Adaptive Token Selection in LLMs: Integrating Forward and Backward Dropping Techniques

Cao Xiaochuan(21091569)

Big Data Technology
xcaoas@connect.ust.hk

Abstract

This study explores a new approach to applying Token Dropping techniques in large-scale language models. We propose innovative strategies that combine forward and backward Token Dropping, aiming to improve the computational efficiency and performance of the model. Through experiments on the gsm8k and minerva math datasets, we compare different dropping strategies, including randomized, attention-score-based, and mixed-methods forward dropping, as well as fixed, dynamic, and standard deviation-based backward dropping.The results show that, although the dropping methods increase the training time, on some tasks significantly improve model performance, especially the attention plus dynamic dropping strategy. This study provides new insights and directions for the optimization of large-scale language models.

1 Introduction

With the rapid development of large-scale language models (LLMs) in the field of natural language processing, their computational efficiency and performance optimization have become a hot research topic.Token Dropping, as an emerging optimization technique, has the potential to significantly improve the efficiency and generalization ability of the model by selectively dropping part of the tokens in the input sequences during the training process.

The core idea of this study is to select a portion of tokens from the input sequence to be dropped in each training step(Hou et al., 2023). The percentage to be dropped can be adjusted according to the task requirements and model performance.The Token Dropping technique is expected to enhance the robustness of the model, reduce overfitting, and improve the generalization ability.

Our research innovatively combines forward dropping(Gu et al., 2022)(Zhang et al., 2023) and backward dropping(Hou et al., 2023) methods. Forward dropping is implemented in the forward propagation phase of the model, while backward dropping performs token selection based on loss information in the backward propagation phase. With this combination, we aim to fully utilize the advantages of both approaches to achieve a better performance and efficiency balance.

The main contributions of this paper include:

1. three forward dropping methods are proposed and implemented: random dropping, attention score-based dropping, and hybrid dropping.

2. three backward dropping strategies are developed: fixed dropping, dynamic dropping, and standard deviation-based dropping.

3. the effects of seven different combinations of forward and backward dropping in terms of training time and model performance were comparatively analyzed.

4. a comprehensive experimental evaluation on the gsm8k and minerva math datasets was conducted, providing valuable insights for future research.

Token Dropping technique originated from the exploration of neural network optimization methods. Early research focused on regularization techniques such as dropout(Srivastava et al., 2014) , which prevent overfitting by randomly dropping neurons. In recent years, researchers have begun to apply similar ideas to the input token level to optimize the performance and efficiency of large language models.

In the field of natural language processing, several studies have explored token-level optimization techniques. For example, (Goyal et al., 2017) proposed an adaptive input representation approach to improve model efficiency by dynamically adjusting the representation of input tokens.(Wang et al., 2020) investigated an importance-based token pruning approach to reduce computational complexity by identifying and retaining key tokens.

However, most existing studies either focus on token selection in the forward propagation phase or only consider loss-based backward optimization. Our work fills this research gap by combining forward and backward dropping methods, providing a more comprehensive framework for Token Dropping techniques.

2 Methodology

This study presents an innovative Token Dropping framework that combines forward and backward dropping strategies. This approach aims to optimize the training process of large language models by selectively dropping certain tokens while maintaining or improving model performance.

2.1 Forward dropping

Forward Token Dropping is implemented in the forward propagation phase of the model to reduce the amount of data to be processed by selectively dropping certain tokens in the input sequence. We implement three different forward dropping methods.

The token dropping is achieved by setting the mask of the token to 0, and it will revert to the original mask after training.

2.1.1 Random dropping

This method uses a random sampling strategy to randomly select a certain percentage of tokens from the input sequence for retention. Its mathematical expression can be summarized as follows:

$I_{keep} = RandomSample(\{1, ..., N\}, [N(1 - r)])$

Where N is the length of the input sequence,r is the set of indexes of the token.The advantage of this method is that it is simple to implement and has low computational overhead, but it may lead to the loss of important information.

2.1.2 Attention-based dropping

The second method is the attention score-based dropping method, which selects tokens to keep based on their attention scores. This method selects tokens to be retained based solely on their attention scores, dropping tokens with low attention scores, tokens that do not have much impact on training, and retaining tokens with large attention scores.This method prioritizes the retention of tokens with high attention scores, which usually contain more semantic information and contextual associations. Therefore, while reducing the number of tokens, it can maximize the retention of key information in the input sequence.Its mathematical expression can be summarized as follows:

$A = mean(attention_{values}, dim = (1, 2))$

$I_{keep} = TopK(A, [N(1 - r)], largest = True)$

where A is the average attention score and the TopK function selects $[N(1 - r)]$ tokens with the highest scores. However, if the model overly relies on tokens with high attention scores, it may lead to overfitting of the training data, thus affecting the generalization ability of the model.

2.1.3 hybrid dropping

So we propose a third method: the hybrid dropping method. It combines random sampling and an attention score based strategy. The method first randomly drop a small portion of tokens, and then selects the remaining tokens to be dropped based on the attention score.Its mathematical expression can be summarized as follows:

$I_{random} = RandomSample(\{1, ..., N\}, [Nr_{random}])$

$I_{attention} = TopK(A, [Nr_{attention}], smallest = True)$

$I_{keep} = \{1, ..., N\}\ (I_{random} \cup I_{attention})$

where r_{random} and $r_{attention}$ denote the proportion of random dropping and attention-based dropping. This method takes into account the importance of tokens while maintaining a certain degree of randomness.The introduction of random dropping reduces the potential bias that may be brought about by relying exclusively on the attention score. This helps to prevent the model from focusing too much on certain specific patterns or features.

2.2 Backward dropping

Backward dropping is usually implemented in the backward propagation phase of the model. The backward dropping we use is based on loss difference. The core idea is to adjust the dropping strategy by evaluating the impact of each token on the model prediction. Specifically, we define three dropping strategies: Fixed Dropping, Dynamic Dropping, and Standard Deviation Dropping. These strategies are based on different criteria to determine which token should be dropped.

Before training, we calculate the reference loss of the reference model. drop token is also specified by setting the mask to 0.

2.2.1 Fixed dropping

The first one is the fix strategy, which drops the token in the input sequence by a pre-set drop rate. Specifically, the fixed drop strategy first calculates the loss difference (loss difference) of each token, i.e., the difference between the loss of the current model and the loss of the reference model, denoted as:

$\delta L = L_{current} - L_{reference}$

where $L_{current}$ and $L_{reference}$ denote the losses of the current model and the reference model to the input sequence, respectively. To achieve fixed drop, we generate the drop mask using the following formula:

$DropMaskfixed = TopK(\delta L, k_{drops})$

where Topk() denotes the selection of k_{drop} token in $\delta L$with the smallest loss variance. The drop ratio k_{drop} is determined by the user-specified drop rate:

$k_{drop} = [drop_{rate}N]$

where N is the length of the input sequence. With this approach, the fixed drop strategy ensures that the same number of tokens are dropped for each training, thus providing a stable training mechanism..

2.2.2 Dynamic dropping

The dynamic drop strategy adapts to the complexity of the input sequence and the model performance by dynamically adjusting the number of token to be dropped. Specifically, the dynamic drop strategy first generates an initial drop mask based on a predefined threshold :

$DropMaskdynamic = I(\delta L < \theta)$

where I() is the indicator function and θ is a predefined threshold. Based on the experimental results, we set $\theta = 0.1$. If the drop rate exceeds the user-specified drop rate, the dynamic drop policy will further adjust the drop mask to ensure that the number of token dropped does not exceed the maximum allowable value. The specific adjustment process is as follows:

$DropMaskdynamic = LimitDrop(DropMaskdynamic, k_{drop})$

where $LimitDrop()$ means to limit the percentage of drops in the drop mask to k_{drop}

2.2.3 Std-based dropping

The standard deviation drop strategy determines the token to be dropped based on the standard deviation of the loss variance. Specifically, the standard deviation drop strategy first calculates the mean μ and standard deviation σ of the loss variance:

$\mu = \frac{1}{N}\sum \delta L_i$

$\sigma = \sqrt{\frac{1}{N-1}\sum(\mu - \delta L_i)^2}$

Then, a preliminary drop mask is generated by the following equation:

$DropMaskstd = I(\delta L < \mu - c \cdot \sigma)$

where c is a constant to control the aggressiveness of the drop. Based on the experimental results, we set c=1.5. Similar to the dynamic drop strategy, if the percentage of drop exceeds the user-specified drop rate, the standard deviation drop strategy also further adjusts the drop mask to ensure that the number of token dropped does not exceed the maximum allowed value:

$DropMaskstd = LimitDrop(DropMaskstd, k_{drop})$

The standard deviation method identifies token whose loss variance is significantly lower than the mean, which may be noisy or unimportant token.

The standard deviation method utilizes the statistical distributional features of the loss variance to ensure that the drop strategy is statistically significant and reliable. Moreover, methods based on statistical distributions usually have high stability.

3 Experiment

To evaluate the effectiveness of our proposed Token Dropping method, we conducted a series of experiments. For the pre-training model we used TinyLlama-v1.1, and for the reference model we used TinyLlamaRef, with the drop-rate set to 0.3, meaning that thirty percent of the tokens are dropped in forward dropping.The fine-tuned dataset was created using small-open-web-math-v2. The evaluation methodology uses the math-evaluation-harness tool, which is a collection of tools specifically designed to evaluate mathematical problem solving skills. It contains math problems of various levels of difficulty and types to fully test the performance of the model.PROMPT-TYPE is set to deepseek-math.

3.1 Time cost

We tested the time for model fine-tunning, which would not include the time to compute the reference model, since computing the reference model loss is consistent.

The test result is as Table 1.

The introduction of the Token Dropping strategy had a significant impact on the training time

of the model. The training time for the baseline model was 31 minutes, while all dropping methods resulted in an increase in training time ranging from 39 to 45 minutes. The random plus fixed dropping strategy added the least amount of time, only 39 minutes, 8 minutes more than the baseline. In contrast, the attention plus random (20 %) plus standard deviation dropping strategy required the longest training time at 45 minutes, nearly 50% more time than the baseline. Attention-plus-fixed dropping and attention-plus-dynamic dropping required 41 and 42 minutes, respectively, showing the additional computational overhead associated with the attention-based approach. Interestingly, hybrid strategies that include a random component (e.g., attention plus random (20%) plus fixed dropping and attention plus random (20%) plus dynamic dropping) both take 44 minutes, suggesting that the introduction of randomness may have increased computational complexity. These results emphasize the need to weigh computational efficiency against potential performance gains when implementing Token Dropping.

3.2 Accuracy

We also tested the accuracy of the fine-tuned model as well as the baseline model,as Table 2shows:

The impact of the Token Dropping strategy on model performance varies from task to task, showing interesting patterns. On the gsm8k dataset, the baseline model performs better, reaching a performance of 0.8. However, most of the dropping strategies resulted in performance decreasing or remaining constant on gsm8k. The only exception is attention plus fixation dropping, which maintains the same 0.8 performance as the baseline. In contrast, on the minerva-math dataset, almost all dropping strategies resulted in significant performance gains. The baseline model achieves only 0.2 performance on minerva-math, whereas attention plus dynamic dropping and attention plus random (20%) plus standard deviation dropping boost performance to 1.0 and 1.2, respectively, which is a huge improvement. Of particular note, Attention plus Dynamic dropping performs well on both datasets, with an average performance of 0.85, the highest of all methods. These results suggest that the Token Dropping strategy is particularly effective when dealing with complex mathematical reasoning tasks (e.g., minerva-math), but may have a negative impact on relatively simple tasks (e.g., gsm8k).

3.3 Conclusion

Token Dropping techniques exhibit a complex trade-off between improving model performance and computational efficiency. While all dropping strategies increase training time, the performance gains they bring on specific tasks may make the tradeoff worthwhile. In particular, the dropping strategies show significant advantages when dealing with complex mathematical reasoning tasks such as minerva-math. The attention-plus-dynamic dropping strategy performs best in balancing performance and training time, achieving good results on both datasets while adding only a modest amount of training time.

These findings reveal several important insights: first, the effectiveness of Token Dropping is highly dependent on the nature of the task and may be more valuable on complex tasks. Second, strategies that combine an attention mechanism with a dynamic or stochastic component seem to strike a better balance between performance and efficiency. Third, despite the increase in training time, the significant increase in performance may justify this additional computational cost in some cases.

4 Limitation and future work

The current dropping method increases training time, which is contrary to our original goal of improving efficiency. Future work should focus on how to reduce the extra computational overhead while maintaining the performance improvement. Second, I believe that we should reduce the number of token drops in the early training phase, and then gradually increase the number of drops after the training has stabilized. However, the implementation of this approach in forward dropping encounters some technical challenges and requires modifications to the model architecture. Finally, we need to further investigate the impact of Token Dropping on the long-term training and fine-tuning process of the model.

5 Conclusion

This study explores new methods for applying Token Dropping techniques to large-scale language models, and by combining forward and backward dropping strategies, we achieve significant performance gains on certain tasks. Although these methods increase training time, they provide new ideas for improving model efficiency and performance.

Table 1: Training time comparison of different dropping methods

Dropping method	Training Time
Baseline	31 min
Random + fixed dropping	39 min
Attention + fixed dropping	41 min
Attention, random(20%) + fixed dropping	44 min
Attention + dynamic dropping	42 min
Attention + std dropping	44 min
Attention, random(20%) + dynamic dropping	44 min
Attention, random(20%) + std dropping	45 min

Table 2: Performance comparison of different dropping methods

Dropping method	gsm8k	minerva_math	avg
Baseline	0.8	0.2	0.5
Random + fixed dropping	0.5	0.4	0.45
Attention + fixed dropping	0.8	0.4	0.6
Attention, random(20%) + fixed dropping	0.6	0.6	0.6
Attention + dynamic dropping	0.7	1.0	0.85
Attention + std dropping	0.4	0.8	0.6
Attention, random(20%) + dynamic dropping	0.6	0.4	0.5
Attention, random(20%) + std dropping	0.5	1.2	0.8

In particular, the attention plus dynamic dropping strategy exhibits excellent overall performance.

Our work opens up new research directions for the optimization of large language models and also reveals some important challenges. Future research should be devoted to solving these challenges and further optimizing the Token Dropping technique so that it can effectively reduce the consumption of computational resources while improving the performance of the model.

References

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. 2017. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*.

Linxi Gu, Weitong Zhang, Zejian Liu, Peng Xu, Qi Meng, Xiang Li, Yuxuan Liang, Yabo Ling, Yijiang Liu, Shuguang Liu, Zhi-Ming Ma, Jianmin Wang, and Mingsheng Long. 2022. Token dropping for efficient bert pretraining. *arXiv preprint arXiv:2203.13240*.

Lu Hou, Jinhua Zhu, James Kwok, Fei Huang, Qifeng Chen, and Ping Luo. 2023. Rho-1: Not all tokens are what you need. *arXiv preprint arXiv:2310.18780*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

Weitong Zhang, Linxi Gu, Qi Meng, Yabo Ling, Yijiang Liu, Shuguang Liu, Jianmin Wang, and Mingsheng Long. 2023. Revisiting token dropping strategy in efficient bert pretraining. *arXiv preprint arXiv:2305.10549*.