A Lightweight Hybrid Architecture for Speech Recognition

Zhenzhou Liu[1,a,#], Chengdong Weng[1,b,#], Muyuan Liu[1,c,#], Haoxing Xu[1,d,#], Situo Xing[1,e,#], Boyu Luan[1,f,#]

[1]*Beijing 21st Century School, Beijing, China*
[a]3508639204@qq.com, [b]joe_wengbest368@163.com, [c]skycityliu@163.com, [d]ffxx616@163.com, [e]2127681619@qq.com, [f]13301017188@163.com
[#]Co-first author

Abstract: *This study proposes a lightweight hybrid architecture for speech recognition, integrating four convolutional layers with spectral normalization, two adaptive max-pooling layers, and two fully connected layers with dropout regularization. The design emphasizes computational efficiency through kernel pruning while maintaining consistent inference performance across hardware platforms. Evaluation using noisy speech datasets demonstrates robust recognition accuracy and real-time processing capabilities. Deployment validation confirms operational stability in edge computing environments, confirming suitability for resource-constrained applications requiring energy-efficient speech recognition.*

Keywords: *Speech Recognition, Deep Learning, Convolutional Neural Network, Fully Connected Neural Network, Pooling Layer*

1. Introduction

Language is the fundamental tool for human communication and expression. In the 1950s, speech recognition technology began to emerge, with the development of the Audrey system by Bell Labs, which successfully recognized the digits from 0 to 9. This marks the inception of research in this field [1]. Since then, speech recognition technology has evolved continuously and had increasing applications. With the emergence of the digital age, speech recognition has played an increasingly important role in our daily lives. Within the contexts of the digital era, speech recognition technology now has become a key component of social life [2]. As a crucial branch of natural language processing, speech recognition technology has showed a significant potent across various domains, including human-computer interaction, the development of intelligent assistants, and speech translation. The accuracy and intelligence of speech recognition technology will continue to improve with developments in deep learning and big data. Furthermore, with progress in hardware devices, speech recognition will become more and more efficient, enabling its application in real scenarios. These technological advancements not only drive progress in artificial intelligence, but also provide technical support for innovation and transformation in related industries.

Currently, several classic large-scale language models, such as BERT (containing 110 million to 340 million parameters), GPT-3 (with 175 billion parameters), and T5 (containing 11 billion parameters), have made significant progress in natural language processing. These models possess abilities in transfer learning, few-shot learning, and zero-shot learning, and exhibit exceptional contextual understanding and reasoning abilities [3]. Liu Yang et al. proposed a speech recognition system based on a one-dimensional convolutional neural network (1D-CNN). The authors thoroughly analyzed the system's framework design and its advantages in speech processing, providing detailed insights into the implementation of the system using TensorFlow, including model construction, training processes, and optimization strategies. Experimental results showed that the system demonstrated high recognition accuracy and stability in both noise-free and mildly noisy environments, and it maintained strong robustness in severely noisy conditions [4]. A neural speech recognition system called Listen, Attend and Spell (LAS) was proposed, which directly transcribes speech into characters, avoiding the use of traditional phoneme models, hidden Markov models (HMM), and other components commonly employed in conventional speech recognition systems. In LAS, the neural network architecture integrates the acoustic model, pronunciation model, and language model, making it an end-to-end trained system with the characteristics of an end-to-end model. Unlike DNN-HMM, CTC, and most other models, LAS does not make independence assumptions about the probability distribution of the output character sequence when processing acoustic sequences. Experimental results show that, in the Google Voice Search task, LAS achieved a word error rate (WER) of 14.1% without using a dictionary or external language model. When a language model was used for rescoring and beam search was applied to the top 32 hypotheses, WER decreased to 10.3%. In comparison, the state-of-the-art CLDNN-HMM model achieved a WER of 8.0% on the same dataset [5]. Kuniaki Noda proposed an audiovisual speech recognition (AVSR) system based on the connectionist hidden Markov model (HMM), which combines deep denoising autoencoders to extract audio features, convolutional neural networks to extract visual features, and a multi-stream HMM to integrate both in order to enhance recognition performance. Experimental results showed that, at a 10 dB signal-to-noise ratio, the denoised MFCC improved recognition accuracy by approximately 65%, while combining visual features further enhanced performance when the signal-to-noise ratio fell below 10 dB [6]. In modern speech recognition, existing deep learning models often face challenges related to excessively large model parameters, high computational resource requirements, and excessive energy consumption. Therefore, designing a speech recognition model that is parameter-efficient, computationally efficient, and energy-efficient has become a critical challenge for the advancement of speech recognition technology [7].

To address this issue, this study proposes a model architecture that combines Convolutional Neural Networks (CNN) [8] and Fully Connected Neural Networks (FCN) [9]. This model consists of four convolutional layers, two pooling layers [10], and two fully connected layers. The purpose of the design is to effectively reduce the number of model parameters, decrease computational complexity and energy consumption, and ultimately improve the efficiency and practicality of speech recognition tasks.

2. System design

2.1. Overall design

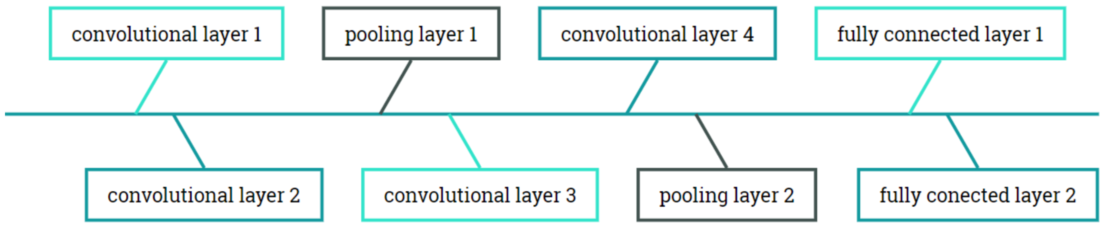

Figure 1: The Arrangement of Neural Network Layers

By the combination of different types of neural network layers and the integration of four optimization algorithms, this model effectively improves the accuracy of speech recognition (Figure 1). The convolutional layer, pooling layer, and fully connected layer are organized in the sequence of "two convolutional + pooling layers," "two convolutional + pooling layers," and "two fully connected layers."

2.1.1. Convolutional Neural Network Layer

In Convolutional Neural Networks (CNNs), which consist of convolutional layers, pooling layers, and fully connected layers, convolutional layer is the core component, which is responsible for extracting local features from the input audio signal. In that layer, the feature matrix is multiplied by the convolution kernel, and the sum is calculated and a feature map that reflects the characteristics of the data in the region is generated. Convolution operations feature weight sharing, meaning that the same convolution kernel applies the same parameters across different regions, which not only reduces the model's parameter count and computational complexity but also enhances the model's generalization ability [12]. Key factors in the convolutional layer include stride and padding. Stride means the step size of the convolutional kernel, while padding affects the spatial dimensions of the output feature map.

2.1.2. Pooling Layer

The next step after the convolutional layer is the pooling layer, which is responsible for sampling the output of the convolutional layer. The pooling layer helps decrease computational and storage demands through reducing the spatial size of the feature map, when it is preserving important features [13]. The primary pooling methods include max pooling and average pooling. The most commonly used pooling method is max pooling, which selects the maximum value in a region to retain prominent features, which leads to stronger robustness against minor distortions and noise. In contrast, average pooling compresses features by averaging all values within the region, which tends to preserve audio features less effectively. Therefore, this model utilizes max pooling. Pooling not only helps reduce computational burdens but also effectively prevents overfitting, avoiding a decrease in test accuracy due to excessive fitting of the training data.

2.1.3. Fully Connected Layer

The fully connected layer is responsible for mapping the features extracted by the convolutional and pooling layers to the final output. In the fully connected layer, each neuron is connected to every neuron in the previous layer, but neurons within the same layer are not interconnected. Through these neurons, the model integrates local features and forms a global understanding [12]. A notable characteristic of the fully connected layer is the large number of parameters it contains, which results in higher computational and storage demands. Therefore, the fully connected layer is typically combined with optimization methods such as dropout and L2 regularization to reduce overfitting and improve generalization. Dropout effectively prevents overfitting, while L2 regularization smooths the weights, preventing the model from overly depending on noise in the training data.

2.1.4. System Process

The input layer of the model consists of 50 single-channel 32x32 images, which are processed by two layers of 3x3 convolutional kernels using the ReLU activation function. After this, the feature map size is reduced through max pooling. Subsequently, convolutional operations are performed in the third and fourth layers, with ReLU activation functions applied for feature extraction and further pooling. The final output consists of a feature map of size 8x8 with 32 input channels. The first layer contains 128 output nodes and uses the ReLU activation function. The second layer contains 10 neurons corresponding to the ten classification categories, with no activation function applied. Overall, the model uses an optimization strategy that combines learning rate decay and the Adam optimizer. Learning rate decay gradually reduces the learning rate during the training process, allowing the model to fine-tune the weights in the later stages of training, avoiding excessive oscillation and overfitting. The Adam optimizer adjusts the learning rate for each parameter based on the gradient, making the training process more efficient and enabling rapid convergence on complex loss functions, reducing the model's sensitivity to hyperparameters.

2.2. Model Optimization

In this project, to prevent model overfitting, we adopted three strategies: regularization, Dropout, and the Adam optimizer. Below is a detailed explanation of these optimization methods:

2.2.1. Regularization

Regularization is generally divided into L1 and L2 regularization. Models using L1 regularization are called lasso regression, and those using L2 regularization are referred to as ridge regression, which helps achieve feature sparsity [13]. This characteristic makes L1 regularization particularly suitable for feature selection.

L1 Loss Function, also known as Least Absolute Deviation (LAD), is defined by the following formula:

$$\sum_{i=0}^{n}|y_i - h(x_i)| \tag{1}$$

S represents the sum, y_i is the value of the i-th sample, and h(xi) is the model output. The optimization objective is to minimize the difference between these values, indicating better model accuracy [13].

L2 Loss Function, also referred to as Least Squares Error (LES), is represented by the formula below:

$$\sum_{i=0}^{n}\left(y_i - h(x_i)\right)^2 \tag{2}$$

The L2 loss minimizes the squared differences between the predicted and actual values, effectively constraining parameter magnitudes to ensure smoother weight distributions. This reduces over-reliance on specific features and prevents overfitting [13].

2.2.2. Dropout

Dropout is a regularization technique commonly used during neural network training. Its core idea is to randomly set a portion of neuron outputs to zero during each forward pass. This random masking mechanism prevents the network from relying too heavily on particular neurons, thereby enhancing model robustness. By suppressing unnecessary neuron co-adaptations, Dropout helps the model learn more generalized feature representations, ultimately improving performance on unseen data[14].

2.2.3. Adam Optimizer

The Adam (Adaptive Moment Estimation) optimizer is an adaptive learning rate optimization algorithm that combines the advantages of momentum and RMS Prop. It can efficiently optimize non-stationary objective functions and is relatively insensitive to hyperparameter settings, making it widely used in deep learning model training[15].

1) Basic Concept of Adam

Adam dynamically adjusts the learning rate for each parameter through first-order moment estimation (mean of gradients) and second-order moment estimation (mean of squared gradients), thus enhancing optimization stability and convergence speed.

Momentum Optimization: Applies exponential moving averages to gradients, stabilizing parameter updates and reducing gradient oscillations.

RMS Prop: Uses exponential moving averages of squared gradients to adaptively adjust learning rates, preventing excessively large or small updates[15].

2) Mathematical Formulation of Adam

Given the gradient gt computed at training step t, Adam updates parameters as follows:

First-order moment estimation (momentum):

$$m_t=\beta_1 m_{t-1}+(1-\beta_1)g \tag{3}$$

Where m_t is the exponentially weighted average of gradients, and β_1 (typically 0.9) is the momentum decay rate.

Second-order moment estimation (squared gradients):

$$v_t=\beta_2 v_{t-1}+(1-\beta_2)gt^2 \tag{4}$$

Where v_t is the exponentially weighted average of squared gradients, and β_2 (typically 0.999) is the RMS Prop decay rate.

Bias Correction:

To address bias in the initial estimates of mt and v_t, perform bias correction:

$$m_t= m_t/(1-\beta^t{}_1),\ v_t= v_t/(1-\beta^t{}_2) \tag{5}$$

Parameter Update:

$$\theta_t=\theta_{t-1}\text{-}\alpha/\sqrt{v_t}+\epsilon \tag{6}$$

Where α is the learning rate, and ϵ (typically 10^{-8}) prevents division by zero[15].

3. Data Processing

3.1. Structure of Audio Data and Dataset Division

3.1.1. Data Structure

The raw data consists of 10 folders, each representing a digit from 0 to 9, with each folder containing 150 audio files. This structure provides a clear framework for the organization and management of data, facilitating subsequent processing and analysis.

3.1.2. Dataset Division

In the entire audio dataset, we adopted a specific division method. Four-fifths of the audio files are allocated to the training set, totaling 10×150×4/5 = 1200 audio files. This training set is crucial as it provides rich material for the model's training, allowing the model to learn various feature patterns within the audio data. The remaining one-fifth, that is, 10×150×1/5 = 300 audio files, is designated as the testing set. The testing set is primarily used to evaluate the model's performance; testing on this portion of data that was not involved in training allows for an accurate measurement of the model's generalization ability and accuracy.

3.2. Original data processing

The following is a detailed description of the data processing flow in Figure 2:

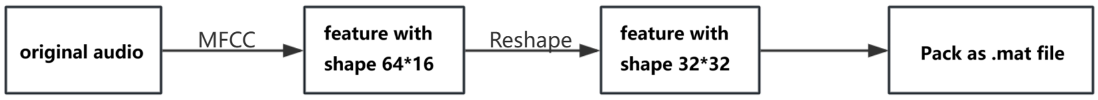

Figure 2: Data Processing Flow

3.2.1. Traversing Raw Data

Utilizing an object-oriented high-level programming language to traverse the entire raw data. This language has high flexibility and operability, effectively handling complex data structures. During the traversal, subsequent operations are performed on each audio file.

3.2.2. Audio Feature Extraction

(1) Application of the MFCC Method

Using the MFCC (Mel-Frequency Cepstral Coefficients) method to obtain the features of each audio file. MFCC is a widely used feature extraction method in the audio processing field, effectively capturing the spectral characteristics of audio signals. By converting the audio signal to the Mel frequency scale and then performing a discrete cosine transform, the resulting coefficients can reflect the essential characteristics of the audio, similar to generating a unique "fingerprint" for each audio file.

(2) Standardization of Feature Format

The extracted MFCC features are standardized to a format of 64×16. This step helps standardize the features of different audio files, facilitating subsequent data operations and model input. In cases where the features of different audio files may vary, standardizing the format ensures consistency in the data during the subsequent processing.

(3) Reshaping

Using the reshape method to change the shape of the MFCC to 32×32×1. This operation adjusts the data structure to better meet the input requirements of specific models or algorithms. For example, in certain deep learning models, there are specific requirements for the shape of input data. Through this reshaping, the audio feature data can better fit the model structure, thereby improving the model's performance.

3.2.3. Data Packaging

The processed data and corresponding labels are packaged and output as .mat files. The .MAT file is a commonly used data storage format, widely applied in data mining, machine learning, and other fields. This format can effectively preserve data and label information, facilitating subsequent research and analysis. For instance, during model training and evaluation, data and labels can be directly read from the .mat files, reducing the time and complexity of data preprocessing.

3.3. Experiment

The dataset consists of many recordings of digits 0 to 9 by different speakers, and the dataset is relatively small. Mel Frequency Cepstral Coefficients features are extracted through data processing. Then, the load program loads and preprocesses the training and testing data from two processed MATLAB files. By using a one-hot encoding function, the labels for digits 0-9 are transformed into one-hot encoded vectors, enabling the model to distinguish between the digits in vector form better. One-hot encoding converts categorical labels into vector form, as shown in Table 1 below.

Table 1. One-Hot Encoding Corresponding to Class Labels

Class Label	One-Hot Encoded Vector
0	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
1	[0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
2	[0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
3	[0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
4	[0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
5	[0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
6	[0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
7	[0, 0, 0, 0, 0, 0, 0, 1, 0, 0]
8	[0, 0, 0, 0, 0, 0, 0, 0, 1, 0]
9	[0, 0, 0, 0, 0, 0, 0, 0, 0, 1]

To further enhance the training efficiency and accuracy of the model, the load program normalizes the data. Normalization scales the data to a range between 0 and 1, and it ensures a uniform training scale for all data, which prevents training deviations due to large disparities among data points. The TensorFlow framework is then used to construct a Convolutional Neural Network model, which is trained using the processed data.

After training, model accuracy on the test data is evaluated and printed out, and adjustments are made to the dropout rate, iteration steps, decay rate, base learning rate, and train batch size. Adjusting the dropout rate enhances the model's generalization ability and allows it to learn the general characteristics of digits. Modifying iteration steps enables models to have a sufficient learning time to improve accuracy. Altering the base learning rate facilitates rapid convergence to an optimal solution and also maintains the stability and efficiency. Adjusting the decay rate ensures stability and optimizes model performance. Changing the train batch size affects the number of training samples per iteration, and it combined with iteration steps can accelerate and refine the training process. Notably, the dropout rate and iteration steps impact model training significantly and are the primary parameters for adjustment.

Table 2. Training and Testing Data Results

Dropout Rate	Base Learning Rate	Decay Rate	Iteration Steps	Train Batch Size	Accuracy (%)
0.96	0.001	0.9	500	50	84%
0.9	0.001	0.99	500	50	86.90%
0.9	0.003	0.5	500	50	90.50%
0.9	0.001	0.9	1000	50	83.64%
0.9	0.001	0.9	1000	50	83.64%
0.999	0.003	0.5	1000	50	86.20%
0.92	0.001	0.99	1000	50	87.20%
0.88	0.001	0.99	1000	50	88%
0.9	0.001	0.99	1000	50	88%
0.96	0.001	0.99	1000	50	88%
0.9992	0.001	0.99	2000	50	86.20%
0.9	0.003	0.999	150	10	94%

During model optimization, the highest average test accuracy was achieved with the following parameters: dropout rate of 0.9, iteration steps of 150, base learning rate of 0.003, decay rate of 0.999, and train batch size of 10. This configuration includes a high dropout rate to prevent overfitting, which is usually beneficial with small datasets. Setting the iteration steps to 150 allows the model to learn data features adequately while avoiding overfitting, thus improving test accuracy. A lower base learning rate reduces gradient oscillation and maintains accuracy, while a slower learning rate decay preserves accuracy and enhances model generalization. A training batch size of 10 improves generalization and adaptability, and its associated gradient fluctuations are mitigated by the low learning rate decay (Shown in Table 2).

During model training, the relationship between mini-batch loss and mini-batch accuracy is recorded and plotted as the Training Loss Curve, shown in Figure 3. The orange line represents the ratio of mini-batch loss to iteration steps. The trend of the orange line indicates high loss in the initial iterations, followed by a significant decrease as training progresses. The overall trend suggests that the model is progressively optimized through iterations, achieving better data fitting.

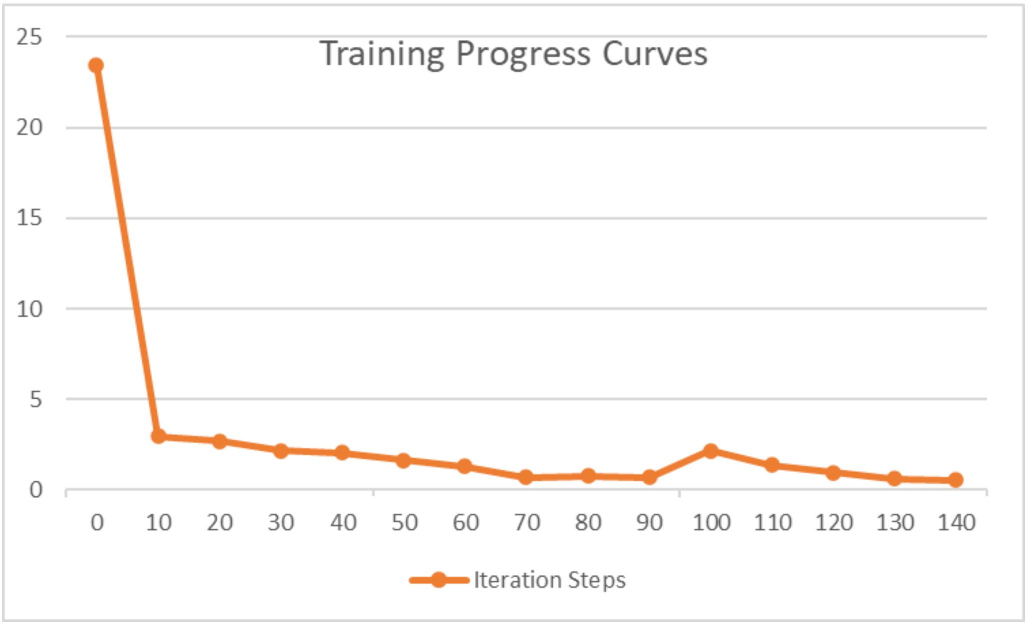

Figure 3. Training Progress Curves

4. Conclusion

In this study, we have proposed a concise and efficient deep learning network architecture designed to address the trade-off between accuracy and efficiency in speech recognition tasks under low-power and low-parameter conditions. Through structural optimization, this network achieves high speech recognition accuracy while maintaining a relatively small model size. To further enhance the performance of the model, we plan to carry out a series of extensions and tests in future research. Specifically, we aim to expand the existing speech recognition dataset by incorporating a broader range of diverse speech samples, including variations in noise, accents, and speaking rates, in order to improve the model's generalization ability and robustness in real-world applications. We believe that with further optimization, the deep learning network will continue to provide efficient and accurate speech recognition services under low-power and low-parameter conditions, effectively addressing the challenges posed by different speech environments.

References

[1] Ma, H., Tang Z.B., Zhang Y., & Zhang Q.L. (2022). Survey on Speech Recognition. Computer Systems Applications, (01), 1-10.
[2] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A. R., Jaitly, N., ... & Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal processing magazine, 29(6), 82-97
[3] Wang, J, K., Qin, D, H., Bai, F, B., et al. (2015). A review of research on fusion techniques for speech recognition and large language models. Computer Engineering and Application,1-13.
[4] Liu, Y., Lian, M, M. (2024). Construction methods of speech recognition systems based on 1D convolutional neural network. Audio Engineering, 48(10): 77 – 79
[5] Chan, W., Jaitly, N., Le, Q., & Vinyals, O. (2016.). Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In: 2016 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE, p. 4960-4964.
[6] Noda, K., Yamaguchi, Y., Nakadai, K., Okuno, H. G., & Ogata, T. (2015). Audio-visual speech recognition using deep learning. Applied intelligence, 42, 722-737.
[7] Yu, D., Wei, X., Zhao, X., Zhang, X., & Xu, B. (2016). Deep speech 2: End-to-end speech recognition in English and Mandarin. In Proceedings of the 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),IEEE. (pp. 5705-5709). .
[8] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document

recognition. Proceedings of the IEEE, 86(11), 2278-2324.
[9] Scabini, L. F., & Bruno, O. M. (2023). Structure and performance of fully connected neural networks: Emerging complex network properties. Physica A: Statistical Mechanics and its Applications, 615, 128585.
[10] Boureau, Y. L., Ponce, J., & LeCun, Y. (2010). A theoretical analysis of feature pooling in visual recognition. In Proceedings of the 27th International Conference on Machine Learning (ICML-10) (pp. 111-118).
[11] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25.
[12] Simonyan, K. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.
[13] Li, J., & Gao, G. (2023). Digital construction of geophysical well logging curves using the LSTM deep-learning network.Frontiers in Earth Science, 10, 1041807.
[14] He, Z. (2024). Optimization and Application of Natural Language Processing Models Based on Deep Learning.Journal of Artificial Intelligence Practice, 7(1), 109–115
[15] Li, D., Ortegas, K. D., & White, M. (2023). Exploring the computational effects of advanced deep neural networks on logical and activity learning for enhanced thinking skills.Systems, 11(7), 319.