Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2023.046692

ARTICLE

PAL-BERT: An Improved Question Answering Model

Wenfeng Zheng[1], Siyu Lu[1], Zhuohang Cai[1], Ruiyang Wang[1], Lei Wang[2] and Lirong Yin[2,*]

[1]School of Automation, University of Electronic Science and Technology of China, Chengdu, 610054, China

[2]Department of Geography and Anthropology, Louisiana State University, Baton Rouge, LA, 70803, USA

*Corresponding Author: Lirong Yin. Email: lyin5@lsu.edu

Received: 11 October 2023 Accepted: 14 December 2023 Published: 11 March 2024

ABSTRACT

In the field of natural language processing (NLP), there have been various pre-training language models in recent years, with question answering systems gaining significant attention. However, as algorithms, data, and computing power advance, the issue of increasingly larger models and a growing number of parameters has surfaced. Consequently, model training has become more costly and less efficient. To enhance the efficiency and accuracy of the training process while reducing the model volume, this paper proposes a first-order pruning model PAL-BERT based on the ALBERT model according to the characteristics of question-answering (QA) system and language model. Firstly, a first-order network pruning method based on the ALBERT model is designed, and the PAL-BERT model is formed. Then, the parameter optimization strategy of the PAL-BERT model is formulated, and the Mish function was used as an activation function instead of ReLU to improve the performance. Finally, after comparison experiments with traditional deep learning models TextCNN and BiLSTM, it is confirmed that PAL-BERT is a pruning model compression method that can significantly reduce training time and optimize training efficiency. Compared with traditional models, PAL-BERT significantly improves the NLP task's performance.

KEYWORDS

PAL-BERT; question answering model; pretraining language models; ALBERT; pruning model; network pruning; TextCNN; BiLSTM

1 Introduction

With the rapid growth of big data, the volume of complex network information is increasing exponentially. As a result, people are increasingly relying on search engines to retrieve relevant information efficiently. However, traditional search engine algorithms are no longer capable of adequately meeting user demands in the face of this overwhelming data volume. Therefore, there is a growing need for more efficient and effective information retrieval methods. The question-answering system (QA) [1,2] is more suitable for users' habits. Users only need to input natural language questions to get the relevant answer information returned by the system, which greatly reduces the time cost for users to obtain information. The successful application of QA will liberate human beings from a large amount of repetitive work, change human production mode, and have an inestimable impact on the progress of human society.

For a long time, the development of deep learning in NLP has been far from its performance in image processing. While image recognition algorithms [3,4] have caught up with and even surpassed human performance, the progress in combining the NLP field with deep learning has been unsatisfactory. The emergence of BERT [5–7] broke this situation. It demonstrated remarkable results in the high-level evaluation of machine reading comprehension, specifically in SQuAD1.1 [8], it outperformed human performance across both measurement indicators and achieved outstanding results in 11 distinct NLP tests. These accomplishments included pushing the GLUE benchmark to 80.4% (a significant increase of 7.6%) and achieving an accuracy of 86.7% (a notable increase of 5.6%) in MultiNLI. The emergence of BERT is like a watershed, which makes people see the great development potential of neural network models in NLP. Despite the great success and social attention of the current emergence of ChatGPT, which is known for being able to answer questions quickly and accurately, the authenticity of the answers has also been widely questioned [9]. In contrast, BERT has its advantages. BERT aims to understand the context of the words in a sentence, thus providing a more accurate answer to a question. BERT belongs to the upstream structure model, which depends on its own characteristics. In order to achieve better results in specific applications such as the QA system, it is also necessary to build the corresponding downstream structure model. In addition, the huge network structure and various parameter volumes of the BERT model are also great obstacles to completing the actual task. Therefore, it is meaningful to optimize the BERT model for better effect.

Pruning [10–12] belongs to a method of model compression. In short, by deleting some unimportant neurons, the number of weights and computations are greatly reduced to improve the operation efficiency of the model. In practical application, a large neural network has some redundancy because of the limitations of application scenarios. Pruning technology is the way to reduce this redundancy. Its core idea is to minimize the connection of neurons on the premise of ensuring no great loss of accuracy.

The pruning process is shown in Fig. 1. First, complete the training of a neural network on a dataset and then use similar but different datasets for pruning. For the trained neural network, calculate the neurons that need pruning according to certain standards, remove the neurons with the least impact, and then conduct fine-tuning [13] on the new dataset. Continue this process until the neurons are reduced to the required goal, and the accuracy is within the loss range.

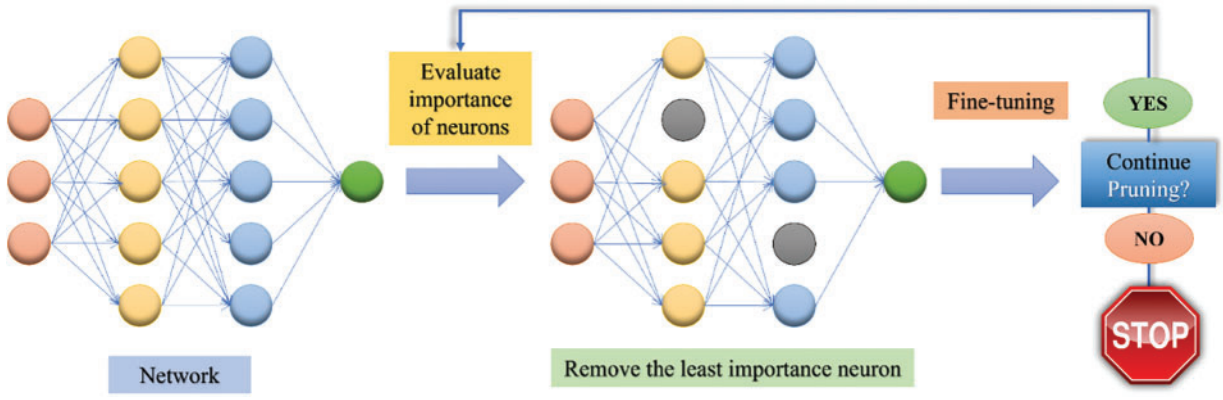

Figure 1: Schematic diagram of pruning process

Many experts and scholars have studied various pruning methods. EvoPruneDeepTL [14] is an evolutionary pruning model based on transfer learning, which replaces the last fully connected layer

with a sparse layer optimized by genetic algorithms, reducing the number of active neurons in the last layer while improving accuracy. A method called progressive local filter pruning (PLFP) was proposed by Wang et al. [15], which retains the capacity of the original model when pruning and achieves excellent performance. Aiming at the problem that the existing residual network usually has a large model volume, a new hierarchical pruning-multi-scale feature fusion residual network (HSP-MFFRN) for IFD was proposed [16], which uses multiple multi-scale feature extraction and fusion modules to extract, fuse and compress multi-scale features without changing the convolution filter size. Also, redundant channels are removed by pruning. The results show that pruning can effectively compress parameters and model volume, and achieve better diagnostic performance.

The traditional zero-order network pruning (setting a threshold for the absolute value of the parameters in the model, which is higher than its retention and lower than its zero setting) is not suitable for the migration learning scenario, because in this scenario, the model parameters are mainly affected by the original model, but need to be fine-tuned and tested on the target task. Therefore, pruning directly according to the model parameters may lose the knowledge from either source or target tasks.

The model used in this paper is based on ALBERT [17], short for "A Lite BERT," is a more efficient and compact variant of the original BERT model. Developed by Google Research, ALBERT maintains high-performance levels in natural language processing tasks while addressing BERT's computational and memory requirement issues. Key features of ALBERT include parameter sharing between layers, factorized embedding parameterization, a Sentence Order Prediction (SOP) pretraining task, and improved training strategies. These innovations reduce the model's size and resource demands, making it a practical choice for various NLP applications.

This study optimizes the ALBERT model through pruning technology to improve its performance in question and answer tasks. The innovation points are mainly reflected in the following aspects:

ALBERT is introduced as the research basis. It is a more efficient and compact variant of the BERT model. It solves BERT's problems in computing and memory requirements through features such as parameter sharing, decomposition of embedded parameters, and Sentence Order Prediction. For low-power environments, a special training optimization strategy is designed, including gradient accumulation and 16-bit quantization, to improve training efficiency. PAl-BERT was introduced as the first derivative pruning model to reduce computational requirements and improve model efficiency by retaining parameters far away from zero during the fine-tuning process. Through these innovations, the article attempts to fill some shortcomings in deep learning applications in the field of natural language processing, proposes a more efficient and compact model, and further improves performance through pruning technology. This series of innovations and improvements make the article uniquely valuable in optimizing question and answer systems.

In our experimental settings, we follow ALBERT's original settings and use the SQuAD 1.1 and SQuAD 2.0 datasets as pre-training experimental datasets. We select TextCNN and BiLSTM as comparative models. Through experimental comparisons and analysis in the dataset CMRC 2018, AP-BERT achieves an accuracy rate of 81.5% in question-answering tasks, demonstrating superior performance to baseline models. Experimental results verify the effectiveness of the proposed model.

2 Dataset

2.1 SQuAD 1.1 and SQuAD 2.0

In this study, SQuAD 1.1 and SQuAD 2.0 [8] were used as English pre-training experimental dataset, for it has sufficient sample number and scientific selection of samples. The dataset can be obtained from: (https://rajpurkar.github.io/SQuAD-explorer/). Table 1 shows the amount and distribution of SQuAD 1.1 and SQuAD 2.0.

Table 1: Number and distribution of positive and negative samples in the SQuAD dataset

	SQuAD 1.1	SQuAD 2.0
Train		
Total samples	87599	130319
Negative samples	0	43498
Total articles	442	442
Articles with negatives	0	285
Development		
Total samples	10570	11873
Negative samples	0	5945
Total articles	48	35
Articles with negatives	0	35
Test		
Total samples	10570	11873
Negative samples	0	5945
Total articles	48	35
Articles with negatives	0	35

Various studies based on the old version of datasets have been well performed. Therefore, the new version of datasets introduces manually marked “unanswerable” questions, thus increasing the difficulty of the whole task.

2.2 CMRC 2018

For another experimental dataset, we will use the Chinese machine reading comprehension dataset CMRC 2018 [18] released by iFLYTEK Joint Laboratory of Harbin University of Technology. The dataset can be obtained from: (https://ymcui.com/cmrc2018/). The content of the dataset is from Chinese Wikipedia, the questions are written manually, and the data form provided is like SQuAD. The training set contains about 10000 pieces of data. Table 2 shows the specific information of the dataset.

Table 2: CMRC 2018 sample quantity

	Train	Development	Test	Challenge
Number of questions	10321	3351	4895	504
Average answers per question	1	3	3	3

(Continued)

Table 2 (continued)

	Train	Development	Test	Challenge
Maximum article characters	962	961	980	916
Maximum question characters	89	56	50	47
Maximum answer characters	100	85	92	77
Average article characters	452	469	472	464
Average question characters	15	15	15	18
Average answer characters	17	9	9	19

However, there are still some differences between Chinese and English datasets, so it is also used as a supplement to the experimental dataset to explore the differences in model and preprocessing in non-English cases. Each article will give several relevant questions, and each question has several manually marked reference answers. Note that each reference answer is considered correct during the evaluation. To confirm the diversity of problems, the dataset includes six common types of problems and other types except common problems. The statistical table of problem types is the same as that in Table 3.

Table 3: CMRC2018 question type statistics

Question type	Percentage
When	12.8%
Where	12.3%
Who	8.6%
What	7.8%
Why	5.7%
How	1.2%
Others	51.4%

3 Method

3.1 Basic Framework of ALBERT Model

ALBERT is a more efficient version of BERT, and it makes improvements in three main areas:

1. Embedded Layer Restructuring:

In BERT, the size of the word embedding layer matches that of the hidden layers. ALBERT, however, suggests a change. It argues that the embedded layer primarily holds context-free information while the hidden layers add context. So, the hidden layer should have a higher dimension. To avoid increasing the parameters in the embedded layer too much when increasing the hidden layer dimension, ALBERT decomposes the embedded layer and introduces an extra embedded layer.

2. Cross-Layer Parameter Sharing:

ALBERT uses a mechanism where all layers share parameters to enhance efficiency. While other methods exist for sharing parameters within specific parts of the model, ALBERT shares all parameters across all layers. This results in smoother transitions between layers, suggesting that parameter sharing improves the stability of the model.

3. Sentence Order Prediction (SOP) Task:

BERT introduced Next Sentence Prediction (NSP) to improve downstream tasks using sentence pairs. However, ALBERT introduces an alternative task called Sentence Order Prediction (SOP). In SOP, two consecutive paragraphs from a single document are used as a positive sample, and a negative sample is created by switching their order. ALBERT argues that NSP is less effective than SOP because it's relatively easier. ALBERT combines topic prediction and coherence prediction into a single task, which helps maintain high scores even if NSP struggles with coherence prediction.

3.2 Design of QA Model Based on ALBERT Improvement

Based on ALBERT model, the model built in this section optimization and pruning are carried out to design a model suitable for QA tasks.

3.2.1 Optimization Strategy of Model Training

In this study, all experimental GPUs are equipped with NVIDIA RTX2060 graphics cards. While these cards offer decent computing power, they may fall short when handling the ALBERT model. Consequently, this section introduces two optimization strategies aimed at reducing the hardware demands for model training. To attain satisfactory training results on platforms with limited computing power, we employ the gradient accumulation method and 16-bit precision training.

The first optimization strategy is gradient accumulation. Take Pytorch as an example. In traditional training of neural networks, Pytorch calculates the gradient after each 'backward ()', and the gradient will not be cleared automatically. If it is not cleared manually, the gradient will continue to accumulate until "CUDA out of memory" appears and an error is reported. Generally, the process is shown in Fig. 2.

From the Fig. 2a, we can clearly see that there are the following five steps:

1) Reset gradient to 0 after the previous batch calculation.

2) Forward propagation, input data in the network to obtain the predicted value.

3) Calculate the loss value according to the predicted value and label.

4) Calculate the parameter gradient by backpropagation through loss.

5) Update the network parameters by the gradient calculated in the previous step.

The gradient accumulation method is to obtain one batch at a time, calculate the gradient once, and continuously accumulate without clearing. As shown in Fig. 2b, the training steps of gradient accumulation are as follows:

1) Forward propagation, input data in the network to obtain the predicted value.

2) Calculate the loss value according to the predicted value and label.

3) Standardize the loss function.

4) Calculate the parameter gradient by back propagation through loss.

5) Repeat steps 1 to 4 to accumulate the gradient instead of resetting.

6) After the gradient accumulation reaches a fixed number of times, update the parameters, and reset the gradient to 0.

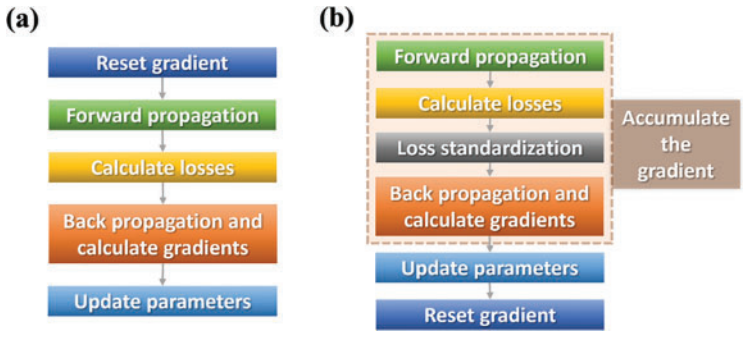

Figure 2: Comparison of traditional training process of neural network (a) and gradient accumulation training process (b)

Gradient accumulation expands the video memory in a disguised way. For example, if the gradient accumulation is set to 6, it can be approximately regarded as expanding the video memory by 6 times. Although the effect will be worse than the actual 6 times if the experimental conditions are not abundant, the gradient accumulation law has high-cost performance.

The second optimization strategy is 16-bit accuracy training [19,20].

Conventionally, models have been trained with 32-bit precision numbers. However, recent research has elucidated that employing 16-bit precision models can yield commendable training results. The foremost rationale for embracing 16-bit precision resides in its capacity to mitigate the memory demands imposed by model parameters and intermediate activations. The computational efficiency of 16-bit arithmetic operations, particularly on GPU hardware, is an instrumental factor. The salient characteristic of 16-bit arithmetic operations is their enhanced parallelizability, resulting in expedited computations. Training deep learning models with 16-bit precision bears the potential to significantly expedite the training process, predominantly by curtailing the temporal overhead incurred in memory transfers and arithmetic operations.

The reduction in precision not only facilitates swifter computations but also leads to a reduction in memory footprint. Storing model parameters and interim activations in a 16-bit format consumes only half the memory compared to 32-bit precision. In numerous cases, models trained with 16-bit precision have demonstrated performance on par with models trained with 32-bit precision. Furthermore, the advantages of 16-bit precision extend to model deployment, particularly in contexts marked by resource constraints, such as edge devices or mobile platforms.

Practically implementing 16-bit precision mandates the installation of the Apex library, an indispensable tool within the PyTorch framework, thoughtfully crafted by NVIDIA. Subsequently, a conscientious transition to 16-bit precision ensues, necessitating adjustments to the data types employed for model weights, activations, and gradients–a transition from ‘float32’ to ‘float16’. To address potential numerical instability issues, Automatic Mixed Precision (AMP) is employed to dynamically manage the precision during forward and backward passes.

3.2.2 First-Order Network Pruning Design Based on ALBERT

Traditional zeroth-order network pruning, which involves setting a threshold for the absolute values of model parameters and retaining those above it while zeroing out those below it, is not suitable for transfer learning scenarios. In transfer learning, model parameters are primarily influenced by the original model but require fine-tuning and testing on the target task. Therefore, directly pruning based on the model parameters themselves may result in the loss of knowledge from either the source task or the target task.

However, first derivative pruning relies on gradients calculated during training to identify and remove less important model parameters. This approach tends to preserve task-specific information better than zero-order pruning. By targeting specific parameters, first-order pruning can produce smaller, more efficient models, making them suitable for deployment in resource-constrained environments.

For the reasons mentioned above, this paper proposes a first-order derivative pruning model for ALBERT, named AP-BERT (where 'A' stands for ALBERT and 'P' stands for Pruning). This model aims to retain parameters that deviate further from zero during the fine-tuning process to mitigate the loss of knowledge in transfer learning scenarios.

Specifically: for model parameters W, give them importance scores S of the same size, then prune mask can be shown as Eq. (1):

$$\boldsymbol{M} = top_v(\boldsymbol{S}) \tag{1}$$

In the forward propagation process, the neural network uses the parameters with mask to calculate the output components a_i, as shown in Eq. (2):

$$a_i = \sum_{n}^{k=1} W_{i,k} M_{i,k} x_k \tag{2}$$

In back propagation, using the idea of Straight-Through Estimator [21], omit $top_v()$ to approximate the gradient of the loss function $\mathscr{L}$ to the importance score S, as shown in Eq. (3):

$$\frac{\partial \mathscr{L}}{\partial S_{i,j}} = \frac{\partial \mathscr{L}}{\partial a_i} \frac{\partial a_i}{\partial S_{i,j}} = \frac{\partial \mathscr{L}}{\partial a_i} W_{i,j} x_j \tag{3}$$

For model parameters, there is shown as Eq. (4):

$$\frac{\partial \mathscr{L}}{\partial W_{i,j}} = \frac{\partial \mathscr{L}}{\partial a_i} M_{i,j} x_j \tag{4}$$

Combining the above two equations, omitting the mask matrix of 0 and 1, we can get Eq. (5):

$$\frac{\partial \mathscr{L}}{\partial S_{i,j}} < 0 \tag{5}$$

According to the gradient decrease, when $\frac{\partial \mathscr{L}}{\partial S_{i,j}} < 0$, the importance $S_{i,j}$ increases, the positive and negative of $\frac{\partial \mathscr{L}}{\partial W_{i,j}}$ and $W_{i,j}$ are different. It means that only when the positive parameter becomes larger or the negative parameter becomes smaller during back propagation, can we get a greater importance score and avoid being pruned.

3.2.3 Parameter Optimization Strategy of PAL-BERT Model

To make the established PAL-BERT achieve the best effect in question-and-answer tasks, it is also necessary to select and test its specific internal parameters to find out the optimal scheme. It is divided into two aspects.

1. Selection of different layers. The official ALBERT_ large model contains a total of 24 layers of coding structure, but this structure is like a black box. It is certain that each layer will contain different semantic information, but we have not decided what the specific semantic information contained in each layer is. Therefore, in order to find the most effective layer or combination, we need to analyze the different layers of PAL-BERT.

2. Avoid overfitting during training. It is necessary to select an optimizer which has an appropriate learning rate to train PAL-BERT model.

The lower layer of ALBERT always has more information, while the upper layer may contain less information. In order to adapt to this situation, it is necessary to select different training rates for different layers.

The iterative method for the parameters of each layer can be shown in Eq. (6):

$$\theta_t^l = \theta_{t-1}^l - \eta^l \cdot \nabla_{\theta^l} \cdot J(\theta) \tag{6}$$

θ_t^l represents the iteration parameter of t step, and η^l is the learning rate of layer l. Its calculation method is shown in the Eq. (7):

$$\eta^l = \varepsilon \cdot \eta^k \tag{7}$$

where ε represents the decay coefficient. When $\varepsilon > 1$, the learning rate decays layer by layer, and $\varepsilon < 1$ indicates that the learning rate expands layer by layer. When $\varepsilon = 1$, the learning rate does not change, which is similar to the random gradient descent (SGD) under normal conditions.

3.2.4 Further Optimization of PAL-BERT Model Performance

ReLU is one of the most widely used activation functions. ReLU is computationally efficient and helps alleviate the vanishing gradient problem, making it suitable for training deep neural networks. It is defined as Eq. (8):

$$F(x) = max(0, x) \tag{8}$$

However, the ReLU function still exhibits several drawbacks. Firstly, the output of ReLU is not zero-centered. Secondly, it suffers from the Dead ReLU Problem, where ReLU neurons become inactive in the negative input region, reducing their responsiveness during training. When $x < 0$, the gradient remains permanently at 0, causing the affected neuron and subsequent neurons to remain unresponsive, and the corresponding parameters are never updated.

In this section, we propose the adoption of the Mish activation function in place of ReLU within the fully connected layers to enhance the model's performance. Mish offers several advantages over ReLU, making it a promising choice. It lacks an upper limit on positive values, avoiding issues related to saturation and maintaining a smoother gradient. These characteristics contribute to improved gradient flow, mitigate dead neuron problems, and ultimately result in enhanced accuracy and generalization within deep neural networks when compared to ReLU.

The image of Mish function is shown in Fig. 3, and the expression is shown in Eq. (9):

$$F(x) = x * tanh\left(\ln\left(1 + e^{x}\right)\right) \tag{9}$$

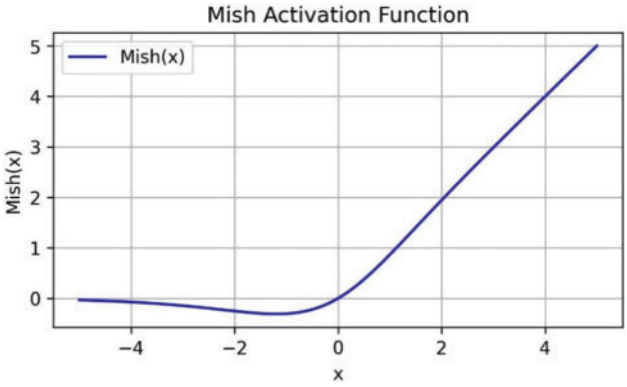

Figure 3: Mish function

Compared to ReLU, Mish allows positive values to reach higher levels without encountering an upper boundary, thus preventing saturation issues associated with limit constraints [22–25]. Unlike ReLU's hard zero boundary, Mish permits function values to be slightly less than zero, facilitating more favorable gradient flow. The most significant advantage of the Mish function over ReLU lies in its inherent smoothness. This smoothness enables better information propagation throughout the neural network when serving as the activation function, leading to notable improvements in accuracy and generalization capabilities.

4 Experiment and Results

4.1 Experimental Environment and Evaluation Criteria

The specific experimental parameters are shown in Table 4.

Table 4: Details of the experimental environment

Experimental environment	Configuration details
CPU	Intel(R) Core(TM) i7-9750H CPU @ 2.60 GHz 2.59 GHz
Memory	16 G
Graphics card	GeForce RTX 2060 8 G
Operating system	Ubuntu18.04
Programing language	Python3.7.5
Deep learning framework	Pytorch1.3

When rating the classification effect, the commonly used Accuracy, Recall and F1 value are used.

Accuracy indicates the proportion of all correct predictions (positive and negative). Recall indicates the proportion of accurately predicted positives to the total number of actual positives. F1 value is the arithmetic mean divided by the geometric mean.

The ALBERT model used in the experiment in this section is ALBERT_large_zh [17]. Its initial model size is 64 M. The specific parameter settings are shown in Table 5. When training, batch is set to 32, the gradient cumulative number is 2, and the learning rate is 2.0e−5. The training rounds number is 5, the model is optimized by AdamW.

Table 5: Parameters of ALBERT_large_zh

Dropout_prob	0.0
Hidden_act	"GELU"
Hidden_size	1024
Embedding_size	128
Initializer_range	0.02
Intermediate	4096
Max_position_embeddings	512
Num_attention_heads	16
Num_hidden_layers	24
Type_vocab_size	2
Vocab_size	21128

4.2 Effects of Different Batch Sizes and Mixing Accuracy on Training Results

Although layer normalization is used in the internal structure of ALBERT. But in fact, the size of the batch will still affect the model's accuracy. Table 6 shows the performance of PAL-BERT model accuracy when using different batch sizes and whether half-precision training is used.

Table 6: Effects of different batch and semi precision training on accuracy

	Batch = 8	Batch = 16	Batch = 32
FP16	0.759	0.787	0.816
FP32	0.762	0.789	——

Among these, 'FP16' indicates the utilization of mixed-precision computing, while 'FP32' signifies not using mixed-precision computing. As shown in the table, different batch sizes of 8, 16, and 32 lead to varying final accuracy results. The best performance is achieved with a batch size of 32. One possible explanation for this is the gradient update stage. With a batch size of 32, the average loss of 32 samples serves as the loss function, and the gradients of this average loss are used for parameter updates. This suggests that if the batch size is too small, it may get stuck in local optima, causing results to oscillate without converging to the global optimum. Therefore, for PAL-BERT training, larger batch size is preferable, albeit it demands higher hardware capabilities.

In the table, when the batch size is set to 32 without using mixed-precision computing, data loss occurs due to GPU memory overflow, rendering computations impossible. However, mixed-precision computing resolves this issue. Experimental results indicate that adopting mixed-precision training significantly enhances GPU training capacity. It is worth noting that, despite NVIDIA's official statement that its Apex mixed-precision training method does not affect model performance, during

PAL-BERT training, there is a slight reduction in final accuracy when using mixed-precision training. Nevertheless, the increase in the trainable batch size resulting from this method more than compensates for its minor adverse effects. Therefore, the use of mixed-precision training remains essential during training.

4.3 Effects of Gradient Accumulation Times on Results

The actual effect of the gradient accumulation method can be seen as a disguised increase in the size of the batch. The calculation results in the table are the conclusions obtained when the gradient accumulation times are 2. When batch = 32, the calculation results of accuracy for different gradient accumulation times are shown in Table 7.

Table 7: Effect of gradient accumulation times on results

Gradient accumulation times	Accuracy
1	0.794
2	**0.815**
3	0.816
4	0.813

As shown in Fig. 4, when the gradient accumulation time is 1, which means, without accumulation, the accuracy is slightly higher than that of batch = 16, and the gradient accumulation for two times is slightly higher, which shows that although the gradient accumulation can be regarded as the increase in the batch size, the performance is slightly worse than that of the actual batch size. The results show that the best effect occurs when the gradient is accumulated 2 or 3 times. However, considering the great increase in calculation time when accumulating 3 times, it is better to accumulate 2 times.

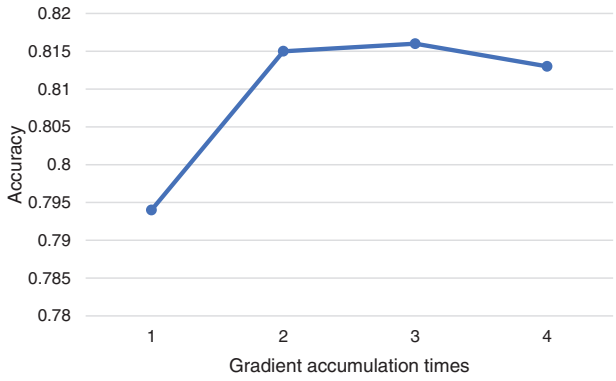

Figure 4: Effect of gradient accumulation times on accuracy

4.4 Effects of Different Training Rates on PAL-BERT Model

Table 8 shows the effects of different learning rates.

When the initial learning rate is high, the decline rate should be relatively low. Because the deep model can learn less, it needs a relatively low learning rate for fitting. Through comparison, it can be

found that the trained model has the highest accuracy when the attenuation coefficient is 0.95 and the learning rate is $2.0e^{-5}$.

Table 8: Effects of different learning rates on results

Learning rate	Attenuation coefficient ε	Accuracy
$2.0e^{-5}$	1.00	0.813
$2.0e^{-5}$	**0.95**	**0.815**
$2.0e^{-5}$	0.90	0.802
$2.0e^{-5}$	0.85	0.783
$2.5e^{-5}$	1.00	0.775
$2.5e^{-5}$	0.95	0.781
$2.5e^{-5}$	0.90	0.792
$2.5e^{-5}$	0.85	0.778

4.5 Performance Comparison of ALBERT with Other Models

To compare PAL-BERT's performance with other models, this section introduces two additional commonly used models: TextCNN [21] and bidirectional long short-term memory networks (BiLSTM) [26,27]. In this experiment, the Jieba word segmentation tool is employed, and open-source 300-dimensional Glove and Word2Vec word vectors are utilized for vectorization.

Table 9 displays the performance impact of question-answering models based on ALBERT and traditional deep learning on the CMRC 2018 dataset.

Table 9: Performance of each model on CMRC 2018 dataset

Model	Accuracy	Recall	F1
PAL-BERT	0.815	0.819	0.796
TextCNN-G	0.768	0.781	0.762
BiLSTM-G	0.776	0.784	0.774
TextCNN-W	0.763	0.798	0.752
BiLSTM-W	0.767	0.770	0.761

TextCNN-G and BiLSTM-W denote the usage of GloVe as the word vector for vectorization, while TextCNN-W and BiLSTM-W indicate the utilization of Word2Vec for vectorization.

For the four models using the combination of traditional neural network and word vector, BiLSTM-G has the best effect, but its result is still about 4% worse than PAL-BERT. It shows that PAL-BERT can perform QA tasks better than the traditional model.

5 Discussion

The NLP model based on pre-training technology represented by the BERT model has led to the emergence of a number of models, including 3 directions: 1) Larger capacity, more training data and parameter models, such as RoBERTa [28], GPT-3 [29], Megatron [30], TuringNLG [31], etc.; 2)

Models that combine more and richer external knowledge, such as ERNIE [32], K-BERT [33], etc.; 3) Models Pursuing faster speed and less memory, such as DistillBERT [34], TinyBERT [35], ALBERT, etc. When it comes to the BERT, the parameter's amount touch to more than 100 M at every turn consumes a lot of computing power and time. However, the ultimate objective of AI should be to develop simpler models capable of delivering superior results with minimal data. Human learning, for instance, does not require an abundance of samples; it relies on the ability to make inferences—a hallmark of genuine intelligence. Therefore, the pursuit of faster speeds and less memory will be the way of the future, allowing models to run not only on mobile platforms, but even on IoT devices with low-power chips.

This study designs a first order pruning model PAL-BERT based on ALBERT and explores the impact of different parameter adjustment strategies on model performance through experiments. The original ALBERT_large_zh model had a size of 64 M, but after gradient descent, 16-bit training, and first-order pruning, the model size was reduced to only 19 M. While sacrificing some accuracy, with the F1 score dropping from 0.878 to 0.796, this reduction in model size enabled it to be deployed and trained effectively on low-computing platforms.

A comparative experiment with traditional deep learning models TextCNN and BiLSTM was designed. The results demonstrate the feasibility of the method proposed in this article and provide a solution for the knowledge question and answer model on a low computing power platform. Although PAL-BERT has shown superior performance in experiments, it may still have some potential shortcomings or limitations compared to traditional models.

Pruning methods involve the choice of some hyperparameters. For PAL-BERT, these hyperparameters may require complex tuning depending on the task and the dataset. PAL-BERT's pruning process aims to reduce the size of the model, but this is often accompanied by a certain degree of information loss. Compared with traditional models, PAL-BERT may require a trade-off between model efficiency and information retention. The performance improvement of PAL-BERT may be more dependent on the nature of the specific task and the quality of the fine-tuning data set. This may result in PAL-BERT's generalization being not as good as traditional models.

However, the methods proposed in this paper have certain limitations. In our training optimization approach, we have opted for gradient descent and 16-bit precision training to address the issue of insufficient computing power. Nevertheless, in the actual training process, the performance of gradient descent is highly sensitive to the choice of the learning rate. Setting it too high may lead to divergence, while setting it too low may result in slow convergence or getting trapped in local minima. Therefore, determining the optimal learning rate for a given dataset remains an unresolved challenge. Furthermore, for 16-bit precision training, the increase in computational efficiency comes at the cost of reduced precision. The impact of this precision reduction may become evident in specific training tasks, and its practicality in models with more complex word embedding still requires further investigation.

6 Conclusion

This paper studies the question answering technology based on the pre-training model, and proposes a pruning model compression method, which successfully shortens the training time and improves the training efficiency. We have improved and optimized the Albert structure in the article. When adapting to specific Q&A tasks, we propose a new model that can give full play to its performance in Q&A tasks.

Firstly, this paper introduces an improved model of ALBERT based on BERT. Its improvement on the BERT model mainly includes the following three aspects: embedded layer decomposition, cross-layer parameter sharing, and SOP sentence order prediction task.

Secondly, two optimization strategies for model training are proposed, namely gradient accumulation and 16-bit precision training.

Thirdly, this paper proposes a first-order derivative pruning model, PAL-BERT for ALBERT. Through the comparative experiment with the traditional deep learning models TextCNN and BiLSTM, this paper explores the impact of different batch sizes and mixing accuracy on the model's performance. The experimental results show that the effect is the best when the gradient is accumulated 2 or 3 times, and there is little difference between them. Given the significant increase in computation time when accumulating three times, it is advisable to limit the accumulation to two times.; In addition, by comparing the effects of different training rates on the PAL-BERT model, it can be found that the trained model has the highest accuracy when the attenuation coefficient is 0.95 and the learning rate is $2.0e^{-5}$. For the four models using the combination of traditional neural network and word vector, BiLSTM-G has the best effect, but its result is still about 4% worse than PAL-BERT. It shows that ALBERT significantly improves the performance of natural language processing tasks compared with the traditional deep learning model.

Acknowledgement: Not applicable.

Funding Statement: Supported by Sichuan Science and Technology Program (2021YFQ0003, 2023YFSY0026, 2023YFH0004).

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Wenfeng Zheng; data collection: Siyu Lu, Ruiyang Wang; software: Zhuohang Cai; analysis and interpretation of results: Wenfeng Zheng, Lei Wang; draft manuscript preparation: Wenfeng Zheng, Siyu Lu, Lirong Yin. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: SQuAD can be obtained from: https://rajpurkar.github.io/SQuAD-explorer/. CMRC can be obtained from: https://ymcui.com/cmrc2018/.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Aithal, S. G., Rao, A. B., Singh, S. (2021). Automatic question-answer pairs generation and question similarity mechanism in question answering system. *Applied Intelligence, 51(11),* 8484–8497. https://doi.org/10.1007/s10489-021-02348-9
2. Rogers, A., Gardner, M., Augenstein, I. (2023). QA dataset explosion: A taxonomy of NLP resources for question answering and reading comprehension. *ACM Computing Surveys, 55,* 197. https://doi.org/10.1145/3560260
3. Chen, H., Geng, L., Zhao, H., Zhao, C., Liu, A. (2022). Image recognition algorithm based on artificial intelligence. *Neural Computing and Applications, 34(9),* 6661–6672. https://doi.org/10.1007/s00521-021-06058-8

4. Ma, W., Tu, X., Luo, B., Wang, G. (2022). Semantic clustering based deduction learning for image recognition and classification. *Pattern Recognition, 124,* 108440. https://doi.org/10.1016/j.patcog.2021.108440
5. Zhao, A., Yu, Y. (2021). Knowledge-enabled BERT for aspect-based sentiment analysis. *Knowledge-Based Systems, 227,* 107220. https://doi.org/10.1016/j.knosys.2021.107220
6. Wang, Y., Cui, L., Zhang, Y. (2021). Improving skip-gram embeddings using BERT. *IEEE/ACM Transactions on Audio, Speech, and Language Processing, 29,* 1318–1328. https://doi.org/10.1109/TASLP.2021.3065201
7. Yu, X., Tang, L., Rao, Y., Huang, T., Zhou, J. et al. (2022). Point-BERT: Pre-training 3D point cloud transformers with masked point modeling. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19313–19322. New Orleans, LA.
8. Guven, Z. A., Unalir, M. O. (2022). Natural language based analysis of SQuAD: An analytical approach for BERT. *Expert Systems with Applications, 195,* 116592. https://doi.org/10.1016/j.eswa.2022.116592
9. Dwivedi, Y. K., Kshetri, N., Hughes, L., Slade, E. L., Jeyaraj, A. et al. (2023). So what if ChatGPT wrote it? Multidisciplinary perspectives on opportunities, challenges and implications of generative conversational AI for research, practice and policy. *International Journal of Information Management, 71,* 102642. https://doi.org/10.1016/j.ijinfomgt.2023.102642
10. Guo, J., Zhang, W., Ouyang, W., Xu, D. (2021). Model compression using progressive channel pruning. *IEEE Transactions on Circuits and Systems for Video Technology, 31(3),* 1114–1124. https://doi.org/10.1109/TCSVT.2020.2996231
11. Vadera, S., Ameen, S. (2022). Methods for pruning deep neural networks. *IEEE Access, 10,* 63280–63300. https://doi.org/10.1109/ACCESS.2022.3182659
12. Wimmer, P., Mehnert, J., Condurache, A. (2022). Interspace pruning: Using adaptive filter representations to improve training of sparse CNNs. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12527–12537. New Orleans, LA.
13. Fang, G., Ma, X., Song, M., Mi, M. B., Wang, X. (2023). DepGraph: Towards any structural pruning. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16091–16101. Vancouver, CANADA.
14. Poyatos, J., Molina, D., Martinez, A. D., Del Ser, J., Herrera, F. (2023). EvoPruneDeepTL: An evolutionary pruning model for transfer learning based deep neural networks. *Neural Networks, 158,* 59–82. https://doi.org/10.1016/j.neunet.2022.10.011
15. Wang, X., Zheng, Z., He, Y., Yan, F., Zeng, Z. et al. (2023). Progressive local filter pruning for image retrieval acceleration. *IEEE Transactions on Multimedia, 25*, 9597–9607. https://doi.org/10.1109/TMM.2023.3256092
16. Cheng, Y., Lin, X., Zhu, H., Wu, J., Shi, H. et al. (2023). A novel hierarchical structural pruning-multiscale feature fusion residual network for intelligent fault diagnosis. *Mechanism and Machine Theory, 184,* 105292. https://doi.org/10.1016/j.mechmachtheory.2023.105292
17. Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P. et al. (2019). ALBERT: A lite bert for self-supervised learning of language representations.. arXiv:1909.11942. https://doi.org/10.48550/arXiv.1909.11942
18. Cui, Y., Liu, T., Che, W., Xiao, L., Chen, Z. et al. (2019). A span-extraction dataset for Chinese machine reading comprehension. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 5883–5889. Hong Kong, China, Association for Computational Linguistics.
19. de Silva, H., Tan, H., Ho, N. M., Gustafson, J. L., Wong, W. F. (2023). Towards a better 16-bit number representation for training neural networks. *Proceedings of the Next Generation Arithmetic*, pp. 114–133. Cham.

20. Liu, C., Zhang, X., Zhang, R., Li, L., Zhou, S. et al. (2022). Rethinking the importance of quantization bias, toward full low-bit training. *IEEE Transactions on Image Processing, 31,* 7006–7019. https://doi.org/10.1109/TIP.2022.3216776
21. Vieira, J. P. A., Moura, R. S. (2017). An analysis of convolutional neural networks for sentence classification. *2017 XLIII Latin American Computer Conference (CLEI)*, pp. 1–5. Cordoba, Argentina. https://doi.org/10.1109/CLEI.2017.8226381
22. Zhang, Z. H., Yang, Z., Sun, Y., Wu, Y. F., Xing, Y. D. (2019). LENET-5 convolution neural network with mish activation function and fixed memory step gradient descent method. *Proceedings of the 2019 16th International Computer Conference on Wavelet Active Media Technology and Information Processing*, pp. 196–199. Chengdu, China.
23. Guo, C., Qiu, Y., Leng, J., Zhang, C., Cao, Y. et al. (2022). Nesting forward automatic differentiation for memory-efficient deep neural network training. *Proceedings of the 2022 IEEE 40th International Conference on Computer Design (ICCD)*, pp. 738–745. Olympic Valley, CA, USA.
24. Xu, S., Gao, X., Wang, Y. (2021). A study on the EDA-based classification of news text. *Proceedings of the Journal of Physics: Conference Series*, 012080. Guangzhou, China.
25. He, C., Tong, Q., Yang, X., Wang, J., Zhu, T. (2021). A multi-layer feature parallel processing method for image captioning. *Proceedings of the 2021 3rd International Conference on Natural Language Processing (ICNLP)*, pp. 255–261. Beijing, China.
26. Kiperwasser, E., Goldberg, Y. (2016). Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics, 4,* 313–327.
27. Wang, H., Zhang, Y., Liang, J., Liu, L. (2023). DAFA-BiLSTM: Deep autoregression feature augmented bidirectional LSTM network for time series prediction. *Neural Networks, 157,* 240–256. https://doi.org/10.1016/j.neunet.2022.10.009
28. Cortiz, D. (2022). Exploring transformers models for emotion recognition: A comparision of BERT, DistilBERT, RoBERTa, XLNET and ELECTRA. *Proceedings of the 2022 3rd International Conference on Control, Robotics and Intelligent System (CCRIS'22)*, Association for Computing Machinery. https://doi.org/10.1145/3562007.3562051
29. Dale, R. (2021). GPT-3: What's it good for? *Natural Language Engineering, 27(1),* 113–118. https://doi.org/10.1017/S1351324920000601
30. Smith, S., Patwary, M., Norick, B., LeGresley, P., Rajbhandari, S. et al. (2022). Using deepspeed and megatron to train megatron-turing NLG 530B, a large-scale generative language model. ArXiv preprint ArXiv:2201.11990. https://doi.org/10.48550/arXiv.2201.11990
31. Saniee, I., Zhang, L., Magnetta, B. (2022). Truncated lottery ticket for deep pruning. *2022 IEEE International Conference on Image Processing (ICIP)*, pp. 606–610. Bordeaux, France. https://doi.org/10.1109/ICIP46576.2022.9897767
32. Feng, Z., Zhang, Z., Yu, X., Fang, Y., Li, L. et al. (2023). ERNIE-ViLG 2.0: Improving text-to-image diffusion model with knowledge-enhanced mixture-of-denoising-experts. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10135–10145. Vancouver, Canada.
33. Yin, J. (2022). Research on question answering system based on BERT model. *2022 3rd International Conference on Computer Vision, Image and Deep Learning & International Conference on Computer Engineering and Applications (CVIDL & ICCEA)*, pp. 68–71. Changchun, China. https://doi.org/10.1109/CVIDLICCEA56201.2022.9824408
34. Sanh, V., Debut, L., Chaumond, J., Wolf, T. (2019). DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. https://doi.org/10.48550/arXiv.1910.01108
35. Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X. et al. (2020). TinyBERT: Distilling bert for natural language understanding. *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4163–4174. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.findings-emnlp.372