Towards Lossless Head Pruning through Automatic Peer Distillation for Language Models

Bingbing Li[1], **Zigeng Wang**[1*], **Shaoyi Huang**[1], **Mikhail Bragin**[1], **Ji Li**[2] and **Caiwen Ding**[1]
[1]University of Connecticut
[2]Microsoft Corporation
{bingbing.li, zigeng.wang, shaoyi.huang, mikhail.bragin, caiwen.ding}@uconn.edu,
changzhouliji@gmail.com

Abstract

Pruning has been extensively studied in Transformer-based language models to improve efficiency. Typically, we zero (prune) unimportant model weights and train a derived compact model to improve final accuracy. For pruned weights, we treat them as useless and discard them. This usually leads to significant model accuracy degradation. In this paper, we focus on attention head pruning as head attention is a key component of the transformer-based language models and provides interpretable knowledge meaning. We reveal the relationship between pruned attention heads and retained heads and provide a solution to recycle the discarded knowledge from the pruned heads, named peer distillation. We also develop an automatic framework to locate the to-be-pruned attention heads in each layer, freeing the time-consuming human labor in tuning hyper-parameters. Experimental results on the General Language Understanding Evaluation (GLUE) benchmark are provided using BERT model. By recycling discarded knowledge from pruned heads, the proposed method maintains model performance across all nine tasks while reducing heads by over 58% on average and outperforms state-of-the-art techniques (e.g., Random, HISP, L0 Norm, SMP).

1 Introduction

Transformer-based language models [Devlin *et al.*, 2018; Yang *et al.*, 2019] have been proven to be highly effective in learning universal language representations and applicable to downstream tasks with slight fine-tuning, as opposed to other machine learning methods [Hochreiter and Schmidhuber, 1997; He *et al.*, 2016; Zhou *et al.*, 2020]. These models tend to suffer from high computational cost and memory usage due to their large model size. To downsize the transformer models, head pruning, a sub-category of weight pruning technique has been widely investigated [Michel *et al.*, 2019; Kovaleva *et al.*, 2019; Voita *et al.*, 2019; Lee *et al.*, 2020].

[*]Z. Wang is now affiliated with Walmart Global Tech, Sunnyvale, CA

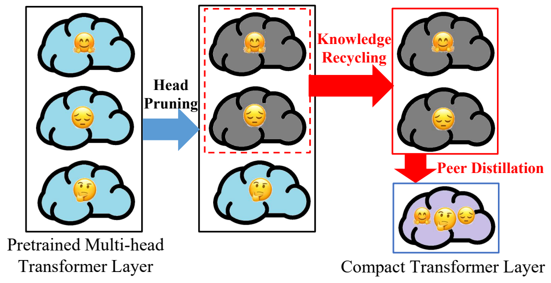

Head Pruning
Knowledge Recycling
Peer Distillation
Pretrained Multi-head Transformer Layer
Compact Transformer Layer

Figure 1: Automatic head pruning based on Peer Distillation (Pruning is designed for heads selection and peer distillation for model performance preservation)

Head pruning usually requires iterative but independent *sparsification* and *retraining* operations, which usually results in a significant drop in model accuracy. Specifically, during sparsification, head masks are updated to select redundant heads for pruning. During retraining, the retained attention heads are updated to recover the previous model accuracy [Michel *et al.*, 2019; Voita *et al.*, 2019]. We observe that such a workflow usually leads to a significant drop in model accuracy. The reason behind the drop in model accuracy is the "knowledge" loss from the pruned attention heads.

In this paper, we propose a counter-traditional pruning strategy, *peer distillation*, and reveal that **the knowledge learned in advance but discarded from the pruned attention heads plays a crucial role in maintaining model accuracy**. The pruned attention heads contain useful grammatical and semantic information to maintain model performance and provide valuable guidance for the training of the retained heads. We also develop an automatic framework to locate the to-be-pruned attention heads in each layer, freeing the time-consuming human labor in tuning hyperparameters. Figure 1 illustrates the "*peer distillation*" process, where we recycle the knowledge from pruned heads to retained heads. To our best knowledge, peer distillation is the first attempt to recycle pruned knowledge and mitigate accuracy degradation. Benefiting from the recycled knowledge, the accuracy of the model is better maintained at higher model sparsity. Furthermore, peer distillation accelerates pruning convergence by pruning more attention heads while maintaining model accuracy.

Our contributions are as follows:

- We provide an automatic head pruning approach that achieves the best accuracy-sparsity tradeoff by reformulating the head pruning optimization. Specifically, we define a straightforward gating function for head pruning and propose a softplus STE function to address the mixed integer back propagation. Additionally, we provide head pruning visualizations to show the variable choice of model substructures during pruning.
- We propose the peer distillation strategy to recycle the discarded knowledge from the pruned heads. Specifically, we distill the pruned heads into the retained heads for each transformer layer. Since the pruned head maintains important "knowledge" to preserve the model accuracy, this strategy helps the retained head absorb this "knowledge" and thus be more capable of maintaining model accuracy.

Experimental results on nine GLUE benchmark tasks [Wang *et al.*, 2018] show that we achieve high compression rates with zero or minor accuracy degradation. The pruned models outperform the original ones with only 41.71% heads left on average. In extreme cases, our proposed method can prune 99.3% (143 of the 144) heads without accuracy degradation on WNLI dataset. On average, we could prune 79.74% heads with only 0.82% accuracy degradation. Compared with other differentiable head pruning, e.g. [Xiao *et al.*, 2019], peer distillation increases the head sparsity by 22.17% with the same model accuracy on MRPC dataset. The method provides an efficient tool to analyze and reduce the redundancy of multi-heads and is suitable for other attention-based models.

2 Related Works and Background

Related Works. The redundancy of the multi-head mechanism has been discovered and investigated. [Michel *et al.*, 2019; Kovaleva *et al.*, 2019] first discusses duplicated attention head patterns. [Raganato *et al.*, 2020] presents that models with fixed single attention head for each layer would nicely preserve model accuracy. [An *et al.*, 2020] analyzes head redundancy from a Bayesian perspective and explains the causes of such redundancy.

Different attention head pruning algorithms are developed. [Michel *et al.*, 2019] prunes attention head greedily based on predefined head importance metric but the pruned heads can never be recovered during training. [Kovaleva *et al.*, 2019] shows the attention head redundancy and manually disables attention heads to improve model performance. [Voita *et al.*, 2019] employs Gumbel softmax to relax the head pruning problem to be a differentiable subnetwork searching problem but more experiments and discussion are expected to prove its effectiveness. [Lee *et al.*, 2020] applies deep Q-learning to automatically prune attention heads but the search time can be comparatively long. More recently, a self-supervised meta-pruning framework (SMP) [Zhang *et al.*, 2021] is designed by combining head importance scoring and Gumbel softmax pruning through representation distance minimization.

Challenges. *Accuracy degradation is inevitable during and after head pruning.* While performing head pruning, since the pruned heads will no longer participate in the training process, the output of the model will change significantly, resulting in an irreversible accuracy loss. During pruning, the retained heads have to recover the model accuracy without any auxiliary information, which is extremely difficult, especially when the pruning granularity (single weight, row, column, head, etc.) is large.

Multi-head Attention. Self-attention plays an important role in Transformer-based language models. In Transformer layers, multiple attention heads work in parallel. The self-attention is calculated based on Query (Q), Key (K), and Value (V) matrices and the attention score matrix is calculated as follows:

$$A = Softmax(\frac{Q * K^T}{\sqrt{D_k}}) \tag{1}$$

where A is the attention score matrix and D_k is the dimension of matrix K.

The multi-head attention mechanism uses different matrics of (Q, K, V) to learn different representation subspaces. After concatenating the derived attention heads, a feed-forward layer is utilized to project the concatenation:

$$\begin{aligned} H_i &= A_i * V_i \\ &= Softmax(\frac{Q_i * K_i^T}{\sqrt{D_k}}) * V_i \\ &= Softmax(\frac{(X * W_i^Q) * (X * W_i^K)^T}{\sqrt{D_k}}) * (X * W_i^V) \end{aligned} \tag{2}$$

$$MH = MultiHead(Q, K, V) = Concat_i(H_i) * W^O \tag{3}$$

where X denotes the input of the i-th attention head, W_i^Q, W_i^K, and W_i^V are attention matrices, W^O is projection matrix, and H_i denotes attention head.

Transformer-based model relies on multiple encoder layers. For each layer, the output is formulated as follows:

$$O = Norm(ResConnect(MH)) \tag{4}$$

where O is the output of the transformer layer, $Norm(\cdot)$ is the layer normalization function, $ResConnect(\cdot)$ denotes the residual connection [Devlin *et al.*, 2018].

3 Peer Distillation for Automatic Head Pruning

We propose the peer distillation head pruning framework for automatic head selection and model performance preservation as shown in Fig. 2. First, differentiable head pruning is performed to select heads by automatically updating binary gates. Then, the discarded "knowledge" related to the pruned heads is recycled and peer distillation is performed to distill the discarded "knowledge" into the derived compact model. It is worth noting that the pruning and peer distillation operations are performed in the same training loop.

Differentiable Head Pruning. In this section, we propose a differentiable method for head pruning. Unlike pruning methods with hard constraints [Han *et al.*, 2015; Boyd *et al.*, 2011; Li *et al.*, 2016; Li *et al.*, 2020], our method obtains model sparsity by updating the gate parameters. This leads to two

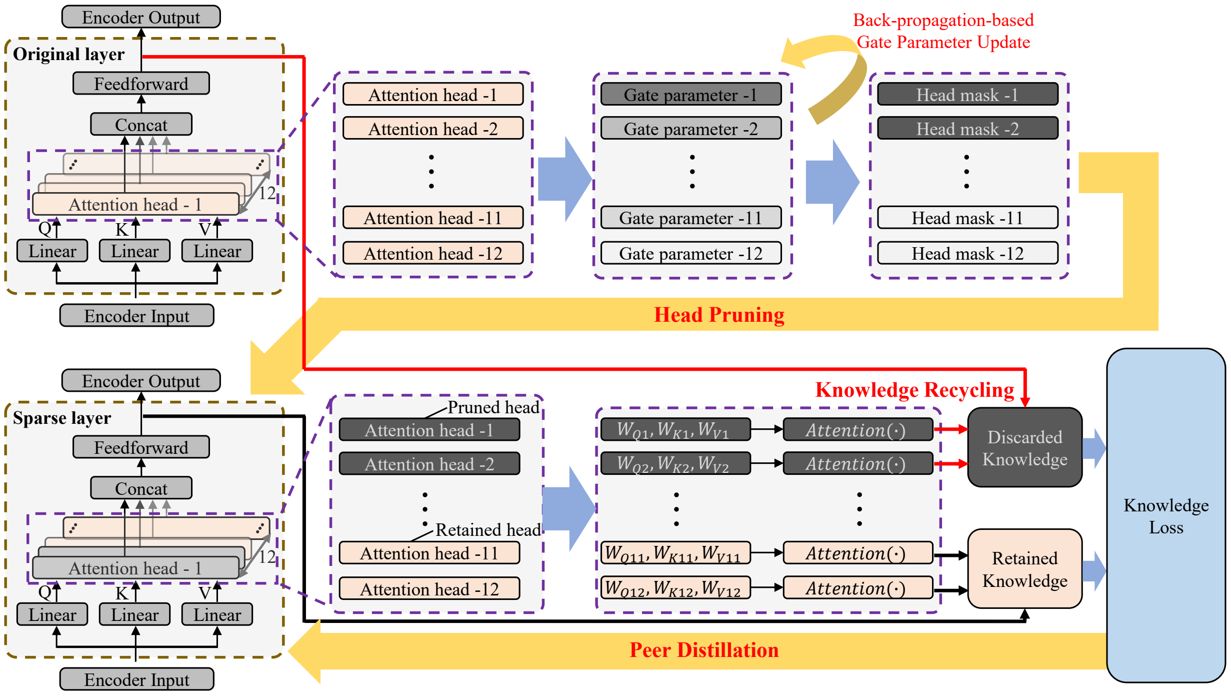


Figure 2: Peer distillation head pruning framework

important benefits: first, we do not have to set expected sparsity for each layer and can obtain model sparsity automatically; second, without hard constraints, the pruning process prunes the model more "smoothly" and nicely remains model accuracy. This makes the retraining process not a necessity and leads to faster training convergence.

In order to achieve sparse attention heads, we define the pruning loss consisting of model accuracy loss and model sparsity loss. The head pruning problem can be formulated by the following steps: first, we introduce the attention head mask M, in which M is composed of lists of binary variables representing the status of their corresponding heads and can be defined as:

$$M_{ij} = \begin{cases} 0, & \text{if corresponding } \textit{head} \text{ is pruned;} \\ 1, & \text{otherwise.} \end{cases} \tag{5}$$

where i and j denote the index of attention head and layer, respectively, and M_{ij} denotes the pruning status of the head.

Second, inspired by the early works on neural network quantization and pruning [Hubara *et al.*, 2016; Xiao *et al.*, 2019], we employ learnable discrete functions called straight through estimators (STEs) g to describe the head mask M and thus M can be formulated as:

$$M_{ij} = g(W'_{ij}) = \begin{cases} 0, & \text{if } W'_{ij} \leq 0 \\ 1, & \text{if } W'_{ij} > 0. \end{cases} \tag{6}$$

where W'_{ij} is the auxiliary parameter to *control* the *open* and *close* of the binary gates, the binary head mask M_{ij} is represented as a step function g with a continuous auxiliary parameter W'_{ij}.

Third, we formulate the head pruning problem as:

$$\min_{W'} \mathcal{L}_p = \min_{W'} \mathcal{L}_A(W'; W) + \mu \cdot \sum g(W'), \tag{7}$$

where $\mathcal{L}_p$ is the pruning loss to update auxiliary parameters W', $\mathcal{L}_A$ is the model accuracy loss with model weight W, and μ is the penalty factor.

However, due to the binary nature of $g(W')$ and the continuous weights W values, the problem described in Eq. 7 is a mixed integer programming problem, which brings difficulties in optimizing it directly using back-propagation.

To update the sparse head structure, we introduce coarse gradients [Hubara *et al.*, 2016] to make the binarized function g differentiable. Coarse gradients provide a good approximation for updating parameter W' through back-propagation and could ensure that the update direction of W'_{ij} gradient reflects the accuracy and sparsity objectives of the model [Xiao *et al.*, 2019].

Different coarse gradients have been practiced and discussed in literature [Srinivas *et al.*, 2017; Yin *et al.*, 2019]. We use Softplus STE in [Xiao *et al.*, 2019] and the auxiliary parameter W' can be updated as:

$$W' \leftarrow W' - l_{rg} * \frac{\partial \mathcal{L}_p}{\partial W'} \tag{8}$$

$$\frac{\mathcal{L}_p}{\partial W'} = \frac{\partial \mathcal{L}_p}{\partial M} * \frac{\partial M}{\partial W'} = \frac{\partial \mathcal{L}_p}{\partial M} * Softplus(W') \tag{9}$$

where l_{rg} is the learning rate of the optimizer to update W'.

Knowledge Recycling. To connect the separate sparsification and training stages, and preserve the "knowledge" of the pruned heads, we introduce knowledge recycling and peer distillation. After differentiable pruning, we obtain the binary head mask M and thus a compact model with fewer attention heads. Previous research [Hu *et al.*, 2018; Voita *et al.*, 2019; Lee *et al.*, 2020; Wang *et al.*, 2020; Huang *et al.*, 2022] just discarded the pruned heads and trained the remaining model parameters directly. However, this will introduce significant accuracy loss. By introducing knowledge recycling, we provide the opportunity to maintain and make full use of the existing knowledge in the entire model to boost model performance.

We select head attention score matrix, A, and layer output, O, as representations of head knowledge. A contains the abstraction of the input and the relationship extraction between input tokens. As an interface with other model layers, the layer output, O, integrates the effective information of all heads and directly reflects the impact of head pruning.

To update the model weight W and model structure (represented by auxiliary parameters W'), two back-propagations are performed. For each iteration (one batch of input), the knowledge recycling process is performed by four steps:

Step1: We do model forward to derive the complete knowledge of the dense model according to Eq. 1, 4, and 5:

$$A^c = Softmax(\frac{Q * K^T}{\sqrt{D_k}}) * M \tag{10}$$

$$\begin{aligned} O^c &= Norm(ResConnect(MH)) \\ &= Norm(ResConnect(Concat_i(A_i * M * V_i) * W^O)) \end{aligned} \tag{11}$$

where M is the current head mask, A^c and O^c are the complete head attention score matrix and layer output before updating M, respectively. Initially, M is an all-ones matrix.

Step2: By leveraging differentiable head pruning in Eq. 8, we update M and derive the structure of the pruned model according to Eq. 6 and 5.

Step3: For the derived pruned model with updated M, we do model forward to derive the retained knowledge (A^r, O^r), where A^r and O^r are the retained attention score matrix and layer output after updating M, respectively.

Step4: To preserve the model performance, we do peer distillation (described in the following subsection).

Peer Distillation. Traditional Knowledge Distillation (KD) method distills "knowledge" from the original model (teacher) to another model (student). For transformer-based models, different heads in the same layer work in parallel to provide abstract representations of input. The pruned heads could provide additional learned model information, which is, however, discarded during the training of the compact model.

The proposed Peer Distillation (PD) method distills the "knowledge" from the pruned model structure to the retained model structure in the same layer. Specifically, the retained heads learn to fit the matrices of head attention score before pruning (teacher) and the objective is defined as:

$$PD_{loss1} = \sum_{j=1}^{N}\sum_{i=1}^{S} MSE(A^c_{ji}, A^r_{ji}) \tag{12}$$

where N is the number of transformer layer of the model (e.g. 12 for BERT$_{\text{BASE}}$ model), S is the number of attention heads of a transformer layer(e.g. 12 for BERT$_{\text{BASE}}$ model), $MSE(\cdot,\cdot)$ is the *mean squared error* loss function, A^c_{ji} and A^r_{ji} are the i-th attention score matrices (formulated in Eq. 10) in the j-th layer before and after differentiable pruning, respectively.

Algorithm 1 Peer distillation head pruning procedure

```
Input: pretrained tranformer-based model $\mathbf{T_M}$, head pruning optimizer $\mathbf{OP_H}$, weight optimizer $\mathbf{OP_W}$
Output: updated model weight $\mathbf{W}$ and head masks $\mathbf{M}$
 1: for every layer $\mathbf{L_j}$ in $\mathbf{T_M}$ do
 2:     for every head $\mathbf{H_i}$ in $\mathbf{L_j}$ do
 3:         Create and initialize auxiliary parameters $\mathbf{W'_{ij}}$
 4:     end for
 5: end for
 6: for every training iteration do
 7:     Do model forward to derive head attention socre $A^c$ and layer output $O^c$ in Eq. 10 and 11
 8:     Calculate head pruning loss $\mathcal{L}_p =$ in Eq. 7
 9:     Update $\mathbf{W'}$ through back propagation by minimizing $\mathcal{L}_p$ using $\mathbf{OP_H}$
10:     Detach $A^c$ and $O^c$ from the computation graph to avoid gradient calculation
11:     Update head masks $\mathbf{M}$ in Eq. 6
12:     Do model forward to derive head attention socre $A^r$ and layer output $O^r$ in Eq. 10 and 11 with updated head masks $\mathbf{M}$
13:     Calculate the mixed loss $\mathcal{L}_m$ in Eq. 14
14:     Update $\mathbf{W}$ through back propagation by minimizing $\mathcal{L}_m$ using $\mathbf{OP_W}$
15: end for
```

We also distill the layer outputs from the unpruned to retained heads to better maintain the original model knowledge. The objective is defined as:

$$PD_{loss2} = \sum_{j=1}^{N} MSE(O^c_j, O^r_j) \tag{13}$$

where O^c_k and O^r_k are the outputs (formulated in Eq. 11) of the k-th layer before and after differentiable pruning, respectively.

Finally, we formulate the parameter update problem of the derived compact model as follows:

$$\min_W \mathcal{L}_m = \min_W \mathcal{L}_A(W; W') + \lambda \cdot (PD_{loss1} + PD_{loss2}) \tag{14}$$

where $\mathcal{L}_m$ is the mixed loss to update model weights W and λ is the penalty factor. PD_{loss1} and PD_{loss2} are peer distillation losses with arguments W (including W^Q, W^K, W^V and W^O) according to Eq. 2 and 3. To notice, while calculating Eq. 14, all model parameters, W, will be updated.

By minimizing the mixed loss, $\mathcal{L}_m$, the retained model parameters are updated to preserve the model accuracy. The previously discarded model information of the pruned heads plays as a "teacher" and transfers its "knowledge" to the "students" (i.e. retained heads).

Algorithm 1 illustrates the peer distillation head pruning procedure. We conduct head pruning and peer distillation in the same training loop and update the model weight and structure in parallel. To notice, we detach the derived discarded knowledge A^c and O^c ("teacher" in the peer distillation method) from DNN graph to ensure that only weights of the compact model are updated.

Pruning Method	MNLI	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	WNLI
None [Devlin *et al.*, 2018]	83.9	91.2	91.1	92.7	53.4	85.8	88.9	66.4	56.3
Random [Zhang *et al.*, 2021]	82.43	90.34	-	91.83	52.37	85.33	80.88	65.77	-
HISP [Michel *et al.*, 2019]	81.69	86.88	-	91.85	54.84	85.96	81.12	65.34	-
$L0$ Norm [Voita *et al.*, 2019]	79.70	85.82	-	91.74	52.10	85.80	77.45	62.45	-
SMP [Zhang *et al.*, 2021]	83.36	90.96	-	92.31	57.26	85.99	85.04	67.87	-
Peer Distillation (Ours)	**83.66**	**91.07**	**91.25**	**92.89**	**60.39**	**86.94**	**88.62**	65.7	**56.34**

Table 1: Head pruning methods comparison of evaluation accuracy among the 9 GLUE benchmark tasks with 50% head sparsity.

Models	MNLI	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	WNLI	Ave.
BERT$_{\text{BASE}}$	83.9	91.2	91.1	92.7	53.4	85.8	88.9	66.4	56.3	
Head sparsity	45.14%	56.94%	36.81%	72.92%	54.17%	63.19%	51.67%	44.44%	99.3%	58.29%
Peer Distillation (Ours)	83.87	**91.3**	**91.38**	**92.89**	**60.39**	**86.94**	**89.52**	**67.87**	**56.34**	

Table 2: Head pruning results of evaluation accuracy using our peer distillation method among the 9 GLUE benchmark tasks. Bold font indicates that the pruned model outperforms the original one.

Models	MNLI	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	WNLI	Ave.
BERT$_{\text{BASE}}$	83.9	91.2	91.1	92.7	53.4	85.8	88.9	66.4	56.3	
Peer Distillation (Ours)	82.9	90.28	90.15	91.77	52.46	84.96	87.99	65.5	56.34	
△ Accuracy	-1.00	-0.92	-0.95	-0.93	-0.94	-0.84	-0.91	-0.90	+0.04	**-0.82**
Head sparsity	76.68%	86.42%	59.5%	90.5%	82.81%	91.72%	67.64%	63.11%	99.3%	**79.74%**

Table 3: Comparison of evaluation accuracy among the 9 GLUE benchmark tasks in extreme cases (within 1% accuracy drop).

4 Evaluation

Datasets. We test our method on GLUE benchmark [Wang *et al.*, 2018] and report the performance of the unpruned and pruned models following the conventions by using accuracy for SST-2, QNLI, MNLI, QQP, RTE and WNLI; Matthews Correlation Coefficient (MCC) for CoLA, F1 scores for MRPC, and Spearman for STS-B. Our pre-trained model is the BERT$_{\text{BASE}}$ [Devlin *et al.*, 2018] model. The model consists of 12 attention layers and 12 heads for each layer. We define the head sparsity as:

$$head\_sparsity = \frac{\#pruned\_heads}{\#model\_heads} \tag{15}$$

Implementation Details. We follow the default finetuning steps for 9 tasks according to Huggingface [Wolf *et al.*, 2019] and obtain baseline models after training for 4 epochs. Then, differentiable pruning is executed for the models.

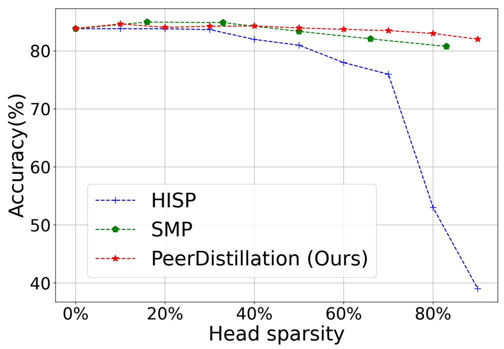


Figure 3: Model accuracy regarding to different head sparsity on MNLI-matched dataset

Baselines. To validate the effectiveness of our proposed method, we introduce four baselines in our experiments. For method **Random**, we report the results of 50% head sparsity from [Zhang *et al.*, 2021]. In [Michel *et al.*, 2019], the Head Importance Score for Pruning (**HISP**) is proposed by ranking the head importance and removing the heads with lower importance score. In our test, we calculate the head importance and prune 50% heads with lower importance scores. Method $\boldsymbol{L0}$ **Norm** represents the Gumbel softmax based pruning method proposed in [Voita *et al.*, 2019]. And the Single-Shot Meta-Pruner (**SMP**) is proposed by [Zhang *et al.*, 2021] in which head importance and Gumbel softmax based pruning are nicely combined.

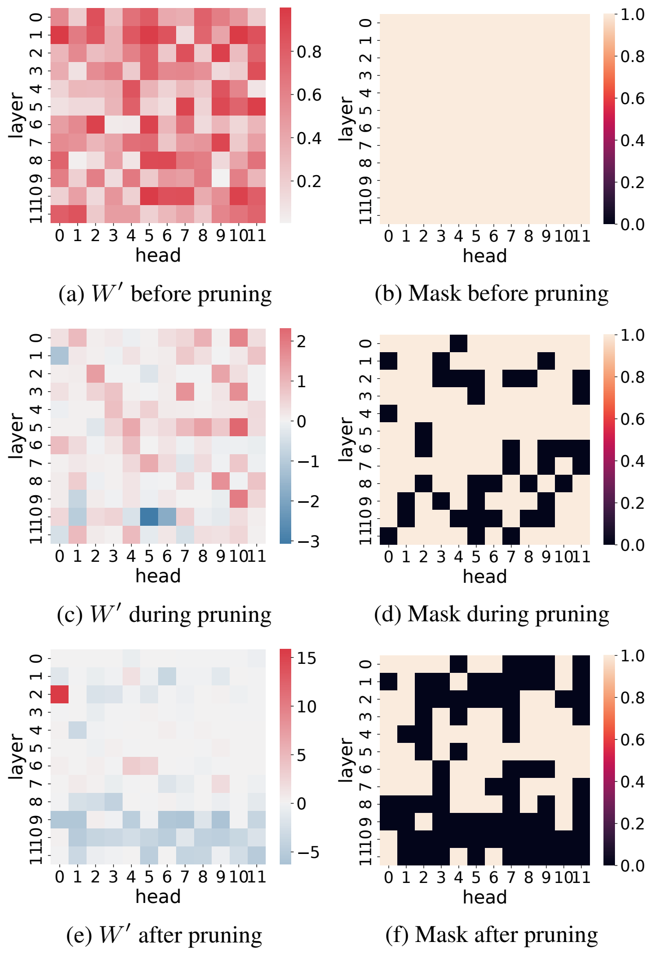
Experimental Results. We show our comparison results in Table 1. For fairness, we compare our head pruning model accuracy with state-of-the-art algorithms with fixed global head sparsity of 50%. Compared with Random method, our proposed method enjoys better model accuracy. HISP [Michel *et al.*, 2019] calculates head importance for pruning and suffers a significant accuracy drop since the importance is not estimated directly according to the final model performance. Our method outperforms $L0$ Norm approach [Voita *et al.*, 2019] with a large margin in all existing 7 GLUE benchmark tasks. SMP [Zhang *et al.*, 2021] improves Gumbel softmax based approach [Voita *et al.*, 2019] by combining head importance scoring and self-supervision, but the discrepancy between model training and model testing prohibits its further advances. Comparing with state-of-the-art head pruning methods, peer distillation takes the lead in 7 of the 8 tasks.

To fully investigate the performance of peer distillation, we test different sparsity levels. As shown in Fig. 3, our method can achieve more than 80% sparsity with only 1.03% accuracy drop and outperforms all existing head pruning methods. Additionally, for 8 of the 9 tasks, our pruned models outperform the original (unpruned) models as shown in Table 2, which is consistent with [Kovaleva *et al.*, 2019]'s study. More specifically, we could achieve 1.20% accuracy increase while pruning more than 58% heads on average. Surprisingly, the pruned model on CoLA dataset achieves 6.99% accuracy increase after pruning 54.17% heads and the pruned model on WNLI dataset has the same accuracy as the original one after pruning 99.3% heads (only one head left). In extreme cases (within 1% accuracy drop), our proposed method could prune 79.74% heads with 0.82% accuracy drop on average as shown in Table 3.

5 Discussion of Differentiable Head Pruning and Peer Distillation

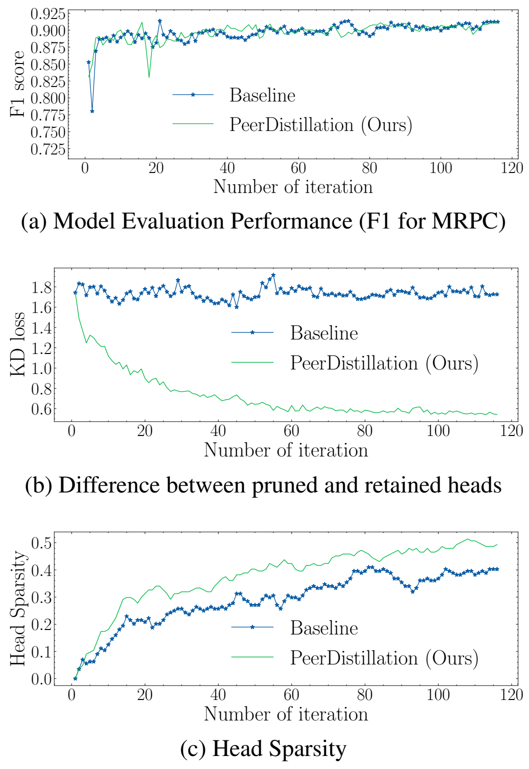
Differentiable Head Pruning. Different from the quantified importance scores of heads [Michel *et al.*, 2019] for pruning, in our method, we use learnable gate parameters to determine the retention of the heads. Fig. 4 shows the update process of the auxiliary parameter W' and head masking gate status $g(W')$ of the 144 attention heads jointly trained with the model weights W by following three stages:

- In the initialization stage, the auxiliary parameters are initialized by following a truncated normal distribution with values greater than 0 and gates are open (Fig. 4a and 4b).
- In the intermediate stage, the gate auxiliary parameters of less important heads are receiving negative gradients. Through training, some auxiliary parameters are below zero (denoted in cold colored boxes in Fig. 4c, which leads to pruning of corresponding attention heads in Fig. 4d.
- In the final stage, in Fig. 4e, more auxiliary parameters drop below zero. The optimization converges when the balance is achieved between the model accuracy component and sparsity component in loss function Eq. 7.



(a) W' before pruning (b) Mask before pruning
(c) W' during pruning (d) Mask during pruning
(e) W' after pruning (f) Mask after pruning

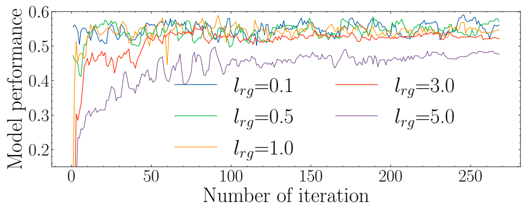
Figure 4: Differentiable pruning process on CoLA dataset



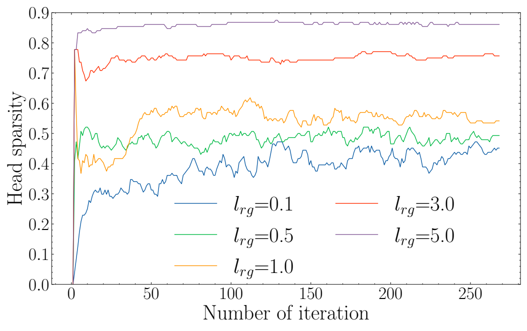
(a) Model Evaluation Performance (F1 for MRPC)
(b) Difference between pruned and retained heads
(c) Head Sparsity

Figure 5: Peer distillation performance on MRPC dataset

Peer Distillation Performance. To verify the effectiveness of the peer distillation, we conduct experiments of differentiable head pruning with and without peer distillation as shown in Fig. 5. Without peer distillation (baseline as the starred blue curve in Fig. 5b), the head difference (PD loss) changes slightly ($\pm$4.32%) during pruning and reveals that the retained heads converge to a different value instead of similar ones to the pruned heads. This proves our assumption that the retained compact model updates independently without absorbing useful knowledge from the pruned heads. With the help of peer distillation ($\lambda = 0.35$), the PD loss is reduced by 68.55% and the retained heads become much close to the pruned ones. **Compared with differentiable head pruning without peer distillation, the head sparsity is increased by 22.17% as shown in Fig. 5c while the evaluation model accuracy (F1 score) are the same (Fig. 5a)**. The result verifies the effectiveness of the peer distillation to preserve model performance with a shrunk model structure.

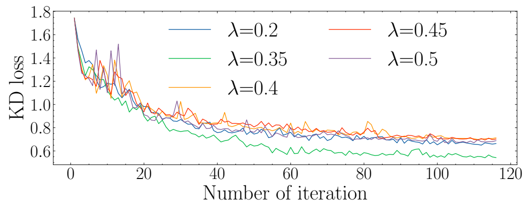


(a) Model Performance (mcc for CoLA)

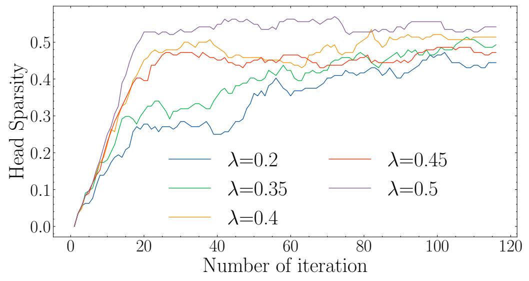


(b) Head Sparsity

Figure 6: Pruning ablation study: differentiable pruning optimizer with different learning rates on CoLA dataset.



(a) PD loss



(b) Head Sparsity

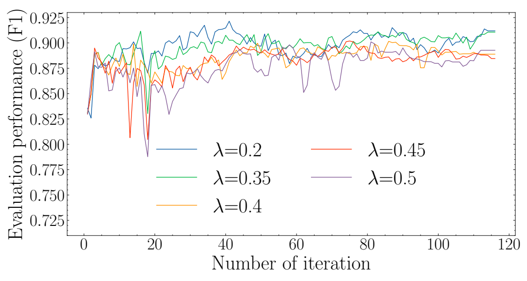
Figure 8: Peer distillation ablation study: PD loss and head sparsity with different PD loss penalty factors

6 Ablation Study

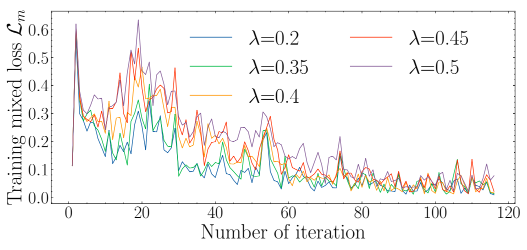
In this section, we perform ablation study over several hyper-parameters when doing differentiable pruning and peer distillation with BERT model and check their effects.

Differentiable Pruning Learning Rate. To solve the optimization problem in Eq. 7, different optimizers are utilized to update weight and gate parameters. Specially, we use the default initial learning rate (3e-5) to update model weights. For auxiliary parameters updating, a larger initial learning rate, l_{rg}, enhances the ability of the gate optimizer to adjust gate parameters and leads to higher sparsity. In our test, while increasing the initial l_{rg}, we could increase the sparsity from 44% to 76% with only 4% performance (mcc for CoLA dataset) drop as shown in Fig. 6. Interestingly, we observe the obvious compete between accuracy and sparsity, since different optimizers (weight and gate optimizer) tend to reduce the loss function in different directions.

Peer Distillation Loss Penalty Factor. While selecting different knowledge loss penalty factors, λ, from 0.2 to 0.5, we observe the similar compact model performance (values changed within $\pm 5.62\%$ in F1 score and ± 0.01 in final training mixed loss) in Fig. 7. However, while increasing λ, significant head sparsity increase (Fig. 8b) is achieved due to the reduced PD loss (Fig. 8a). To notice, the penalty factor should not be chosen too large since it might affect the model accuracy loss in Eq. 14 and leads to model accuracy degradation. In experiments, we choose $\lambda = 0.35$ for the final evaluation.



(a) Model Performance (F1 for MRPC)



(b) Training loss

Figure 7: Peer distillation ablation study: model performance with different PD loss penalty factors

7 Conclusion

In this work, we propose a novel peer distillation head pruning method. We leverage differentiable pruning to automatically select heads and derive a compact model structure. Then knowledge recycling and peer distillation are proposed to make use of the discarded knowledge represented by the pruned heads. The retained heads exploit the recycled knowledge from the pruned heads and ensure the maintenance of the learned knowledge and thus model accuracy. With the help of recycled knowledge from the pruned heads, higher head sparsity is achieved with a slight loss in accuracy. Our results outperform the state-of-the-art head pruning results and validate the effectiveness of our method.

Acknowledgements

We would like to acknowledge the USDA-NIFA Agriculture and Food Research Initiative (Award No.: 2022-67023-36399 to CD).

References

[An *et al.*, 2020] Bang An, Jie Lyu, Zhenyi Wang, Chunyuan Li, Changwei Hu, Fei Tan, Ruiyi Zhang, Yifan Hu, and Changyou Chen. Repulsive attention: Rethinking multi-head attention as bayesian inference. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 236–255, 2020.

[Boyd *et al.*, 2011] Stephen Boyd, Neal Parikh, and Eric Chu. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[Devlin *et al.*, 2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[Han *et al.*, 2015] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in Neural Information Processing Systems*, 28, 2015.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Hu *et al.*, 2018] Minghao Hu, Yuxing Peng, Furu Wei, Zhen Huang, Dongsheng Li, Nan Yang, and Ming Zhou. Attention-guided answer distillation for machine reading comprehension. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2077–2086, 2018.

[Huang *et al.*, 2022] Shaoyi Huang, Dongkuan Xu, Ian Yen, Yijue Wang, Sung-En Chang, Bingbing Li, Shiyang Chen, Mimi Xie, Sanguthevar Rajasekaran, Hang Liu, et al. Sparse progressive distillation: Resolving overfitting under pretrain-and-finetune paradigm. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 190–200, 2022.

[Hubara *et al.*, 2016] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Advances in neural information processing systems*, pages 4107–4115, 2016.

[Kovaleva *et al.*, 2019] Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. Revealing the dark secrets of bert. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4365–4374, 2019.

[Lee *et al.*, 2020] Hyun Dong Lee, Seongmin Lee, and U Kang. Auber: Automated bert regularization. *arXiv preprint arXiv:2009.14409*, 2020.

[Li *et al.*, 2016] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[Li *et al.*, 2020] Bingbing Li, Zhenglun Kong, Tianyun Zhang, Ji Li, Zhengang Li, Hang Liu, and Caiwen Ding. Efficient transformer-based large scale language representations using hardware-friendly block structured pruning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 3187–3199, 2020.

[Michel *et al.*, 2019] Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, pages 14014–14024, 2019.

[Raganato *et al.*, 2020] Alessandro Raganato, Yves Scherrer, and Jörg Tiedemann. Fixed encoder self-attention patterns in transformer-based machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 556–568, 2020.

[Srinivas *et al.*, 2017] Suraj Srinivas, Akshayvarun Subramanya, and R Venkatesh Babu. Training sparse neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 138–145, 2017.

[Voita *et al.*, 2019] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, 2019.

[Wang *et al.*, 2018] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, 2018.

[Wang *et al.*, 2020] Hanrui Wang, Zhekai Zhang, and Song Han. Spatten: Efficient sparse attention architecture with cascade token and head pruning. *arXiv preprint arXiv:2012.09852*, 2020.

[Wolf *et al.*, 2019] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

[Xiao *et al.*, 2019] Xia Xiao, Zigeng Wang, and Sanguthevar Rajasekaran. Autoprune: Automatic network pruning by regularizing auxiliary parameters. *Advances in neural information processing systems*, 32, 2019.

[Yang *et al.*, 2019] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764, 2019.

[Yin *et al.*, 2019] Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley J. Osher, Yingyong Qi, and Jack Xin. Understanding straight-through estimator in training activation quantized neural nets. In *International Conference on Learning Representations*, 2019.

[Zhang *et al.*, 2021] Zhengyan Zhang, Fanchao Qi, Zhiyuan Liu, Qun Liu, and Maosong Sun. Know what you don't need: Single-shot meta-pruning for attention heads. *AI Open*, 2:36–42, 2021.

[Zhou *et al.*, 2020] Shanglin Zhou, Bingbing Li, Caiwu Ding, Lu Lu, and Caiwen Ding. An efficient deep reinforcement learning framework for uavs. In *2020 21st International Symposium on Quality Electronic Design (ISQED)*, pages 323–328. IEEE, 2020.