. Vogel, “A plant-wide industrial process control problem,” Comput. Chem. Eng., vol. 17, no. 3, pp. 245–255, Mar. 1993.

[34] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, “The 1999 DARPA off-line intrusion detection evaluation,” Comput. Netw., vol. 34, no. 4, pp. 579–595, 2000.

[35] Y. Zhang, T. Zhou, X. Huang, L. Cao, and Q. Zhou, “Fault diagnosis of rotating machinery based on recurrent neural networks,” Measurement, vol. 171, Feb. 2021, Art. no. 108774.

[36] X. Chen, B. Zhang, and D. Gao, “Bearing fault diagnosis based on multi-scale CNN and LSTM model,” J. Intell. Manuf., vol. 32, no. 4, pp. 971–987, Apr. 2021.

[37] X. Wang, D. Mao, and X. Li, “Bearing fault diagnosis based on vibro-acoustic data fusion and 1D-CNN network,” Measurement, vol. 173, Mar. 2021, Art. no. 108518.