Global-Pruner: A Stable and Efficient Pruner for Retraining-Free Pruning of Encoder-Based Language Models

Guangzhen Yao[1], **Yuehan Wang**[2],[*] **Hui Xu**[3][†], **Long Zhang**[1], **Miao Qi**[1][‡]
[1]School of Information Science and Technology, Northeast Normal University, China
[2]School of Teacher Education, Jiangsu University, Zhenjiang, China
[3]Institute for Intelligent Elderly Care, Changchun Humanities and Sciences College, China
{ yaoguangzhen, xuh504, longzhang, qim801}@nenu.edu.cn
yhwang0809@stmail.ujs.edu.cn

Abstract

Large language models (LLMs) have achieved significant success in complex tasks across various domains, but they come with high computational costs and inference latency issues. Pruning, as an effective method, can significantly reduce inference costs. However, current pruning algorithms for encoder-based language models often focus on locally optimal solutions, neglecting a comprehensive exploration of the global solution space. This oversight can lead to instability in the solution process, thereby affecting the overall performance of the model. To address these challenges, we propose a structured pruning algorithm named G-Pruner (Global Pruner), comprising two integral components: PPOM (Proximal Policy Optimization Mask) and CG²MT (Conjugate Gradient Squared Mask Tuning), utilizing a global optimization strategy. This strategy not only eliminates the need for retraining but also ensures the algorithm's stability and adaptability to environmental changes, effectively addressing the issue of focusing solely on immediate optima while neglecting long-term effects. This method is evaluated on the GLUE and SQuAD benchmarks using $\text{BERT}_{\text{BASE}}$ and DistilBERT models. The experimental results indicate that without any retraining, G-Pruner achieves significant accuracy improvements on the $\text{SQuAD}_{2.0}$ task with a FLOPs constraint of 60%, demonstrating a 6.02% increase in F1 score compared with baseline algorithms.

1 Introduction

In recent years, Transformer-based pre-trained language models (PLMs) Li et al. (2024); Guimarães et al. (2024); Ho et al. (2024); Xu et al. (2024); Kojima et al. (2024) have dominated the field of natural language processing (NLP) Shamshiri et al. (2024); Oyewole et al. (2024); Zheng et al. (2024); Raza et al. (2024); Mei et al. (2024) due to their outstanding performance. However, the significant advantages of PLMs come with a substantial increase in model size and high computational costs. Pruning, as an optimization technique, can effectively reduce model complexity to enhance generalization ability and operational efficiency. Pruning techniques include structured pruning He and Xiao (2023); Fang et al. (2023); Sun et al. (2020); Liu et al. (2021a); Hou et al. (2020a); Iandola et al. (2020); Kitaev et al. (2020); Xia et al. (2022) and unstructured pruning Cheng et al. (2023); Santacroce et al. (2023); Wang et al. (2020); Shi et al. (2024); Zhang et al. (2024); Dery et al. (2024) aiming to improve efficiency by eliminating redundant parts of the model. Particularly, structured pruning has become a key technology for addressing size and speed issues in encoder-based language models, systematically removing redundancies without significantly impairing model performance.

Despite this, existing pruning methods still have limitations in practical applications. For example, Kwon et al. (2022) avoided the high costs associated with retraining by employing three techniques: mask search, mask rearrangement, and mask tuning. However, this greedy-based pruning method has been proved to be effective only in the short term and faced challenges in finding global optima, particularly when applied to complex or dynamically changing tasks. Moreover, the K-Prune Park et al. (2023) algorithm aimed to minimize pruning errors and enhance accuracy by preserving knowledge from pre-trained models. However, it did not fully consider the accuracy of weight selection and the long-term stability of the pruning strategy. Similarly, the KCM Nova et al. (2023) framework could quickly compress models and minimize performance loss by accurately assessing the importance of neurons in the short term. However, it overlooked the long-term stability and adaptability of the model to complex tasks, especially under

[*]Guangzhen Yao and Yuehan Wang contributed equally.
[†]corresponding author.
[‡]corresponding author.

high FLOPs constraints. Although these pruning methods can enhance the efficiency of models in the short term, they typically have a common drawback: they primarily focus on finding local optima and neglect the exploration of global optima.

To address these issues, we introduce a new retraining-free pruning framework for Transformer models—G-Pruner, efficiently locating global optima quickly without retraining. This strategy integrates two advanced technologies: PPOM and CG²MT. In the PPOM phase, the algorithm first conducts a comprehensive mask search, then fine-tunes and optimizes the selected masks using the PPO (Proximal Policy Optimization) technique from reinforcement learning. Subsequently, in the CG²MT phase, we enhance the efficiency and stability of solving asymmetric matrix problems through an improved CGS (Conjugate Gradient Squared) solver.

The primary contributions of this study include:

- We propose a structured pruning algorithm named G-Pruner, designed to prune encoder-based language models with high precision without the need of retraining.
- We conduct a comprehensive evaluation using the GLUE and SQuAD benchmarks on $\text{BERT}_{\text{BASE}}$ and DistilBERT models to demonstrate the performance of G-Pruner. We find that our method not only outperforms frameworks that are retraining-free but also surpasses other frameworks that do require retraining at the same pruning cost.
- Under the same FLOPs constraints, G-Pruner significantly outperforms some existing pruning techniques in pruning time without sacrificing model accuracy. Even under the strict constraint of allowing a maximum accuracy reduction of no more than 1%, $\text{BERT}_{\text{BASE}}$ achieves 60-70% of the original FLOPs across all tasks.

2 Related Work

2.1 Pruning For Encoder-Based Language Model

Pruning enhances model efficiency by removing insignificant weights or components such as attention heads or layers. There are two types: unstructured and structured. Unstructured pruning reduces model size by eliminating individual parameters. For example, Sanh et al. (2020) offered a straightforward first-order weight pruning method for fine-tuning pre-trained models, significantly boosting performance while maintaining high sparsity. Second-order methods like oBERT Kurtic et al. (2022) used approximate second-order information to reduce storage and computational demands of BERT models. Structured pruning simplifies models on a larger scale by removing entire components. For example, Hardware-friendly block structure pruning techniques Li et al. (2020) improved compression ratios and speed through optimizations. FLOP Wang et al. (2019) reduced model size and enhanced training and inference speed by maintaining dense weight matrix structures rather than sparse representations. SLIP Lin et al. (2020) improved pruning efficiency through feature layer normalization and unit block identification. Sajjad et al. (2023) tackled reducing layers in pre-trained Transformer models while maintaining task-specific performance. EBERT Liu et al. (2021b) dynamically determined pruning strategies per input sample, significantly cutting computational load and memory use. DynaBERT Hou et al. (2020b) adjusted BERT model size and latency adaptively, addressing deployment challenges on edge devices with diverse hardware performance.

2.2 Pruning For Retraining-free Structured Model

Data-independent neural pruning algorithm Mussay et al. (2019) and post-training weight pruning methods for deep neural networks Lazarevich et al. (2021) aimed to effectively reduce model size while minimizing accuracy loss. In the domain of structured pruning, the concept of "neuron merging" Kim et al. (2020) and RED's data-independent structured compression technique Yvinec et al. (2021) were employed by utilizing various technologies to maintain or enhance model accuracy without incurring accuracy losses. However, these methods overlooked challenges such as thorough weight selection analysis, long-term stability, and maintaining performance under high sparsity. To effectively address these challenges, a novel post-training pruning framework named G-Pruner is introduced.

3 Background and Baseline Description

The core of the pruning problem is to find the optimal methods for masking while considering sparsity constraints. This study focuses on the compression of encoder-based language models, notably BERT$_{\text{BASE}}$ and DistilBERT. These encoder-based language models consist of two primary sub-layer archetypes: Multi-Head Attention (MHA) and Feed-Forward Network (FFN). In this section, we explain how to mask the attention heads and feed-forward networks.

3.1 Structured Pruning by Masking

The formulas for MHA and FFN are expressed as follows:

$$\mathrm{MHA}\left(x; m_l^{\mathrm{MHA}}\right) = \sum_{i=1}^{H} m_{l,i}^{\mathrm{MHA}} \circ \mathrm{Att}_i(x) \quad (1)$$

$$\mathrm{FFN}\left(x; m_l^{\mathrm{FFN}}\right) = \left(\sum_{i=1}^{N} m_{l,i}^{\mathrm{FFN}} \circ W_{:,i}^{(2)} \sigma\left(W_{i,:}^{(1)} x + b_i^{(1)}\right)\right) + b^{(2)} \quad (2)$$

where the mask variable $m_{l,i}^{\mathrm{MHA}}$ for the i^{th} attention head in the l^{th} layer is used to decide whether to retain (mask value of 1) or prune (mask value of 0) that head. The operator "$\circ$" denotes the Hadamard product (element-wise multiplication) to determine each attention head's contribution to the output.

In this paper, we have drawn on the research findings of Kwon et al. (2022) to formalize the pruning problem of encoder-based language models as a constrained optimization problem concerning a mask. The goal is to minimize the loss function $L(m)$ while ensuring that the computational cost (measured in FLOPs or latency) of the model pruned according to mask m remains within acceptable limits. Given a mask m, the optimization formula is as follows:

$$\arg\min_{m} L(m) \quad \text{s.t.} \quad Cost(m) \leq C \quad (3)$$

where $Cost(m)$ denotes the FLOPs or latency of the architecture after pruning by mask, $L(m)$ represents the loss function, and C is the given constraint on FLOPs or latency.

Within a given FLOPs constraint C, the objective is to find the optimal mask configuration m, such that the FLOPs of the pruned model are reduced and the impact on performance is minimized. The problem can be formalized as:

$$\arg\min_{m} \sum_{i \in Z(m)} I_{ii} \quad \text{s.t.} \quad F_{\text{head}} \left\| m_{\text{MHA}} \right\|_0 + F_{\text{filter}} \left\| m_{\text{FFN}} \right\|_0 \leq C \quad (4)$$

where F_{head} and F_{filter} respectively represent the FLOPs required to execute a head and a filter, while $\|m_{\text{MHA}}\|_0$ and $\|m_{\text{FFN}}\|_0$ respectively represent the number of retained heads and filters in the MHA and FFN layers.

3.2 Baseline Description

In our study, we adopt Kwon et al.'s approach as the baseline method. The framework consists of three stages: mask search, mask rearrangement, and mask tuning. During the mask search stage, the Fisher information matrix is used to identify which attention heads and filters are crucial and should be retained, and which are relatively less important and can be pruned. Following the initial steps of mask search, the mask rearrangement process relies on a greedy algorithm, which reselects the heads and filters to be pruned by analyzing interactions between layers within the model. In the final phase of mask tuning, linear least squares are used to minimize reconstruction error and optimize the values of the non-zero mask variables. Since the mask search method based on the Fisher information matrix has been widely proven effective, no further improvements are pursued in this study.

4 Methodology

4.1 Framework Overview

As illustrated in Figure 1, our framework is divided into two main stages: the PPOM module (Section 4.2) and the CG²MT module (Section 4.3). During the PPOM mask optimization phase, we utilize Fisher information to determine which attention heads and filters are crucial and should be retained, and which are relatively unimportant and can be pruned. Subsequently, with the aid of reinforcement learning, the already identified mask patterns are adjusted to better explore intra-layer interactions among mask variables to optimize model performance. Subsequently, in the CG²MT mask tuning phase, the non-zero mask variables are fine-tuned by restructuring inter-layer output signals to compensate for any potential accuracy loss caused by pruning. The framework is designed to incorporate three primary inputs: a Transformer model fine-tuned for a specific downstream task, a small-scale sample dataset (typically containing 1,000 to 2,000 examples), and a resource constraint condition.

4.2 PPOM(Proximal Policy Optimization Mask)

While Fisher information-based mask search effectively identifies key model parameters, it doesn't guarantee minimal gradient impact during early pruning stages. Thus, initial pruning results often need detailed reordering and optimization. Com-

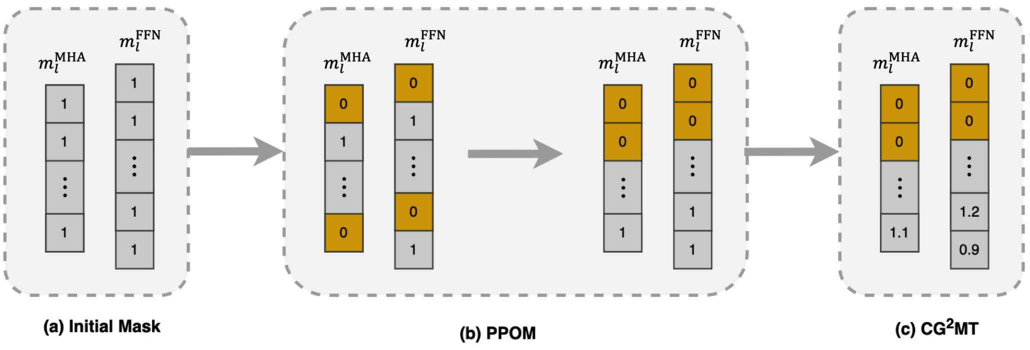

Figure 1: **Overview of the G-Pruner framework. (a) Mask variables initialized to 1. (b) PPOM (Section 4.2) and (c) CG²MT (Section 4.3).**

mon greedy algorithms attempt to reduce overall gradient impact through local optimization but may not fully optimize model performance long-term. To address this, we propose using the PPO algorithm for further mask refinement. Initially, we analyze masks derived from Fisher information and gradient data for each layer, focusing on weight matrices, pruning masks, and gradients. For layers where all elements are fully pruned or untouched, the original mask remains. For other layers, we reorder neurons or attention heads based on their impact on model performance and gradients.

4.2.1 Design Actors and Critics

In our study, we employ the Actor-Critic framework, combining the value function (Critic) and the policy function (Actor) to learn jointly. The primary task of the Actor network is to intelligently generate policies $\pi_\theta(a_t \mid s_t)$ tailored to different states s_t. It not only handles decision-making for individual states but also manages challenges posed by multidimensional and complex state spaces. In specific environmental states, the Actor network employs intricate computations to output a series of probability distributions directly linked to potential actions. Particularly when integrated closely with attention mechanisms, the Actor network can finely assess and optimize different attention heads or neurons.

During the pruning process, "state" refers to the current parameter state of the neural network, including weight matrices, pruning masks, gradients, and other information. The Actor network receives these state representations as inputs and generates a probability distribution describing the likelihood of each action (e.g., preserving or pruning a neuron). The length of the output vector equals the number of actions and can be a two-dimensional vector where each element represents the probability of a corresponding action. This probability distribution can be expressed as:

$$\pi_\theta(a_t \mid s_t) = \text{softmax}(f_\theta(s_t)) \tag{5}$$

where $f_\theta(s_t)$ denotes the output layer of the Actor network with parameters θ, predicting scores for each action a_t given state s_t. The softmax function transforms these scores into a probability distribution, ensuring that the probabilities of all actions sum to 1.

The Critic network, as a core component of the value function estimator, is primarily used to assess the expected impact of each pruning operation on the overall performance of neural networks, specifically the expected cumulative return. Based on the Critic network's output of expected cumulative return, each pruning decision is evaluated for its effectiveness. Higher expected returns indicate potential benefits to network performance, while lower returns may lead to performance degradation.

Its design aims to output the expected value $V_\omega(s_t)$ of the current state to guide policy updates in the Actor network. Specifically, the Critic network is trained by minimizing the mean squared error (MSE) between its predicted value and the actual reward:

$$L(\omega) = \mathbb{E}[(y_t - V_\omega(s_t))^2] \tag{6}$$

where ω represents the Critic network parameters, y_t is the expected cumulative reward at time step t, and $V_\omega(s_t)$ is the Critic network's output layer responsible for predicting the expected cumulative reward value given state s_t.

$$y_t = R_t + \gamma V_\omega(s_{t+1}) \tag{7}$$

where R_t denotes the reward at time step t, γ is the discount factor, and $V_\omega(s_{t+1})$ is the estimated state value function at time step $t+1$.

The evaluation results of the Critic network are used as feedback to adjust the pruning strategies generated by the Actor network. This feedback directly influences the decision-making process of the Actor network, enabling it to intelligently select pruning operations. Through continuous learning and evaluation, the Critic network dynamically adjusts pruning strategies. For instance, in each pruning iteration, based on the evaluation results of the current state, the Critic network can recommend whether to retain or prune specific layers or neurons, thereby maximizing the overall network performance. In summary, the Critic network collaborates with the Actor network to evaluate the effectiveness of its strategies and provide feedback, optimizing the pruning decision-making process of neural networks.

4.2.2 Pruning Execution

Based on the policy (probability distribution) generated by the Actor network, pruning operations are selected. These operations can be binary (retain or prune) or more complex (applying different pruning probabilities to each neuron or attention head). According to the policy outputted by the Actor network, a corresponding pruning mask M is generated to determine whether each neuron or attention head should be pruned. The process of generating the pruning mask is as follows:

$$M = \text{Bernoulli}(\pi_\theta(a_t \mid s_t)) \tag{8}$$

where $\pi_\theta(a_t \mid s_t)$ is the probability distribution outputted by the Actor network. The Bernoulli function generates a binary vector M, where each element represents the operation on the corresponding neuron or attention head (1 for retain, 0 for prune).

4.2.3 Algorithm Updates

In the pruning task, the advantage function calculates the expected gain or loss after performing pruning operations. This metric is used in the PPO algorithm to compute policy gradients, guiding the Actor network to update its policy to maximize long-term cumulative rewards. The formula for the advantage function is:

$$A(s_t, a_t) = y_t - V_\omega(s_t) \tag{9}$$

where y_t represents the expected cumulative reward after taking action a_t in state s_t, and $V_\omega(s_t)$ is the estimated state value function outputted by the Critic network, indicating the expected cumulative reward in state s_t.

In the pruning task, the PPO algorithm updates the Actor network parameters by maximizing the objective function of proximal policy optimization before and after policy updates. The primary goal of the Actor network is to generate a probability distribution for pruning decisions to optimize the performance or efficiency of the neural network. Specifically, the PPO algorithm first computes the importance sampling ratio $r_t(\theta)$ between the new and old policies:

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \tag{10}$$

where $\pi_\theta(a_t \mid s_t)$ and $\pi_{\theta_{\text{old}}}(a_t \mid s_t)$ denote the probabilities of taking action a_t under state s_t for the new and old policies, respectively.

The objective function of PPO aims to maximize the advantage function $A(s_t, a_t)$, while constraining the policy update magnitude through a clipping function $\rho_{\text{clip}}(r_t(\theta))$. The formula is as follows:

$$J^{\text{CLIP}}(\theta) = \mathbb{E}_{(s_t,a_t)\sim\pi_{\theta_{\text{old}}}}\left[\min(r_t(\theta)A(s_t,a_t), \rho_{\text{clip}}(r_t(\theta))A(s_t,a_t)\right] \tag{11}$$

By maximizing the objective function $J^{\text{CLIP}}(\theta)$, we effectively update the Actor network parameters θ to optimize pruning decision strategies.

In the PPO algorithm, Actor network parameter θ is updated using policy gradient methods with the update formula:

$$\theta \leftarrow \theta + \sigma^A \nabla_\theta J(\theta) \tag{12}$$

where σ^A is the learning rate of the Actor network.

The Critic network also updates its parameter ω to more accurately estimate the performance change of the neural network after pruning operations. The update formula for the Critic network is:

$$\omega \leftarrow \omega - \sigma^C \nabla_\omega L(\omega) \tag{13}$$

where σ^C is the learning rate of the Critic network.

In each iteration, the Actor network determines pruning probabilities for each neuron using current model gradient information, guiding network structure evolution. Simultaneously, the Critic network assesses expected model performance post-pruning, balancing exploration and efficiency. Despite initial mask imperfections, the PPO algorithm reduces reliance on single Fisher information, enhancing method effectiveness by analyzing intra-layer interactions. This adaptive approach optimizes performance iteratively throughout pruning.

4.3 CG²MT(Conjugate Gradient Squared Mask Tuning)

In the PPOM stages, to simplify the search during the model pruning process, the mask values are strictly constrained to 0 or 1. As the process advances, this restriction is gradually relaxed, the non-zero variables in the mask can be adjusted to any real number, with the objective of restoring the accuracy of the pruned model by fine-tuning the mask variables. Nonetheless, when the least squares method is used for solving, numerical instability may be encountered, especially when facing extremely unstable or ill-conditioned problems. To address such challenges, the CGS solver provides an optimization strategy for efficiently solving asymmetric matrix problems. This solver performs double the computations in each iteration and squares the residuals, which not only accelerates convergence but also enhances the stability of the algorithm.

In our framework, we utilize the CGS solver to adjust the mask variables in the pruned model to minimize the reconstruction errors between different layers. The specific operations are as follows: Starting from the first layer of the model, we use the remaining heads or filters after pruning to reconstruct the output activations of the original model. This process can be formally represented by the following mathematical formula:

$$\underset{m_l}{\operatorname{argmin}} \|x + \mathrm{layer}(x; m_l) - (x' + \mathrm{layer}(x'; 1))\|_2^2 \tag{14}$$

where x and x' are the inputs to the pruned and original model layers, respectively, and layer can be either MHA or FFN. Furthermore, we simplify this problem into a CGS solver problem, expressed by the following formula:

$$\arg\min_{m_l} \|Am_l - b\|_2^2 \tag{15}$$

where vector b represents the difference between the output activations of the two models. Matrix A represents the output activations of the heads or filters pruned by a binary mask.

Considering the large scale of matrix A, direct application of CGS solver might lead to numerical stability issues. Therefore, we reparameterize the CGS solver problem and transform it into a damped problem, enhancing the stability of the solution by fixing the damping value at 1. The formula is expressed as:

$$\arg\min_{r_l} \|Ar_l + A \cdot 1 - b\|_2^2 \tag{16}$$

where $m_l = 1 + r_l$. Additionally, to prevent the adjusted masks from negatively impacting the model's accuracy, we restrict the range of the adjusted mask variables to between $[-10, 10]$. If we find that the mask of any layer exceeds this range, we discard that layer's mask and cease further mask adjustments.

Algorithm 1 CGS Solver Iterative Mask Optimization Algorithm

```
1: Initialize: Start with an initial guess $x_0$, compute the initial residual $r_0 = b - Ax_0$, set $p_0 = r_0$, initialize step size coefficient $\alpha_0 = 0$, auxiliary variables $u_0 = 0$, $v_0 = Ap_0$, and $r_0$ is the initial direction vector for iteration.
2: Iteration step: For each iteration $k = 0, 1, 2, \ldots$ until convergence criteria are met.
3: Compute step size coefficient: $\alpha_k = \frac{r_k^T r_k}{v_k^T v_k}$
4: Update auxiliary variable: $q_{k+1} = u_k - \alpha_k v_k$
5: Update solution vector: $x_{k+1} = x_k + \alpha_k (q_{k+1} + u_k)$
6: Update residual vector: $r_{k+1} = r_k - \alpha_k (q_{k+1} + u_k + v_k)$
7: Check for convergence: If $\|r_{k+1}\|$ is small enough, stop the algorithm.
8: Calculate correction coefficient: $\beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$
9: Update another auxiliary variable: $u_{k+1} = r_{k+1} + \beta_k q_{k+1}$
10: Update direction vector: $v_{k+1} = Au_{k+1}$
```

This iterative process starts from the first layer of the neural network and progresses to the final layer, ensuring that while model parameters are reduced, performance loss is minimized as much as possible. Each iteration entails precise tuning of the mask variables, with the goal of preserving accuracy while pruning the model architecture. This intricately crafted optimization process enables us to strike a fine balance between the model's complexity and performance, guaranteeing that the pruned model maintains accuracy levels comparable to those of the original model while streamlining its structure.

Table 1: **G-pruner compares its accuracy against the baseline model under various FLOPs constraints.**

BERT$_{BASE}$																		
Method	QQP			MNLI			SST-2			QNLI			SQuAD$_{1.1}$			SQuAD$_{2.0}$		
FLOPs	60%	65%	70%	60%	65%	70%	60%	65%	70%	60%	65%	70%	60%	65%	70%	60%	65%	70%
Baseline	87.40	87.55	87.71	80.52	81.54	81.52	90.50	90.91	91.30	87.04	87.46	87.92	83.82	84.34	84.89	72.29	72.88	73.41
G-pruner	90.63	90.84	90.98	82.87	83.41	83.92	92.89	93.22	93.50	90.50	90.82	91.10	87.52	88.05	88.57	78.31	78.62	78.93
	+3.23%	+3.29%	+3.27%	+2.35%	+2.87%	+2.40%	+2.39%	+2.31%	+2.20%	+3.46%	+3.36%	+3.18%	+3.70%	+3.71%	+3.68%	+6.02%	+5.74%	+5.52%
DistilBERT																		
Method	QQP			MNLI			SST-2			QNLI			SQuAD$_{1.1}$			SQuAD$_{2.0}$		
FLOPs	60%	65%	70%	60%	65%	70%	60%	65%	70%	60%	65%	70%	60%	65%	70%	60%	65%	70%
Baseline	85.92	86.32	86.71	78.83	79.25	79.64	88.60	88.82	89.00	84.90	85.06	85.41	80.12	80.73	81.46	62.00	62.35	62.71
G-pruner	89.20	89.55	89.93	81.05	81.49	81.89	90.85	91.08	91.23	88.20	88.44	88.65	83.22	84.03	84.76	67.29	67.64	67.93
	+3.28%	+3.23%	+3.22%	+2.22%	+2.24%	+2.25%	+2.25%	+2.26%	+2.23%	+3.30%	+3.38%	+3.24%	+3.10%	+3.30%	+3.30%	+5.29%	+5.29%	+5.22%

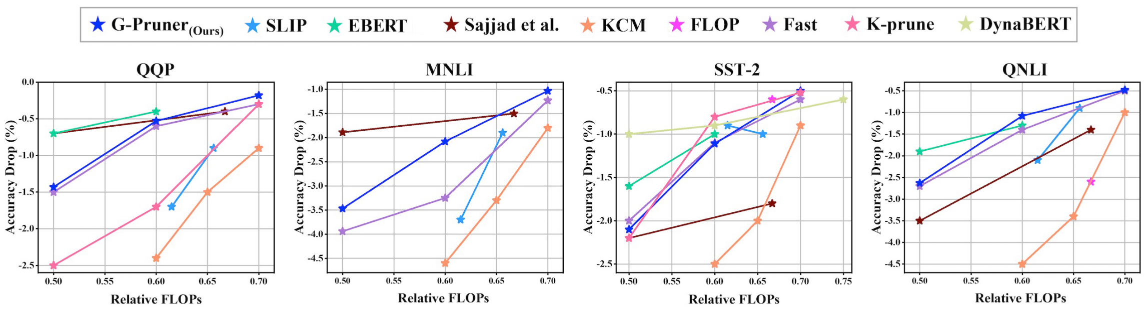

Figure 2: **Based on the BERT$_{BASE}$ model, we compare the performance of our pruning method with several existing structured pruning techniques.**

5 Experiments

5.1 Experimental Setup

Datasets and Pretrained models. Our research utilizes PyTorch v1.9.1 and Hugging Face's Transformers v4.12.0. Experiments are conducted on a single NVIDIA GeForce RTX 3090 GPU for efficiency and result reproducibility. We evaluate our pruning method on popular benchmarks: GLUE for tasks like QQP (364K), SST-2 (67K), MNLI (392K), and QNLI (105K), and SQuAD1.1 (88K) and SQuAD2.0 (130K) for question-answering. We focus on BERT$_{BASE}$ and DistilBERT models.

Competitors and and Performance Comparison. In our research, we conduct detailed comparisons of our pruning method with several domain-specific retraining-free algorithms: KCM Nova et al. (2023), Kwon et al. (2022), and K-prune Park et al. (2023). Additionally, we compare against recent retraining-based algorithms like Flop Wang et al. (2019), SLIP Lin et al. (2020), Sajjad et al. Sajjad et al. (2023), EBERT Liu et al. (2021b), and DynaBERT Hou et al. (2020b). These comparisons focus on performance metrics under various FLOPs constraints. Given the slight variations in baseline accuracy among these papers, directly comparing the absolute accuracy of pruned models is challenging. To facilitate effective comparisons, we adopt accuracy degradation (i.e., the difference in accuracy between pruned and original models) as the primary evaluation metric. Regarding pruning efficiency, our focus is primarily on performance under a 60% FLOPs constraint.

Baseline Configuration. We use BERT$_{BASE}$ and DistilBERT as our baseline models, maintaining their original architectures and configurations. For pruning, we randomly select 2,000 samples from their training sets to ensure swift and efficient processing, avoiding overfitting while preserving model accuracy. We evaluate accuracy on GLUE tasks and F1 score on SQuAD tasks. To ensure reliable results, we conduct experiments with ten random seeds and report average outcomes.

5.2 Accuracy Comparison

As shown in Figure 2, while all methods inevitably sacrifice some degree of accuracy when reducing FLOPs, our approach exhibits the least accuracy degradation in most cases. Particularly under more lenient FLOPs constraints, its performance advantage becomes more pronounced. This suggests that at the same pruning cost, our method achieves significantly higher accuracy compared to other algorithms. In other words, if we can maintain the same level of accuracy as other algorithms, we can perform more extensive pruning operations.

As shown in Table 1, we compare the accuracy of BERT$_{BASE}$ and DistilBERT models against the baseline model under different FLOPs constraints. The results indicate a significant improvement in

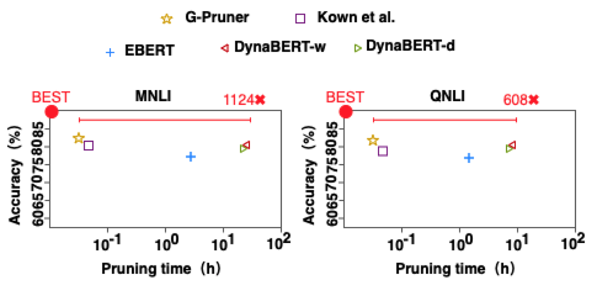

Figure 3: **Under a 60% FLOPs constraint, the accuracy of compressed models is compared with the time cost required for pruning.**

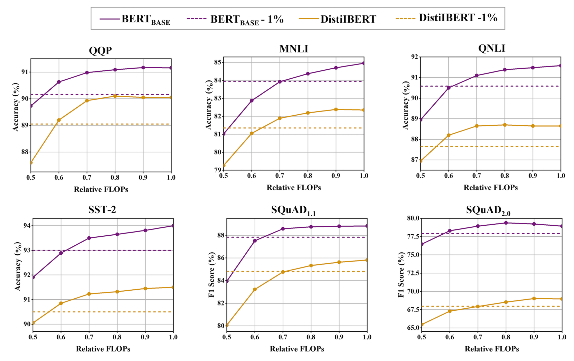

Figure 4: **Despite a strict 1% maximum allowable drop in accuracy,BERT$_{\text{BASE}}$ achieves 60–70% of the original FLOPs for all tasks.**

accuracy with G-Pruner. Particularly, under a 60% FLOPs constraint, the model achieves a 6.02% higher F1 score on the SQuAD$_{2.0}$ task compared to the baseline model.

5.3 Speed Comparison

As shown in Figure 3, we evaluate the cost-effectiveness of each pruning algorithm by comparing the model accuracy at a 60% compression rate on the MNLI and QNLI datasets and the time required for pruning (measured in hours). Notably, G-pruner not only shows higher accuracy than other methods in all experimental settings but also significantly reduces pruning costs, by up to 1124×.

5.4 FLOPs

As illustrated in Figure 4, we further analyzed the accuracy variations of BERT$_{\text{BASE}}$ and DistilBERT under different FLOPs constraints. Our analysis demonstrates that with just a 1% decrease in accuracy, BERT$_{\text{BASE}}$ maintains 60-70% of its original FLOPs across all tasks.

5.5 Ablation Studies

As shown in Table 2, we conducted ablation studies on the PPOM and CG²MT enhancement modules. While maintaining 60% of FLOPs, we set mask search as the baseline pruning method and then compared it with the addition of PPOM and CG²MT modules. The results indicate that introducing the PPOM module slightly reduces model speed, but adjusting the CG²MT module significantly reduces the time required for model pruning. Additionally, both PPOM and CG²MT modules significantly improve accuracy. For instance, in the QNLI task, the CG²MT module increases the accuracy of the BERT$_{\text{BASE}}$ model by 2.03%, while the CG²MT module shows a more pronounced improvement, boosting accuracy by 2.29%.

Table 2: **The ablation study, the accuracy results under the 60% FLOPs constraint.**

Accuracy(%)					
	QQP	MNLI	SST-2	QNLI	Avg. Diff
Mask search	87.40	80.52	90.50	87.04	-
+ PPOM	89.84	81.87	91.25	89.33	+1.70
+ CG²MT	89.67	81.58	91.66	89.07	+1.63
+ PPOM + CG²MT	90.63	82.87	92.89	90.50	+2.85

Pruning Time(s)					
	QQP	MNLI	SST-2	QNLI	Avg. Diff
Mask search	30.21	31.44	52.45	53.38	-
+ PPOM	40.15	41.57	63.44	64.52	+10.55
+ CG²MT	9.07	10.58	16.56	16.47	-28.70
+ PPOM + CG²MT	13.43	14.55	21.21	21.03	-24.31

6 Conclusion

In this work, we introduce a structured pruning algorithm named G-Pruner, which achieves high-precision pruning without the need to retrain Transformer models. By incorporating two novel techniques, PPOM and CG²MT, we effectively address the shortsightedness problem commonly encountered in traditional methods when assessing the importance of attention heads and feed-forward neural networks. Simultaneously, our approach significantly optimizes the iterative process, reducing numerical instability during computation and achieving faster convergence. Under the same FLOPs constraints, G-Pruner significantly outperforms all existing pruning techniques in pruning time without sacrificing model accuracy.

Funding. This work was supported by the National Natural Science Foundation of China (No.62107009) and the Fund of Jilin Provincial Department of Education Project (No.JJKH20241427KJ).

References

Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. 2023. A survey on deep neural network pruning-taxonomy, comparison, analysis, and recommendations. *arXiv preprint arXiv:2308.06767*.

Lucio Dery, Steven Kolawole, Jean-Francois Kagey, Virginia Smith, Graham Neubig, and Ameet Talwalkar. 2024. Everybody prune now: Structured pruning of llms with only forward passes. *arXiv preprint arXiv:2402.05406*.

Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. 2023. Depgraph: Towards any structural pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16091–16101.

Nuno Guimarães, Ricardo Campos, and Alípio Jorge. 2024. Pre-trained language models: What do they know? *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 14(1):e1518.

Yang He and Lingao Xiao. 2023. Structured pruning for deep convolutional neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Xanh Ho, Anh Khoa Duong Nguyen, An Tuan Dao, Junfeng Jiang, Yuki Chida, Kaito Sugimoto, Huy Quoc To, Florian Boudin, and Akiko Aizawa. 2024. A survey of pre-trained language models for processing scientific text. *arXiv preprint arXiv:2401.17824*.

Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. 2020a. Dynabert: Dynamic bert with adaptive width and depth. *Advances in Neural Information Processing Systems*, 33:9782–9793.

Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. 2020b. Dynabert: Dynamic bert with adaptive width and depth. *Advances in Neural Information Processing Systems*, 33:9782–9793.

Forrest N Iandola, Albert E Shaw, Ravi Krishna, and Kurt W Keutzer. 2020. Squeezebert: What can computer vision teach nlp about efficient neural networks? *arXiv preprint arXiv:2006.11316*.

Woojeong Kim, Suhyun Kim, Mincheol Park, and Geunseok Jeon. 2020. Neuron merging: Compensating for pruned neurons. *Advances in Neural Information Processing Systems*, 33:585–595.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.

Takeshi Kojima, Itsuki Okimura, Yusuke Iwasawa, Hitomi Yanaka, and Yutaka Matsuo. 2024. On the multilingual ability of decoder-based pre-trained language models: Finding and controlling language-specific neurons. *arXiv preprint arXiv:2404.02431*.

Eldar Kurtic, Daniel Campos, Tuan Nguyen, Elias Frantar, Mark Kurtz, Benjamin Fineran, Michael Goin, and Dan Alistarh. 2022. The optimal bert surgeon: Scalable and accurate second-order pruning for large language models. *arXiv preprint arXiv:2203.07259*.

Woosuk Kwon, Sehoon Kim, Michael W Mahoney, Joseph Hassoun, Kurt Keutzer, and Amir Gholami. 2022. A fast post-training pruning framework for transformers. *Advances in Neural Information Processing Systems*, 35:24101–24116.

Ivan Lazarevich, Alexander Kozlov, and Nikita Malinin. 2021. Post-training deep neural network pruning via layer-wise calibration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 798–805.

Bingbing Li, Zhenglun Kong, Tianyun Zhang, Ji Li, Zhengang Li, Hang Liu, and Caiwen Ding. 2020. Efficient transformer-based large scale language representations using hardware-friendly block structured pruning. *arXiv preprint arXiv:2009.08065*.

Junyi Li, Tianyi Tang, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2024. Pre-trained language models for text generation: A survey. *ACM Computing Surveys*, 56(9):1–39.

Zi Lin, Jeremiah Zhe Liu, Zi Yang, Nan Hua, and Dan Roth. 2020. Pruning redundant mappings in transformer models via spectral-normalized identity prior. *arXiv preprint arXiv:2010.01791*.

Yuanxin Liu, Zheng Lin, and Fengcheng Yuan. 2021a. Rosita: Refined bert compression with integrated techniques. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8715–8722.

Zejian Liu, Fanrong Li, Gang Li, and Jian Cheng. 2021b. Ebert: Efficient bert inference with dynamic structured pruning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4814–4823.

Taiyuan Mei, Yun Zi, Xiaohan Cheng, Zijun Gao, Qi Wang, and Haowei Yang. 2024. Efficiency optimization of large-scale language models based on deep learning in natural language processing tasks. *arXiv preprint arXiv:2405.11704*.

Ben Mussay, Margarita Osadchy, Vladimir Braverman, Samson Zhou, and Dan Feldman. 2019. Data-independent neural pruning via coresets. *arXiv preprint arXiv:1907.04018*.

Azade Nova, Hanjun Dai, and Dale Schuurmans. 2023. Gradient-free structured pruning with unlabeled data. In *International Conference on Machine Learning*, pages 26326–26341. PMLR.

Adedoyin Tolulope Oyewole, Omotayo Bukola Adeoye, Wilhelmina Afua Addy, Chinwe Chinazo Okoye, Onyeka Chrisanctus Ofodile, and Chinonye Esther Ugochukwu. 2024. Automating financial reporting with natural language processing: A review and case analysis. *World Journal of Advanced Research and Reviews*, 21(3):575–589.

Seungcheol Park, Hojun Choi, and U Kang. 2023. Accurate retraining-free pruning for pretrained encoder-based language models. In *The Twelfth International Conference on Learning Representations*.

Shaina Raza, Muskan Garg, Deepak John Reji, Syed Raza Bashir, and Chen Ding. 2024. Nbias: A natural language processing framework for bias identification in text. *Expert Systems with Applications*, 237:121542.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. 2023. On the effect of dropping layers of pre-trained transformer models. *Computer Speech & Language*, 77:101429.

Victor Sanh, Thomas Wolf, and Alexander Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning. *Advances in neural information processing systems*, 33:20378–20389.

Michael Santacroce, Zixin Wen, Yelong Shen, and Yuanzhi Li. 2023. What matters in the structured pruning of generative language models? *arXiv preprint arXiv:2302.03773*.

Alireza Shamshiri, Kyeong Rok Ryu, and June Young Park. 2024. Text mining and natural language processing in construction. *Automation in Construction*, 158:105200.

Xinyu Shi, Jianhao Ding, Zecheng Hao, and Zhaofei Yu. 2024. Towards energy efficient spiking neural networks: An unstructured pruning framework. In *The Twelfth International Conference on Learning Representations*.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. Mobilebert: a compact task-agnostic bert for resource-limited devices. *arXiv preprint arXiv:2004.02984*.

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

Ziheng Wang, Jeremy Wohlwend, and Tao Lei. 2019. Structured pruning of large language models. *arXiv preprint arXiv:1910.04732*.

Mengzhou Xia, Zexuan Zhong, and Danqi Chen. 2022. Structured pruning learns compact and accurate models. *arXiv preprint arXiv:2204.00408*.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. 2024. Wizardlm: Empowering large pre-trained language models to follow complex instructions. In *The Twelfth International Conference on Learning Representations*.

Edouard Yvinec, Arnaud Dapogny, Matthieu Cord, and Kevin Bailly. 2021. Red: Looking for redundancies for data-freestructured compression of deep neural networks. *Advances in Neural Information Processing Systems*, 34:20863–20873.

Shaowei Zhang, Rongwang Yin, and Mengzi Zhang. 2024. Dynamic unstructured pruning neural network image super-resolution reconstruction. *Informatica*, 48(7).

Haotian Zheng, Kangming Xu, Huiming Zhou, Yufu Wang, and Guangze Su. 2024. Medication recommendation system based on natural language processing for patient emotion analysis. *Academic Journal of Science and Technology*, 10(1):62–68.