ASME Journal of Mechanical Design
Online journal at:
https://asmedigitalcollection.asme.org/mechanicaldesign

Shuhuai Gu
School of Data Science & Engineering,
South China Normal University,
Shanwei, Guandong 516625, China
e-mail: 20218131010@m.scnu.edu.cn

Qi Xi
Assistant Professor
School of Data Science & Engineering,
South China Normal University,
Shanwei, Guandong 516625, China
e-mail: xiqi@m.scnu.edu.cn

Jing Wang[1]
Assistant Professor
School of Data Science & Engineering,
South China Normal University,
Shanwei, Guandong 516625, China
e-mail: wangjingscnu@m.scnu.edu.cn

Peizhen Qiu
School of Data Science & Engineering,
South China Normal University,
Shanwei, Guandong 516625, China
e-mail: 20218131022@m.scnu.edu.cn

Mian Li
Fellow ASME
Professor
University of Michigan-Shanghai Jiao Tong
University Joint Institute Shanghai Jiao Tong
University,
Shanghai 200240, China
e-mail: mianli@sjtu.edu.cn

TCN-GAWO: Genetic Algorithm Enhanced Weight Optimization for Temporal Convolutional Network

This article proposes a genetic algorithm (GA)-enhanced weight optimization method for temporal convolutional network (TCN-GAWO). TCN-GAWO combines the evolutionary process of the genetic algorithm with the gradient-based training and can achieve higher predication/fitting accuracy than traditional temporal convolutional network (TCN). Performances of TCN-GAWO are also more stable. In TCN-GAWO, multiple TCNs are generated with random initial weights first, then these TCNs are trained individually for given epochs, next the selection-crossover-mutation procedure is applied among TCNs to get the evolved offspring. Gradient-based training and selection-crossover-mutation are taken in turns until convergence. The TCN with the optimal performance is then selected. Performances of TCN-GAWO are thoroughly evaluated using realistic engineering data, including C-MAPSS dataset provided by NASA and jet engine lubrication oil dataset provided by airlines. Experimental results show that TCN-GAWO outperforms existing methods for both datasets, demonstrating the effectiveness and the wide range applicability of the proposed method in solving time series problems. [DOI: 10.1115/1.4064809]

Keywords: artificial intelligence, data-driven design, machine learning

1 Introduction

To solve the problem that the traditional convolutional neural network (CNN) cannot deal with time series data well, researchers began to improve the CNN and obtained the temporal convolutional network (TCN). TCN improves on CNN by incorporating dilated convolutions, which allow for the efficient processing of variable-length time series data. Since first proposed in 2016 [1], TCN has become an important research topic in the field of time series data processing. It not only improves the ability to model the time dependence but also reduces the number of parameters and improves the efficiency. TCN has been widely used in many fields such as time series prediction [2], time series classification [3], action segmentation [4], and so on. Although TCN has shown advantages in processing time series data, there still exit weight-related problems in the TCN model, such as overfitting, poor initialization, nonconvex loss surface, and vanishing gradient [5].

The weights of a neural network determine the ability of the network to learn patterns in data and play a key role in the accuracy and performance of the network. If the weights are not properly optimized, the network may fail to learn the underlying patterns in the data and may produce inaccurate predictions [6].

Therefore, the optimization of neural network weights is crucial for the performance and accuracy of the network. Commonly used gradient descent algorithms [7] update parameters by continuously moving in the direction of the negative gradient, gradually approaching the optimal solution, which can sometimes lead to getting stuck in local minima, resulting in suboptimal solutions [8]. Stochastic gradient descent [9] introduces randomness by using gradients from small batches of samples during each iteration, which can help escape local minima. Alternatively, mini-batch gradient descent [10] employs medium-sized batches of samples for parameter updates to avoid getting trapped in local optima. However, there are numerous local minima and saddle points within the loss function, especially in high-dimensional nonconvex optimization problems like deep learning. Gradient descent may trap the

[1]Corresponding author.

Contributed by the Design Automation Committee of ASME for publication in the Journal of Mechanical Design. Manuscript received May 6, 2023; final manuscript received February 8, 2024; published online March 5, 2024. Assoc. Editor: Conrad S. Tucker.

loss function in these local regions, hindering further optimization and resulting in poor generalization. To avoid local minima in the process of training neural networks, researchers have developed various momentum-based techniques, such as the basic momentum [11], Nesterov momentum [12], RMSprop [13], and adaptive moment estimation (Adam) [14]. Notably, Adam stands out as the most widely used method for training neural networks.

Genetic algorithm (GA) is an optimization algorithm that simulates the natural selection of Darwin's biological evolution theory and the biological evolution process of genetic mechanism [15]. It is suitable for many optimization problems and has the advantages of strong global search ability, high efficiency, parameter-free, analysis-free, and strong scalability. Using GA to optimize the weights of a neural network can be an alternative to traditional gradient-based optimization algorithms. Thus, it is plausible to contemplate coupling both gradient-based and GA optimization methods, harnessing the global search capabilities of GA to expedite the convergence of the gradient descent algorithms toward local optimal solutions.

In recent years, the use of GA to optimize the neural networks has attracted the attention of researchers. According to recent studies conducted by Lu [16], Yu and Zhang [17], Arhore et al. [18], and Luo et al. [19], GA has proven to be effective in optimizing the hyper-parameters and architecture of CNN, long short-term memory (LSTM), and deep neural network (DNN), a mathematical model designed to imitate human brains [19], resulting in higher accuracies of CNN, LSTM, and DNN models. In a comparative study conducted by Gonçalves et al. [20] between GA and particle swarm optimization as optimizers for CNN architecture, they found that the optimum CNN obtained through GA exhibited higher accuracy. In addition, studies conducted by Elsken et al. [21] and Tseng et al. [22] showed that GA was an effective tool for optimizing hyperparameters and neural network architecture.

As an improvement over CNN, TCN makes full use of convolution operations to extract information in the time dimension, thereby more accurately capturing the temporal relationships in time series data [1]. In a series of sequence modeling tasks, Bai et al. [23] compared TCN with traditional recursive architectures such as LSTM and gated recurrent unit (GRU). The results clearly show that compared to the baseline recursive architecture, TCN exhibits significant advantage, highlighting its excellent performance in sequence data processing. After the proposal of TCN, Zhang et al. [24] and Chen et al. [25] utilized GA to optimize the hyperparameters and structure of TCN and also found that the TCN obtained through GA demonstrated higher accuracy. These works focused on optimizing the hyperparameters of a TCN. Although choosing optimal hyperparameters for a TCN is important, once a neural network is complex enough to model a real application, the effect of optimizing the hyperparameters alone may not be that satisfactory. A better performance can be obtained by optimizing the weights of a neural network. Thus, this article proposes a novel approach by employing GA to optimize the weights of TCN model, fully using the diversity and global search capability of GA as well as the strong ability of TCN in handling sequence data.

This study proposes a GA-enhanced weight optimization method for TCN (TCN-GAWO). TCN-GAWO employs GA to optimize weights during TCN's gradient-based training process. Furthermore, GAWO can also be applied to other gradient-based neural networks. However, TCN possesses an inherent advantage in handling time series data and continues to demonstrate exceptional performance even with limited training data. In current industrial domains such as aerospace [26] and wind power [27], TCN finds extensive application. Therefore, this study selects TCN as the target for weight optimization based on GA.

The overall architecture of TCN-GAWO is illustrated in Fig. 1. First, data are preprocessed and used as input for TCN-GAWO. What is more, multiple TCNs are generated as individuals with randomly initialized weights (represented by the circles in the training process diagram of Fig. 1), and their weights are extracted to form their chromosomes. Subsequently, these TCNs are individually trained for a specified number of epochs (as indicated by the arrows in the training process diagram of Fig. 1, each TCN individual is trained for m epochs). Then, a selection-crossover-mutation process is applied among these TCN individuals (as depicted by the rectangles in the training process diagram of Fig. 1), generating a new generation of the population. The new population continues to undergo training and the selection-crossover-mutation process

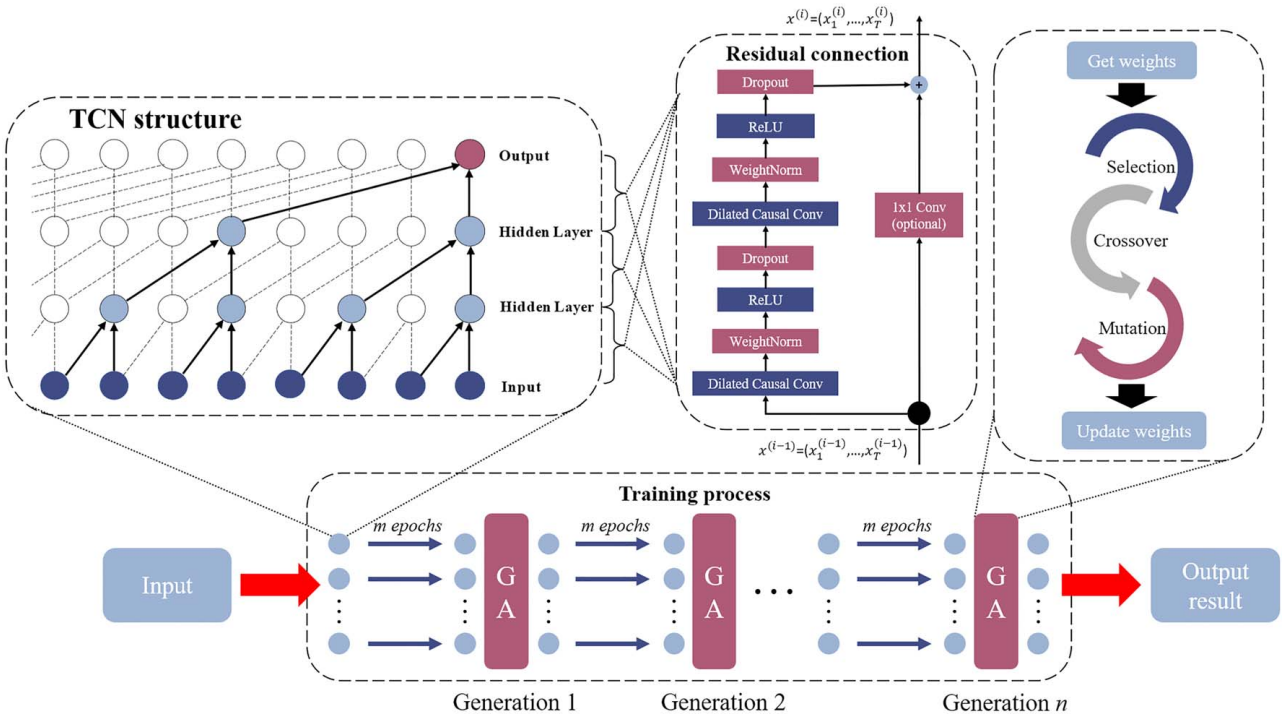

Fig. 1 The overall architecture of TCN-GAWO

to produce the next generation. This process is repeated, and when generating the nth generation of the population, the individual with the highest fitness is selected as the output result. This approach combines the advantages of gradient-based training and GA, addressing overfitting issues commonly encountered in traditional TCNs while being capable of handling large-scale time series datasets.

The novelty of the TCN-GAWO algorithm lies in its successful integration of traditional GA with gradient-based training methods to optimize the training process of TCN, thereby enhancing search efficiency. Traditional genetic algorithms often struggle to find optimal solutions in high-dimensional spaces, but the introduction of gradient information makes the GA search more precise and efficient. This hybrid approach facilitates faster convergence to optimal solutions. Furthermore, TCN-GAWO strikes a balance between diversity and convergence. By incorporating gradient-based training steps within each individual, the algorithm enables information exchange among different individuals and introduces new variations when necessary, thus balancing the tradeoff between diversity and convergence.

The proposed method is first applied to the C-MAPSS dataset [28] provided by NASA, and the results show that TCN-GAWO outperforms traditional TCN in both prediction accuracy and fitting.

To further validate the effectiveness of the TCN-GAWO in real engineering applications, the proposed method is applied to obtain a baseline model of jet engine lubrication oil quantity during the flight. Lubrication oil quantity is critical for aircraft engine and should be carefully monitored [29]. Because of the complexity of engine lubrication system, it is difficult to establish an analytical model for lubrication oil quantity. Moreover, the model of lubrication oil quantity is different for engines even of the same type due to batch-to-batch variations in manufacturing. Thus, in this article, the proposed method are used to generate models of lubrication oil quantity for each engine using realistic aircraft recoding data during flights. Model inputs consists of 11 parameters including engine rotor speed, aircraft speed/altitude/attitude, etc. Experiment results show that the model obtained by TCN-GAWO can effectively describe the variation of lubrication oil quantity during flights (phase of descend), and the accuracy is higher than the traditional neural networks including TCN, LSTM, and GRU, demonstrating the applicability of the proposed method.

The rest of the article is organized as follows. Section 2 introduces the background knowledge used in our proposed method. Section 3 presents our proposed method. Section 4 presents a case study for the NASA C-MAPSS dataset. Section 5 presents a real application of airplane engine lubrication oil dataset. Section 6 concludes the article.

2 Knowledge

2.1 Genetic Algorithm. GA is an optimization algorithm based on the principles of biological evolution [30]. It simulates the processes of genetic inheritance, natural selection, and survival of the fittest found in nature and is used to tackle a wide range of complex problems. GA has been successfully applied in various fields, including function optimization, machine learning, and artificial intelligence. They have proven to be efficient and effective in solving problems that are challenging for traditional methods. The following is an explanation of the main steps of GA:

(1) *Population Initialization:* Generate a random set of potential solutions represented as individuals with chromosomes. Population size and encoding depend on the problem.
(2) *Fitness Function:* The fitness function is crucial for assessing solution quality. It is tailored to the problem, quantifying how well a solution meets optimization objectives. For instance, in path optimization, shorter paths receive higher fitness scores.
(3) *Selection:* To simulate natural selection, GA uses selection operators like tournament or roulette wheel selection. Solutions with higher fitness have a better chance of becoming parents, preserving and improving the best solutions.
(4) *Crossover:* In this step, selected parents exchange genes to create offspring. Crossover introduces new solutions and diversity into the population. For path problems, sequence crossover can generate new paths.
(5) *Mutation:* Genetic mutation is mimicked by randomly altering genes in individuals to diversify the population. Low mutation rates prevent disruption of better solutions. In path problems, mutation might involve changing one or more nodes in a path.
(6) *End Condition:* After each generation of operations, it is necessary to evaluate whether the end condition has been met. If the end condition is met, the algorithm stops and outputs the optimal solution. Otherwise, it returns to step 3 to continue evolving the next generation.

2.2 Temporal Convolutional Network. TCN is a convolutional neural network-based model designed specifically for processing time series data. It excels at capturing intricate patterns within time series data due to its utilization of convolutional layers, making it particularly adept at handling long-term dependencies and sequences of variable lengths. The key architectural components of a TCN can be classified into three main categories: causal convolutions for sequence modeling, dilated convolutions for capturing historical data patterns, and residual connections for improved information flow [23]. The structural overview of TCN is illustrated in Fig. 1.

2.2.1 Causal Convolutions. Causal convolution layers are a specialized form of convolutional layers that exclusively consider the current and past input information, deliberately excluding any knowledge of future input data. This design choice aligns TCN with practical scenarios, as it avoids using information from the future to predict the current time point in time series prediction tasks.

2.2.2 Dilated Convolutions. Dilated convolutions are a unique convolutional operation that enhances network depth by increasing the spacing between the convolution kernel and the input, as opposed to adding more network layers. This technique facilitates a more effective capture of long-range correlations within the data and extends network depth without the need for an excessive number of layers. The dilated convolution operation, denoted as F, at element t in the sequence is defined as follows:

$$F(t) = (X *_d f)(t) = \sum_{i=1}^{k} f(i) \cdot x_{t-d \cdot i} \tag{1}$$

where $X = (x_1, x_2, \ldots, x_k)$ is a 1D sequence, $F = (f_1, f_2, \ldots, f_k)$ is a filter, d is the dilation factor, and $t - d \cdot i$ accounts for the direction of the past.

2.2.3 Residual Connections. Residual connections, a technique originating from deep neural networks, are employed in TCN to enhance information flow and gradient propagation throughout the network. The fundamental concept behind residual connections is to create direct connections between the input and output of each layer, thereby bypassing intermediate transformations and non-linear operations. This enables the network to learn an identity mapping or a shortcut through the network, rather than requiring the network to learn a more complex function. This approach is particularly beneficial in addressing the vanishing gradient problem, which commonly occurs in deep networks when gradients become too small to effectively train deeper layers.

In the TCN, the simple connections between the layers are replaced by the residual structures as shown in Fig. 1. At the same time, x is transformed by 1×1Conv to solve the problem that x and $F(x)$ cannot be added directly due to different channel numbers.

2.3 Baseline Methods. To further validate the effectiveness of the proposed model, in addition to the TCN model without GA optimization, additional baseline methods are applied in the case study and real application as comparisons.

The transformer [31] has achieved significant success in sequence modeling architectures, demonstrating unparalleled performance in various applications. Transformer-based models for time series analysis have also seen a surge in popularity. However, Zeng et al. [32] have provided a different perspective, demonstrating why transformer may not be as effective as claimed in long-term time series forecasting works. Therefore, since this study focuses on dealing with time series data, transformer-based models are not employed as baseline methods.

Building upon the aforementioned discussions, this study has chosen two dedicated models for handling time series data as the baseline methods. The results provide a comprehensive assessment of the model's predictive performance.

The first method is long short-term memory. It is a type of recurrent neural network (RNN) architecture that is capable of capturing long-term dependencies in sequential data by incorporating memory cells and gates to control information flow [33].

The second method is GRU. It is a type of RNN architecture that effectively balances the tradeoff between computational efficiency and model capacity through the use of gating mechanisms to regulate the information flow [34].

3 Proposed Method

TCN-GAWO is divided into three parts, i.e., model initialization, GA optimization, and output, as shown in Fig. 2.

3.1 Model Initialization. First, the dataset is imported and then preprocessed by normalization. The normalization is defined as follows:

$$x_{\text{norm}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{2}$$

where x_{norm} is the normalized data, x is raw sensor data, and $x_{\max}$ and $x_{\min}$ are the maximum and minimum values of x, respectively.

Then after determining the structure and hyperparameters of TCNs, a specified number of TCN models are generated as the initial population of GA and the weights of TCN model are initialized using the Glorot uniform initialization method [35]. Generating a specified number of TCN models at random ensures genetic diversity among individuals in GA, enabling initial individuals to be distributed as widely as possible within the solution space to avoid premature convergence to local optima. The utilization of the Glorot uniform initialization method ensures that the TCN models have a balanced scaling of the weights during the initialization process, leading to improved performance in terms of convergence and generalization. The weights of each TCN model are interpreted as the chromosome of the respective individual, as shown in Eq. (3).

$$\text{chromosome} = [w_1, w_2, w_3 \ldots w_n] \tag{3}$$

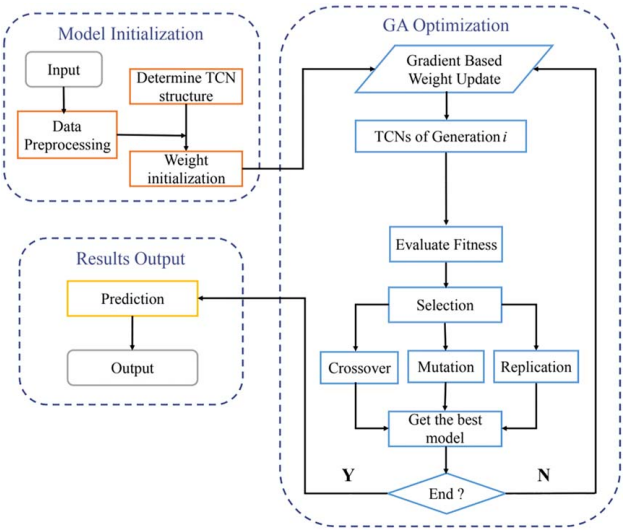

Fig. 2 The process diagram of TCN-GAWO

3.2 Genetic Algorithm Optimization. After completing the model initialization, each TCN individual undergoes weight updates through gradient-based training for a certain number of epochs in an attempt to learn patterns and features within the time series data. During this phase, GA does not directly intervene in the training process but rather awaits the individuals to finish their independent training. Once all TCN individuals have completed their training, GA intervenes to assess the fitness of all individuals within the population and performs operations such as selection, crossover, and mutation.

In the training phase prior to a selection-crossover-mutation process, N TCN model undergoes independent training using the same input data. However, during a selection–crossover-mutation process of the GA, TCN models are interrelated.

This study primarily employed classical GA [30] to optimize the training process of TCN. The main steps of the optimization process of the GA are as follows.

(1) *Fitness Evaluation*: In the initial phase of population generation, the first step is to calculate the fitness of all individuals. The fitness function for individuals is defined as the inverse of the root-mean-squared error (RMSE) of the TCN model, as shown in Eq. (4).

$$\text{fitness} = \frac{n}{\sqrt{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}} \tag{4}$$

where y_i is the true value, $\hat{y}_i$ is the predicted value, and the n is the number of testing samples.

(2) *Selection*: Utilizing a roulette wheel selection method [36] to pick individuals with higher fitness values. Depending on the set crossover and mutation rates, it is determined whether the selected individuals should undergo crossover or mutation operations. If not required, they are directly copied and added to the next generation population. This approach allows for the selection of individuals with higher adaptability, transmitting their traits to the next generation and improving the overall performance of the population over time.

(3) *Crossover*: The production of superior individuals is achieved through the exchange of chromosomes between two parents. Two parent individuals with relatively higher fitness are randomly selected from the current population. A crossover point is chosen within the chromosomes of these two parent individuals. They are then split at the crossover point, and their genetic segments are exchanged to generate new offspring. This process allows for the combination of advantageous characteristics from both parents, resulting in the generation of new individuals. The specific steps are illustrated in Eqs. (5)–(7).

$$\text{parent}_1 = \left[w_1^1, w_2^1 \ldots, w_{p-1}^1, w_p^1, w_{p+1}^1 \ldots w_n^1\right] \tag{5}$$

$$\text{parent}_2 = \left[w_1^2, w_2^2 \ldots, w_{p-1}^2, w_p^2, w_{p+1}^2 \ldots w_n^2\right] \tag{6}$$

$$\text{offspring} = \left[w_1^1, w_2^1 \ldots, w_{p-1}^1, w_p^2, w_{p+1}^2 \ldots w_n^2\right] \tag{7}$$

where parent_1 and parent_2 are the chromosomes of the parents, offspring is the chromosome of the offspring after crossing, p is a randomly selected point, and $p \in (1, n)$.

(4) *Mutation*: The process of mutation is employed to enhance the diversity and quality of the population. This study randomly selects a portion of genes from the chromosome. As shown in Eq. (8), if w_i is chosen, it will be subtracted by a random value, where the magnitude of this random value is determined by $0.1 \times \sigma \cdot N(0, 1)$. If w_i is not chosen, it remains unchanged.

$$w_i' = \begin{cases} w_i - w_i \times 0.1 \times \sigma \cdot N(0, 1), & \text{chosen} \\ w_i, & \text{notchosen} \end{cases} \tag{8}$$

(5) *The Stopping Criterion*: The stopping criterion can be a realistic engineering requirement or a target value. After each round of training and iteration, the model is evaluated, and the iteration should stop if the stopping criterion is satisfied.

By integrating the iterative of GA with the training process of TCN networks, a new generation of offspring is first trained for a certain amount of epochs, followed by evaluations, selections, crossovers, and mutations using GA. These operations are repeated until the stop criterion is achieved, resulting in an iterative process optimizing the population's performance, as shown in Fig. 2.

Algorithm 1 TCN-GAWO

Require: The number of epochs, population number, end conditions
Ensure: Best individual (TCN model with optimal weight configuration)
$t = 1$
$S^1 = \{\}$
Population number $= N$
For $i = 1, 2, ..., N$ **do**
  Initial TCN_i^t with random weights using Glorot uniform initialization method
  Perform gradient-based training on TCN_i^t for a number of epochs.
  Evaluate fitness of TCN_i^t
  Add TCN_i^t to the S^1
end for
While End conditions do not met **do**
  $t = t + 1$
  $S^t = \{\}$
  Select parent from S^{t-1} using the roulette wheel method
  Generate offspring via crossover, mutation, copy
  For $i = 1, 2, ..., N$ **do**
    Perform gradient-based training on TCN_i^t for a number of epochs.
    Evaluate fitness of TCN_i^t
    Add TCN_i^t to the S^t
  end for
  Select the best individual from S^t
end while

3.3 Results Output. After training and optimization, the model weights are updated, and the resulting models are used for simulation and prediction.

Finally, the evaluation of the model is carried out using the mean absolute error (MAE), mean-square error (MSE), and root-mean-square error (RMSE), as shown in Eqs. (9)–(11).

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right| \tag{9}$$

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2 \tag{10}$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2} \tag{11}$$

where y_i is the true value, $\hat{y}_i$ is the predicted value, and the n is the number of testing samples.

The proposed method is concluded in Algorithm 1.

4 Case Study

4.1 Description of Dataset. This case study utilizes the C-MAPSS dataset of aircraft turbofan engines provided by NASA. The data are composed of four parts: FD001, FD002, FD003, and FD004. Among them, FD001 includes complete records of 100 turbofan engines from start to failure under a single operating condition and a single failure mode. The other three datasets contain data on multiple operating conditions and failure modes. What is more, FD001 is a typical case in the C-MAPSS dataset, widely accepted as a standard for evaluating and comparing the performance of different algorithms. It is a relatively small dataset, making it relatively simple and economical for computational and experimental purposes. This makes the FD001 dataset more accessible and applicable in academic research, as it can be used as a benchmark case in the early stages of research or algorithm validation. So in this case study, the FD001 dataset is used.

The FD001 dataset is composed of three files: training set, test set, and RUL. The training set contains complete records of 100 engines from their start of operation until failure, measured in terms of operating cycles and remaining useful life. The set contains 20,631 records, each with the engine's identification number, current cycle count, three engine settings, and 21 sensor values. The test set contains a subset of randomly selected records from 100 engines, which do not represent complete records from start to failure, comprising of 13,096 records. The RUL file contains the true remaining useful life of the last recorded cycle for each of the 100 engines in the test set.

In order to highlight the advantages of our proposed model, similar to many projects using C-MAPSS as a dataset, preliminary feature analysis is conducted before utilizing the measurements from these sensors. This analysis aimed to eliminate constant values and the least time-varying features to mitigate their impact on the target values.

For longer time series data, sliding windows are often used to segment the data. In order to capture the time correlation among data, the sliding window is used to encapsulate the data at adjacent time points. The RUL of the aircraft engines in the study is established by considering the RUL value of the final data point within the defined time window. Extensive experiments have been conducted to select suitable model structure and hyperparameters. The article determines the appropriate time window size to be 10 and the step size to be 1. A TCN neural network with one TCN layer and one fully connected layer is adopted. The optimal combination of TCN is determined to be filters = 32, kernelsize = 2, batchsize = 16, and dilations = [1, 2, 4, 8]. The optimal combination of GA is determined to be initial crossoverrate = 0.5 and mutationrate = 0.1. The decision was made to terminate training at 40 epochs, as the TCN-GAWO had achieved the desired level of accuracy on the C-MAPSS dataset at this point. Training 4 epochs per iteration of the TCN-GAWO.

In addition to the metrics discussed in Sec. 3.3, this study considers an application-specific scoring function to compare the various model performances. This scoring function was proposed as an evaluation indicator specifically tailored to C-MAPSS dataset by Saxena et al. [28] in the 2008 PHM data challenge competition. This score function has since gained significant prominence in the field of RUL prediction, as shown in Eq. (12).

$$\text{Score} = \begin{cases} \sum_{i=1}^{n} e^{-(d/a_1)} \text{ if } d < 0 \\ \sum_{i=1}^{n} e^{-(d/a_2)} \text{ if } d \geq 0 \end{cases} \tag{12}$$

where s is the computed score, n is the number of testing samples, $d = \hat{t}_{RUL} - t_{RUL}$ (i.e., Estimated RUL – True RUL), $a_1 = 10$, and $a_2 = 13$.

Table 1 Comparing score across various population sizes in Sec. 4

Population size	Score
10	330.52
15	300.76
20	287.34
25	256.04
30	249.35
35	245.32

This study conducted ten experiments and averaged the results to compare different population sizes and their corresponding score values in the C-MAPSS dataset, as shown in Table 1. It can be observed that when the population size is 25, the score value is 256.04, which is significantly smaller than when the population sizes are smaller than 25. Furthermore, as the population size continues to increase, the reduction in score value is not substantial. Therefore, a population size of 25 is chosen.

4.2 Application Results. In the experiments, this study ensures consistent parameter settings for each network structure, including the same number of units, learning rate, batch size, activation function, and an equal number of training epochs for all models. In addition, for LSTM, GRU, and TCN, this study employs gradient descent algorithms, while for TCN-GAWO, this study utilizes a weight optimization algorithm that combines genetic algorithms with gradient-based training. Notably, the number of trainable parameters for LSTM stands at 6049, while GRU has 4641 trainable parameters, and TCN boasts 16,513 trainable parameters. This approach allowed for a fair and standardized comparison among the different models.

During the training process, the training dataset is divided into a training set and a validation set, with a ratio of 8:2. In order to gain a more comprehensive understanding of model's performance under different data distributions, thereby assessing its reliability and stability more accurately, the overall performance of the model is evaluated on the training set, validation set, and testing set, as shown in Figs. 3–5.

In the test set, this study conducted ten experiments and averaged the results to determine that TCN required a training time of 367.1 s, while TCN-GAWO demanded a training time of 9358.3 s. Four evaluation metrics MAE, MSE, RMSE, and score are obtained through four different models of LSTM, GRU, TCN, and TCN-GAWO. The results of four models are compared in Table 2. In the evaluation index statistics, the proposed TCN-GAWO model has the best prediction results, with lower RMSE and prediction scores on the C-MAPSS dataset. Figure 6 shows the comparison between the predicted results of the TCN, TCN-GAWO and the true results. Figure 5 illustrates the change in RMSE of TCN as the number of epochs increases during training. It can be observed that when the number of epochs exceeds 10, the RMSE values fluctuate between 14 and 18. As indicated by the RMSE values of TCN-GAWO in Fig. 5, after optimization by GA, the model can break through the local optimization so that the RMSE value is less than 14.

It can be seen that the proposed TCN-GAWO model significantly outperforms the other three models and has a better prediction effect

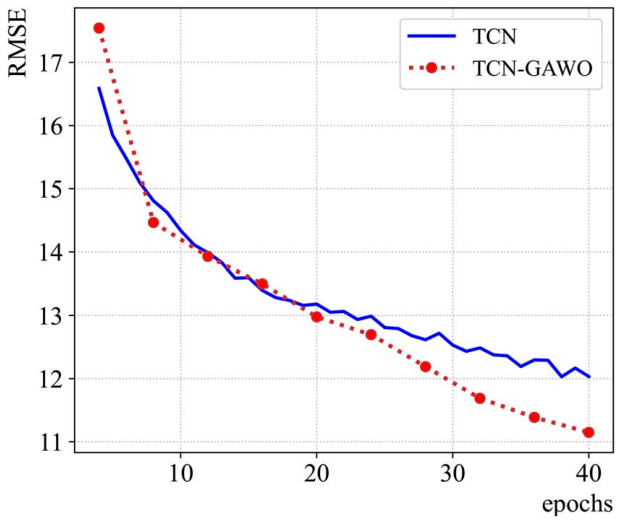

Fig. 3 Training loss of TCN and TCN-GAWO in Sec. 4

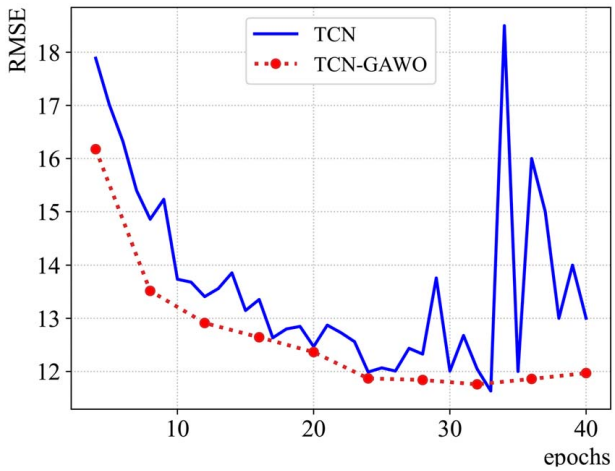

Fig. 4 Validation loss of TCN and TCN-GAWO in Sec. 4

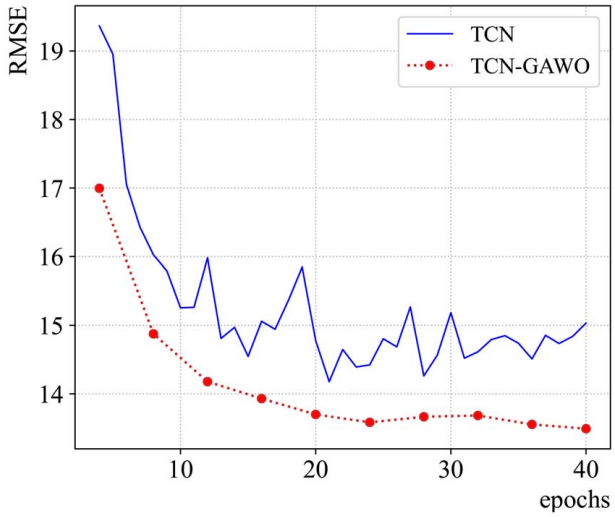

Fig. 5 Test loss of TCN and TCN-GAWO in Sec. 4

Table 2 Comparison of prediction results of four different models in Sec. 4

Mode	MAE	MSE	RMSE	Score
LSTM	13.23	274.97	16.58	541.26
GRU	12.46	257.53	16.05	480.85
TCN	11.82	231.16	15.20	355.14
TCN-GAWO	10.41	187.62	13.68	256.04

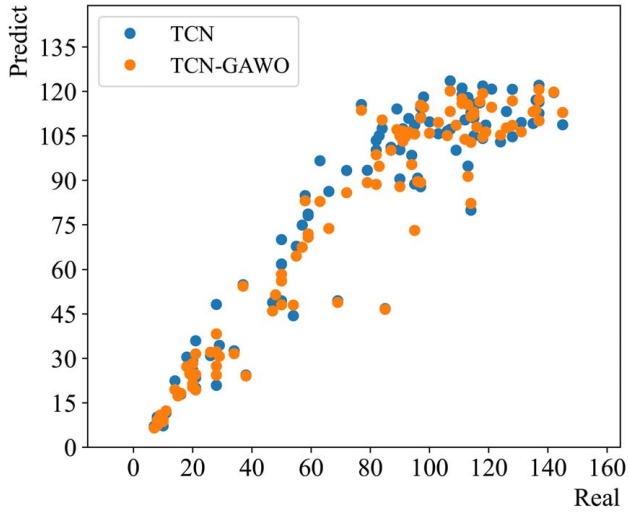

Fig. 6 The true RUL and the predicted RUL by TCN and TCN-GAWO in Sec. 4

on the C-MAPSS dataset, which indicates the advantages of the proposed framework.

To further explore the computational cost of our proposed method, comparisons are conducted for different models with the same computational resources. For fair comparisons, since 25 TCN models are chosen as individuals of a GA generation with each TCN model trained for 4 epochs and 10 GA generations, a cumulative computational budget is 1000 training epochs. Therefore, in the first comparison, 1000 epochs is employed for a single TCN model, with the results shown in the first line of Table 3. The second comparison is to train 25 randomly initialized TCN models separately for 40 epochs each, and select the best performing TCN model (with the results shown in the second line of Table 3) to compare with our proposed TCN-GAWO model. The result of our TCN-GAWO model is shown in the last line of Table 3. From these results, it can be seen that TCN-GAWO achieves the best performance in all the indices.

5 Real Application

5.1 Background. In aircraft engine failures, oil system failures account for a significant proportion. However, the mechanism of oil quantity changes during flight is very complex and is influenced by various factors such as engine manufacturing batch and flight conditions. It is difficult to directly establish a physical model. Currently, in practical applications, oil system failure monitoring is mostly based on empirical and qualitative analysis, which is not precise enough and lacks quantification. Therefore, this article applies the proposed method to input the key factors influencing oil quantity changes and trained a TCN-GAWO to obtain a predicted oil quantity.

5.2 Description of Dataset. In different phases of flight, the inherent characteristics of changes in lubrication oil consumption vary due to differences in the engine's working conditions.

Table 3 Index values under different scenarios for case study in Sec. 4

Model	MAE	MSE	RMSE	Score
Single TCN(1000 epochs)	12.52	269.03	16.40	549.69
Best TCN(40 epochs)	11.34	209.92	14.49	347.99
TCN-GAWO	10.41	187.62	13.68	256.04

Table 4 Input parameters

Serial number	Name	Significance
1	N2	High-pressure rotor speed
2	ROLL	Rolling corner
3	d_ROLL	Roll angle difference
4	dd_ROLL	Roll angle second difference
5	PITCH	Pitch angle
6	d_PITCH	Pitch angle difference
7	dd_PITCH	Pitch angle second difference
8	GS	Ground speed
9	d_GS	Ground speed acceleration
10	ALT	Barometric height
11	d_ALT	Barometric height difference value

Therefore, it is necessary to model each flight phase separately. This article selects the descent phase with significant fluctuations in lubrication oil consumption for modeling. The model first performs temperature calibration on the lubrication oil consumption data to remove the influence of thermal expansion and contraction factors. Then, based on engineering expertise, this study has chosen 11 potential factors (as detailed in Table 4) that may affect the quantity of engine lubricating oil as inputs for model. Furthermore, to elevate the quality of our dataset and bolster signal processing, this study employs wavelet decomposition techniques, effectively mitigating noise and compressing the dataset. Finally, TCN-GAWO is used to train the data model to obtain the lubrication oil consumption. Extensive experiments have been conducted to select suitable hyperparameters. The optimal combination of TCN is determined to be filters = 64, kernelsize = 2, batchsize = 64, and dilations = [1, 2, 4, 8, 16]. The optimal combination of GA is determined to be initial crossover rate of 0.5 and mutation rate of 0.1. The decision was made to terminate training at 40 epochs, as the TCN-GAWO had achieved the desired level of accuracy on the lubrication oil dataset at this point. Training 4 epochs per iteration of the TCN-GAWO.

This study conducted ten experiments and averaged the results to compared different population sizes and their corresponding RMSE values in the lubrication oil dataset, as shown in Table 5. It can be observed that when the population size is 20, the RMSE value is 0.316, which is significantly smaller than when the population sizes are 5, 10, and 15. Furthermore, as the population size continues to increase, the reduction in RMSE value is not substantial. Therefore, a population size of 20 is chosen.

5.3 Application Results. In the experiments, this study ensures consistent parameter settings for each network structure, including the same number of units, learning rate, batch size, activation function, and an equal number of training epochs for all models. In addition, for LSTM, GRU, and TCN, this study employs gradient descent algorithms, while for TCN-GAWO, this study utilizes a weight optimization algorithm that combines genetic algorithms with gradient-based training. What is more, the number of trainable parameters for LSTM stands at 19,099, while GRU has 14,465 trainable parameters, and TCN boasts 289,665 trainable parameters.

Table 5 Comparing RMSE across various population sizes in Sec. 5

Population size	RMSE
5	0.327
10	0.324
15	0.320
20	0.316
25	0.315
30	0.315

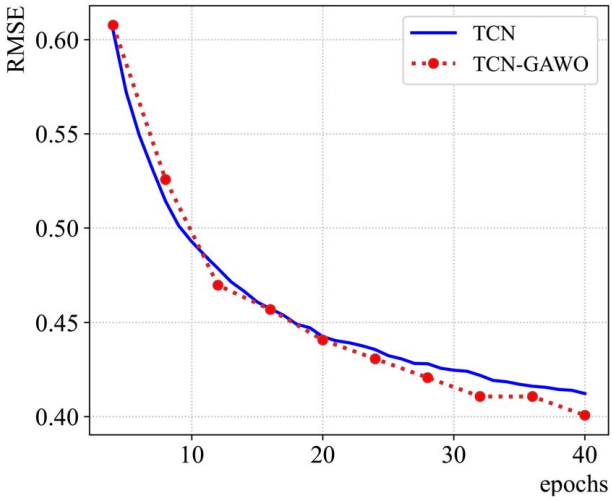

Fig. 7 Training loss of TCN and TCN-GAWO in Sec. 5

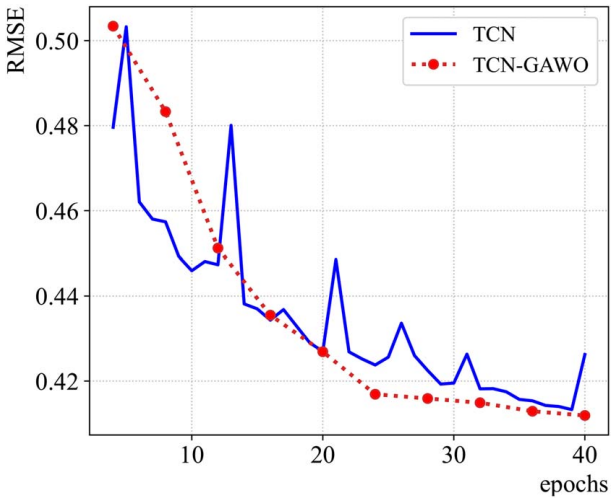

Fig. 8 Validation loss of TCN and TCN-GAWO in Sec. 5

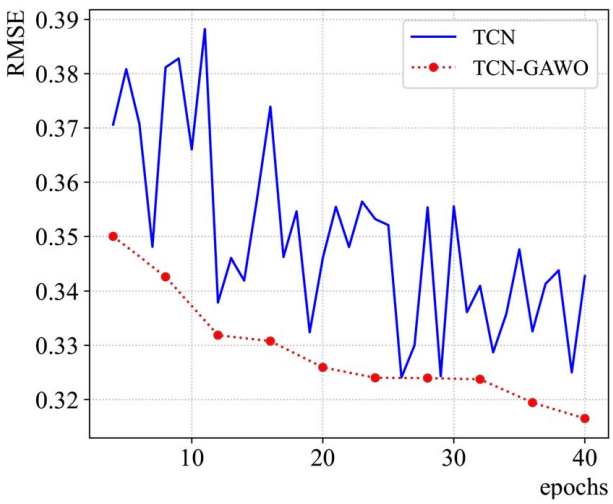

Fig. 9 Test loss of TCN and TCN-GAWO in Sec. 5

Table 6 Comparison of prediction results of four different models in Sec. 5

Mode	MAE	MSE	RMSE
LSTM	0.256	0.115	0.338
GRU	0.249	0.112	0.335
TCN	0.245	0.107	0.327
TCN-GAWO	0.233	0.100	0.316

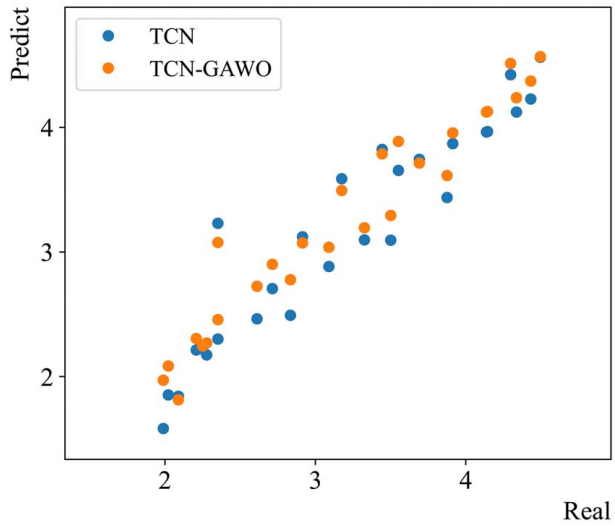

Fig. 10 The true result and the predicted result by TCN and TCN-GAWO in Sec. 5

During the training process, the training dataset is divided into a training set and a validation set in an 8:2 ratio. In order to gain a more comprehensive understanding of the model's performance under different data distributions, thereby assessing its reliability and stability more accurately, the overall performance of the model is evaluated on the training set, validation set, and testing set (as shown in Figs. 7–9). In the training and validation sets, as the number of training iterations increases, the loss values of TCN-GAWO gradually become smaller than those of TCN, and the loss values of TCN-GAWO tend to be more stable than TCN's.

In the test set, this study conducted ten experiments and averaged the results to determine that TCN required a training time of 1183.5 s, while TCN-GAWO demanded a training time of 29,374.6 s. The results of four models are compared using three performance metrics: MAE, MSE, and RMSE in Table 6. The proposed TCN-GAWO model demonstrates superior prediction performance in the evaluation metrics, with lower MAE, MSE, and RMSE on the same dataset. In Fig. 10, a comparison is made between TCN and TCN-GAWO, and it can be seen that the prediction performance of TCN is improved after GA optimization.Figure 9 shows how TCN's RMSE changes with an increase in the duration of the training period. It can be observed that when the number of epochs exceeds 5, the RMSE values fluctuate between 0.33 and 0.39. However, by breaking through local optimization through GA optimization, the RMSE decreases with an increase in epoch, leading to an RMSE value of less than 0.32, demonstrating its high feasibility for predicting the dataset.

These findings suggest that the TCN-GAWO model is a promising approach compared to the other three models (Table 6).

6 Conclusion

This article presents an improved method for optimizing the TCN neural network using GA. In the proposed model, the iterative

process of the GA is combined with the training process of the neural network to enhance the weight optimization. The GA-enhanced weight optimization enables the model to achieve stronger adaptability and higher accuracy in time series prediction tasks. The effectiveness of the proposed method is validated by using both the C-MAPSS dataset and a realistic engineering dataset of jet engine lubrication system. Results show that the proposed model outperforms existing baseline methods including TCN, LSTM, and GRU, demonstrating the applicability of the proposed method in solving time series problems in engineering applications.

TCN-GAWO can improve the prediction accuracy of a TCN by optimizing its weights, avoiding the local minima and overfitting. However, the shortcoming of TCN-GAWO is associated with its computational cost, which makes it less applicable in applications with limited computational resources. How to reduce the computational cost is in our future work. However, when tasks demand high-precision predictions with sufficient computational resources, TCN-GAWO can demonstrate its superior performance.

In summary, TCN-GAWO can improve the prediction accuracy of a TCN by optimizing its weights, avoiding the local minima and overfitting. However, the shortcoming of TCN-GAWO is associated with its computational cost, which makes it less applicable in applications with limited computational resources. How to reduce the computational cost is in our future work. However, when tasks demand high-precision predictions with sufficient computational resources, TCN-GAWO can demonstrate its superior performance.

Acknowledgment

The authors would like to thank the editor and reviewers for their time and suggestions. This work was supported by Shanwei City 2023 Provincial Science and Technology Innovation Strategy Special Project No.2023A011.

Conflict of Interest

There are no conflicts of interest.

Data Availability Statement

The datasets generated and supporting the findings of this article are obtainable from the corresponding author upon reasonable request.

References

[1] Lea, C., Vidal, R., Reiter, A., and Hager, G. D., 2016, "Temporal Convolutional Networks: A Unified Approach to Action Segmentation," Computer Vision—ECCV 2016 Workshops, Amsterdam, The Netherlands, Oct. 8–10 and 15–16, G. Hua and H. Jégou, eds., Springer International Publishing, pp. 47–54.

[2] Zeng, J., Zhao, Y., Li, G., Gao, Z., Li, Y., Barbat, S., and Hu, Z., 2024, "Vehicle Crashworthiness Performance Prediction Through Fusion of Multiple Data Sources," ASME J. Mech. Des., **146**(5), p. 051707.

[3] Alqahtani, S., Mishra, A., and Diab, M., 2019, "Efficient Convolutional Neural Networks for Diacritic Restoration," Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Hong Kong, China, Nov. 5–7, Association for Computational Linguistics, pp. 1442–1448.

[4] Zhang, Y., Ren, K., Zhang, C., and Yan, T., 2022, "SG-TCN: Semantic Guidance Temporal Convolutional Network for Action Segmentation," 2022 International Joint Conference on Neural Networks (IJCNN), Padua, Italy, July 18–23, pp. 1–8.

[5] Pascanu, R., Mikolov, T., and Bengio, Y., 2013, "On the Difficulty of Training Recurrent Neural Networks," Proceedings of the 30th International Conference on Machine Learning, Atlanta, GA, June 17–19, S. Dasgupta and D. McAllester, eds., PMLR, Vol. 28, pp. 1310–1318.

[6] Rumelhart, D. E., Hinton, G. E., and Williams, R. J., 1986, "Learning Representations by Back-Propagating Errors," Nature, **323**(6088), pp. 533–536.

[7] De, S., Maute, K., and Doostan, A., 2021, "Reliability-Based Topology Optimization Using Stochastic Gradients," Struct. Multidiscipl. Optim., **64**(5), pp. 3089–3108.

[8] Bottou, L., 2010, "Large-Scale Machine Learning With Stochastic Gradient Descent," Proceedings of COMPSTAT'2010: 19th International Conference on Computational Statistics, Paris France, Aug. 22–27, 2010 Keynote, Invited and Contributed Papers, Springer, pp. 177–186.

[9] Li, Z., Zhou, F., Chen, F., and Li, H., 2017, "Meta-sgd: Learning to Learn Quickly for Few-Shot Learning" arXiv preprint arXiv:1707.0983.

[10] Dokuz, Y., and Tufekci, Z., 2021, "Mini-Batch Sample Selection Strategies for Deep Learning Based Speech Recognition," Appl. Acoust., **171**, p. 107573.

[11] Qian, N., 1999, "On the Momentum Term in Gradient Descent Learning Algorithms," Neural Netw., **12**(1), pp. 145–151.

[12] Nesterov, Y., 1983, "A Method for Unconstrained Convex Minimization Problem With the Rate of Convergence O (1/k2)," Dokl. Akad. Nauk. SSSR, **269**(3), p. 543.

[13] Tieleman, T., and Hinton, G., 2012, "Lecture 6.5-rmsprop: Divide the Gradient by a Running Average of Its Recent Magnitude," COURSERA: Neural Netw. Mach. Learn., **4**(2), pp. 26–31.

[14] Kingma, D. P., and Ba, J., 2014, "Adam: A Method for Stochastic Optimization," CoRR, abs/1412.6980.

[15] Mitchell, M., 1996, *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, MA.

[16] Lu, T.-C., 2021, "Cnn Convolutional Layer Optimisation Based on Quantum Evolutionary Algorithm," Connect. Sci., **33**(3), pp. 482–494.

[17] Yu, Y., and Zhang, M., 2021, "Control Chart Recognition Based on the Parallel Model of Cnn and Lstm With Ga Optimization," Expert. Syst. Appl., **185**, p. 115689.

[18] Arhore, E. G., Yasaee, M., and Dayyani, I., 2022, "Optimisation of Convolutional Neural Network Architecture Using Genetic Algorithm for the Prediction of Adhesively Bonded Joint Strength," Struct. Multidiscipl. Optim., **65**(9), p. 256.

[19] Luo, X., Oyedele, L. O., Ajayi, A. O., Akinade, O. O., Delgado, J. M. D., Owolabi, H. A., and Ahmed, A., 2020, "Genetic Algorithm-Determined Deep Feedforward Neural Network Architecture for Predicting Electricity Consumption in Real Buildings," Energy AI, **2**, p. 100015.

[20] Gonçalves, C. B., Souza, J. R., and Fernandes, H., 2022, "Cnn Architecture Optimization Using Bio-inspired Algorithms for Breast Cancer Detection in Infrared Images," Comput. Biol. Med., **142**, p. 105205.

[21] Elsken, T., Metzen, J. H., and Hutter, F., 2019, "Neural Architecture Search: A Survey," J. Mach. Learn. Res., **20**(1), pp. 1997–2017.

[22] Tseng, I., Cagan, J., and Kotovsky, K., 2012, "Concurrent Optimization of Computationally Learned Stylistic Form and Functional Goals," ASME J. Mech. Des., **134**(11), p. 111006.

[23] Bai, S., Kolter, J. Z., and Koltun, V., 2018, "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling," ArXiv, abs/1803.01271.

[24] Zhang, R., Sun, F., Song, Z., Wang, X., Du, Y., and Dong, S., 2021, "Short-Term Traffic Flow Forecasting Model Based on Ga-tcn," J. Adv. Transpor., **2021**, pp. 1–13.

[25] Chen, Z., Chen, B., and Chen, X., 2022, "Remaining Useful Life Prediction of Turbofan Engine Based on Temporal Convolutional Networks Optimized by Genetic Algorithm," J. Phys. Conf. Series, **2181**(1), p. 012001.

[26] Tan, B., Li, Q., and Zhang, T., 2023, "Application of TCN Algorithm in Aircraft System," International Conference on Signal Processing, Computer Networks, and Communications (SPCNC 2022), Zhengzhou, China, Dec. 16–17, H. Wang, ed., Vol. 12626, International Society for Optics and Photonics, SPIE, p. 126262B.

[27] Zhu, J., Su, L., and Li, Y., 2022, "Wind Power Forecasting Based on New Hybrid Model With Tcn Residual Modification," Energy AI, **10**, p. 100199.

[28] Saxena, A., Goebel, K., Simon, D., and Eklund, N., 2008, "Damage Propagation Modeling for Aircraft Engine Run-to-Failure Simulation," 2008 International Conference on Prognostics and Health Management, Denver, CO, Oct. 6–9, pp. 1–9.

[29] Eastwick, C. N., and Johnson, G., 2008, "Gear Windage: A Review," ASME J. Mech. Des., **130**(3), p. 034001.

[30] Holland, J. H., 1992, "Genetic Algorithms," Sci. Am., **267**(1), pp. 66–73.

[31] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I., 2017, "Attention Is All You Need," Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, Long Beach, CA, Dec. 4–9, Curran Associates Inc., pp. 6000–6010.

[32] Zeng, A., Chen, M., Zhang, L., and Xu, Q., 2023, "Are Transformers Effective for Time Series Forecasting?," Proceedings of the AAAI Conference on Artificial Intelligence, Washington, DC, Feb. 7–14, Vol. 37, pp. 11121–11128.

[33] Hochreiter, S., and Schmidhuber, J., 1997, "Long Short-Term Memory," Neural Comput., **9**(8), pp. 1735–1780.

[34] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y., 2015, "Gated Feedback Recurrent Neural Networks," International Conference on Machine Learning, Lille, France, July 6–11, PMLR, pp. 2067–2075.

[35] Glorot, X., and Bengio, Y., 2010, "Understanding the Difficulty of Training Deep Feedforward Neural Networks," Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, Sardinia, Italy, May 13–15, pp. 249–256.

[36] Adam, L., and Dorota, L., 2012, "Roulette-Wheel Selection Via Stochastic Acceptance," Physica A: Stat. Mech. Appl., **391**(6), pp. 2193–2196.