AutoAttention: Automatic Attention Head Selection Through Differentiable Pruning

Anonymous ACL submission

Abstract

Multi-head attention is considered as a driving force and key component behind the state-of-art transformer models. However, recent research reveals that there are many redundant heads with duplicated patterns in each layer. In this work, we propose an automatic pruning strategy using differentiable binary gates to remove redundant heads. We relax the binary head pruning problem into a differentiable optimization by employing Straight Through Estimators (STEs), in which the model weights and head-sparse model structure can be jointly learned through back-propagation. In this way, attention heads can be pruned efficiently and effectively. Experimental results on the General Language Understanding Evaluation (GLUE) benchmark are provided using BERT model. We could reduce more than 57% heads on average with zero or minor accuracy drop on all nine tasks and even achieve better results than state-of-the-arts (e.g., Random, HISP, $L0$ Norm, SMP, etc). Furthermore, our proposed method can prune more than 79% heads with only 0.82% accuracy degradation on average. We further illustrate the pruning procedure and parameters change through the head attention visualization and show how the trainable gate parameters determine the head mask and the final attention map.

1 Introduction

Transformer based language models (Devlin et al., 2018; Yang et al., 2019; Bao et al., 2020) have been proven to be highly effective in learning universal language representations and applicable to downstream tasks with slight fine-tuning. Transformer structure can nicely capture the long-term dependencies in natural language but suffers from high computational cost and memory usage.

To downsize the transformer models, different neural network compression techniques have been proposed, such as parameter sharing (Lan et al., 2020; Raganato et al., 2020), knowledge distillation (Sanh et al., 2019; Jiao et al., 2020), weight pruning (Gordon et al., 2020; Li et al., 2020), neuron pruning (Correia et al., 2019; Prasanna et al., 2020) etc. Among the compression approaches, pruning attention heads has been warmly studied due to its contribution to model interpretability and direct computational complexity reduction (Michel et al., 2019; Voita et al., 2019).

The redundancy of the multi-head mechanism was first discussed in (Michel et al., 2019; Kovaleva et al., 2019), and they prune duplicated attention heads manually or in a greedy manner. In order to automatically locate important attention heads in each layer, Gumbel softmax (Voita et al., 2019) and reinforcement learning (Lee et al., 2020) approaches have been employed. Gumbel softmax relaxes the binary optimization problem into a differentiable problem and enables the head-sparse model structure learning in one-shot, while the reinforcement learning approach uses a deep Q network to learn a pruning policy which achieves a better pruning performance with longer search time.

In order to effectively and efficiently slim attention heads, in this work, we propose an automatic single-shot head pruning algorithm AutoAttention by leveraging the differentiable binary gates (which determine the pruning status of attention heads) controlled by Straight Through Estimators (STEs). Comparing with Gumbel softmax sampling, our approach can provide a more direct objective and gradients for optimizing the binary gates, which leads to a better performance in head pruning. Our contributions are as follows:

- We propose differentiable head pruning algorithm to automatically learn head-sparse transformer architectures, while human heuristic methods rely on manual sparsity distribution and are sub-optimal and time-consuming due to the sensitivity of different layers.

- We transform the discrete head pruning problem into a smooth continuous optimization problem with STEs and achieve better results than the state-of-the-arts in both model accuracy and head sparsity. Specifically, we provides direct head regularization and functionally avoid test accuracy drops caused by the train-test discrepancy issue appearing in other differentiable approaches such as Gumbel softmax.

- As the first attempt, we utilize head attention visualization to illustrate the pruning procedure and the parameter change (weight parameter for model accuracy and gate parameter for head pruning) before and after pruning.

- Head functionality analysis is novelly conducted through pruning, in which the head redundancy and head functional variations across different layers are examined and discussed. We hope our work will push forward the explainability of AI.

We evaluate our AutoAttention on nine GLUE benchmark tasks (Wang et al., 2018). Experimental results show that we achieve high compression rates with zero or minor accuracy degradation. The pruned models outperform the original ones with only 42.75% heads left on average. In extreme cases, our proposed method can prune 79.74% heads with only 0.82% accuracy degradation on average and prune 99.3% (143 of the 144) heads without accuracy degradation on WNLI 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 Preliminaries

Related Works. The redundancy of the multi-head mechanism has been discovered and investigated in many literatures. (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. (Raganato et al., 2020) proposes to set attention unit head size to input sequence length, and independent of the number of heads. (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 sensitivity based 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 total search time can be comparatively long. (Wang et al., 2020) proposes a token-head sparsification co-design algorithm powered by a specially designed *top-k engine* where quantization is also applied to achieve best hardware performance. 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.

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 as follows

$$Attention(Q, K, V) = Softmax(\frac{Q \times K^T}{\sqrt{D_k}})V \tag{1}$$

where D_k represents 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:

$$\begin{aligned} H_i &= Attention(Q_i, K_i, V_i) \\ &= Attention(X * W_i^Q, X * W_i^K, X * W_i^V) \end{aligned} \tag{2}$$

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

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

3 Differentiable Head Pruning

In this section, we propose AutoAttention, a differentiable method for head pruning. Unlike pruning methods with hard constraints (Han et al., 2015; Boyd et al., 2011; Li et al., 2016; Zhang et al., 2020), AutoAttention obtains model sparsity by updating the gate parameters. This leads to two

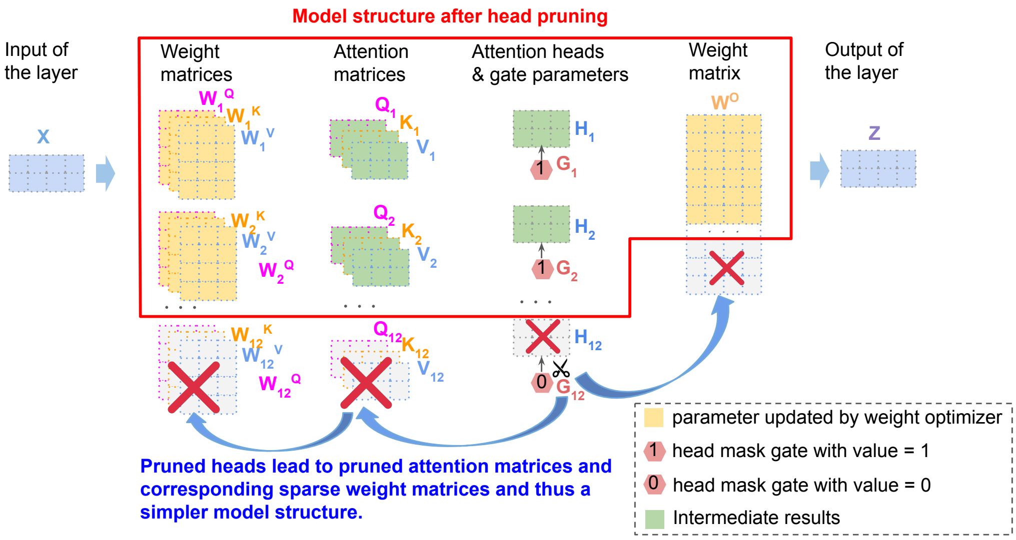


Figure 1: Differentiable gate pruning to remove redundant heads (illustrated using one attention layer of BERT)

important benefits: first, we do not have to set expected sparsity for each layer and can obtain model sparsity automatically; second, since the pruning process prune the model more "smoothly", the accuracy degradation is not significant. This makes the retraining process not a necessity and leads to faster training convergence.

3.1 AutoAttention: Differentiable Gated Head Pruning

In order to achieve sparse attention heads, we introduce attention head penalization into the loss function. Let $F(\cdot)$ be the accuracy loss function of the transformer model with model weight W. The head pruning problem can be formulated as:

$$\min_{W} F(W) + \mu \cdot ||H||_0, \tag{4}$$

where μ is the penalty factor and $||H||_0$ denotes 0-norm of the heads, representing the number of un-pruned attention heads in the transformer model. The optimization objective is to remove the redundant heads while maintaining the model performance. As illustrated in Fig. 1, we introduce attention head masking gates G, in which G is composed of lists of binary variables, representing the status of their corresponding heads:

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

where i and j denote the index of head and attention layer, respectively, and G_{ij} denotes the pruning status of the head. By combining Eq. 4 and Eq. 5, we can reformulate the head pruning problem as:

$$\min_{W,G} F(W,G) + \mu ||H \odot G||_0, \tag{6}$$

Considering the binary variables in G, the equation above can be simplified as:

$$\min_{W,G} F(W,G) + \mu \cdot \sum G \tag{7}$$

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

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 masking gates G.

$$G_{ij} = g(W'_{ij}) = \begin{cases} 0, & \text{if } W'_{ij} \le 0 \\ 1, & \text{if } W'_{ij} > 0. \end{cases} \tag{8}$$

where the binary masking gate G_{ij} is represented as a step function g with a continuous auxiliary parameter W'_{ij}. Combining with Eq. 7, the problem can be re-formulated as:

$$\min_{W,W'} \mathcal{L} = \min_{W,W'} F(W,W') + \mu \cdot \sum g(W'), \tag{9}$$

where W' are lists of auxiliary parameters with the same size of the attention heads which *control* the *open* and *close* of the binary gates.

The model weight W can be updated through back-propagation as $W_{k+1} = W_k - l_r * \frac{\partial F}{\partial W}$, where l_r is the learning rate of the weight optimizer.

To update the sparse head structure, coarse gradients (Hubara et al., 2016) are introduced in STEs 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. Linear STEs have been applied to neural network weight pruning in (Srinivas et al., 2017). ReLU or clipped ReLU STEs have been proved to be unbiased estimators (Yin et al., 2019). Softplus STEs are recommended in (Xiao et al., 2019) due to the smoothness of their gradients, where the curse of non-recoverability in network pruning caused by zero gradients is also discussed. We use Softplus STE and the auxiliary parameter W' can be updated as:

$$\begin{aligned} W'_{k+1} &= W'_k - l'_r * \frac{\partial \mathcal{L}}{\partial W'} \\ &= W'_k - l'_r * \frac{\partial \mathcal{L}}{\partial G} * \frac{\partial G}{\partial W'} \\ &= W'_k - l'_r * \frac{\partial \mathcal{L}}{\partial G} * Softplus(W') \end{aligned} \tag{10}$$

where l'_r is the learning rate of the gate optimizer.

3.2 Differentiable Head Pruning Method Comparison

Differentiable pruning methods enable one-shot training in which the weights and model structures are learned jointly through back-propagation and approximated related $L0$ regularization methods have been designed. Besides using STEs in AutoAttention, Gumbel softmax has also been introduced in head pruning (Voita et al., 2019).

In (Voita et al., 2019), the $L0$ norm is stochastically relaxed, in which each gate g is represented by a random variable drawn from Hard Concrete (aka Gumbel softmax) distributions (Louizos et al., 2018). The Hard Concrete distribution belongs to a parameterized family of mixed discrete-continuous distributions over [0, 1] and the non-zero probability mass at 0 can be described as:

$$P(g = 0|\phi) \tag{11}$$

where ϕ is the distribution parameter. The relaxed $L0$ norm penalization term is formulated as:

$$L_c(\phi) = \sum (1 - P(g = 0|\phi)) \tag{12}$$

and the entire head pruning objective function is

$$\min_{W,\phi} F(W, \phi) + \mu \cdot L_c(\phi) \tag{13}$$

where W is the model weights and $F(W, \phi)$ is the general accuracy loss function. We can solve this optimization problem through back-propagation with re-parameterization trick (Kingma and Welling, 2013) to calculate the gradients for ϕ.

The differentiability of structure search is through approximated Gumbel softmax parameterized by ϕ in (Voita et al., 2019), and through STEs parameterized by W'. However, (Voita et al., 2019) also introduces discrepancy between original complete network and the pruned sub-network during the model evaluation procedure, in which gate values (0 or 1) depend on which of the values $P(g_i = 0|\phi_i)$, $P(g_i = 1|\phi_i)$ is larger. Thus, there exist a un-avoided gap between model training and model testing. Fortunately, similar to (Voita et al., 2019), our AutoAttention method applies smooth and differentiable optimization to pruning task, but the discrepancy is largely avoided by directly optimizing binary gates.

4 Evaluation

Datasets. We test our method on GLUE benchmark. It consists of 9 tasks and covers a diverse range of dataset sizes, text genres, and degrees of difficulty (Wang et al., 2018). More specifically, we conduct tests on the Corpus of Linguistic Acceptability (CoLA) (Warstadt et al., 2018) dataset for single-sentence tasks, the Stanford Sentiment Treebank (SST-2) (Socher et al., 2013) for movie review classification, the Microsoft Research Paraphrase Corpus (MRPC) (Dolan and Brockett, 2005), the Semantic Textual Similarity Bench-mark (STS-B) (Cer et al., 2017), and the Quora Question Pairs (QQP) (Chen et al., 2018) for paraphrase similarity matching tasks, and the Multi-Genre Natural Language Inference Corpus (MNLI) (Williams et al., 2018), the Question-answering NLI (QNLI) (Wang et al., 2018), the Recognizing Textual Entailment (RTE) (Wang et al., 2018), and the Winograd NLI (WNLI) (Levesque et al., 2012) for inference tasks.

Pre-trained Model and Evaluation Metrics. Our pre-trained model is the BERT$_{\text{BASE}}$ (Devlin et al., 2018), which consists of 12 attention layers and 12 heads for each layer. Following (Wang et al., 2018), we use accuracy for SST-2, QNLI, MNLI, QQP, RTE and WNLI; Matthews Correlation Co-

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

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	-
$\boldsymbol{L0}$ 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	-
AutoAttention (ours)	**83.66**	**91.07**	**91.25**	**92.89**	**60.39**	**86.94**	**88.62**	65.7	**56.34**

Table 2: Comparison of evaluation accuracy using our gate head pruning methods 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.
$\text{BERT}_{\text{BASE}}$	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%	41.67%	44.44%	99.3%	57.25%
AutoAttention prune	83.87	**91.3**	**91.38**	**92.89**	**60.39**	**86.94**	**89.52**	**67.87**	**56.34**	

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

Models	MNLI	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	WNLI	Ave.
$\text{BERT}_{\text{BASE}}$	83.9	91.2	91.1	92.7	53.4	85.8	88.9	66.4	56.3	
AutoAttention prune	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%**

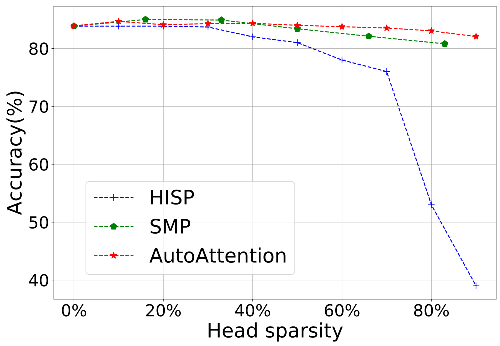


Figure 2: Model performance regarding to different head sparsity on MNLI-matched dataset

efficient (MCC) for CoLA, F1 scores for MRPC, and Spearman for STS-B.

We define the head sparsity as:

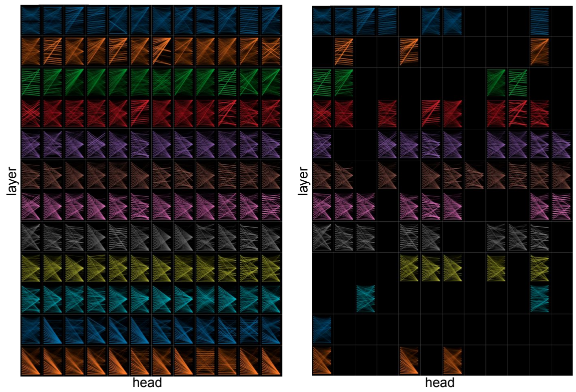
$$head\_sparsity = \frac{\#pruned\_heads}{\#model\_heads} \tag{14}$$

Implementation Details. We follow the default finetuning steps for 9 tasks according to Huggingface (Wolf et al., 2019) and obtain the baseline models after training for 4 epochs. Then, gate pruning is executed for the whole models. For weight and gate pruning, we use different optimizers and select different learning rates to achieve better balance between accuracy and head sparsity.

Baselines. To validate the effectiveness of our proposed method, we introduce four baselines. In Method **Random**, 50% heads are randomly selected to prune. We report the results 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 combined.

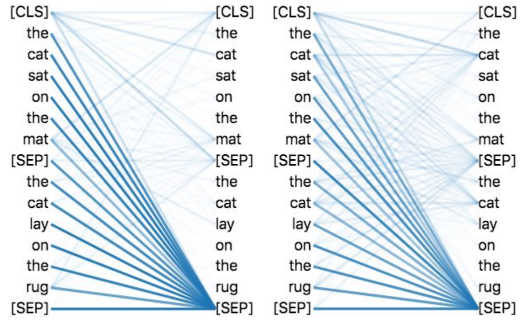
Experimental Results. We show our result comparisons in Table 1. For fairness, we compare our head pruning model accuracy with state-of-the-art algorithms with fixed global head sparsity of 50%. Comparing with Random method, our AutoAttention enjoys better model accuracy thanks to the head penalization to find the redundant heads. HISP (Michel et al., 2019) calculates head importance for pruning and suffers significant accuracy drop since the importance is not estimated directly according to the final model performance. Comparing with $L0$ Norm approach (Voita et al., 2019), AutoAttention outperforms it with a large margin in all existing 7 GLUE benchmark tasks, which practically proves that our proposed STE based head pruning approach better avoids the discrepancy of the learned model structure between model training and model testing. 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, AutoAttention takes the lead in 7 of the 8 tasks. Fig. 2 shows the performance of different pruning methods. While increasing head sparsity, AutoAttention can achieve more than 80% sparsity with only 1.03% accuracy drop and outperform 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 57% 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 1 head left). For different tasks, we investigate the limit of our AutoAttention method and obtain the head sparsity in extreme cases (within 1% accuracy drop). As shown in Table 3, we could prune 79.74% heads with 0.82% accuracy drop on average. For one self-attention module, the memory is reduced from 9.43 MB to 1.98 MB and the FLOPs from 100.9 to 21.2 million.



(a) Before Pruning (b) After Pruning

Figure 3: Attention heads before and after pruning on CoLA dataset with BERT model



(a) Single head before pruning (b) Single head after pruning

Figure 4: Head before (left) and after (right) gate pruning on CoLA dataset with BERT model (corresponding to the detail views of the 5th head of 12th layer in Fig. 3(a) and Fig. 3(b))

5 Head Distribution Discussion

5.1 Head Pruning Visualization

We use bertviz tool (Vig, 2019) to obtain the head attention maps. Fig. 3 show the attention map before and after pruning. In Fig. 4, we show the detail of a single head of the unpruned and pruned models corresponding to the 5th head of 12th layer of Fig. 3(a) and Fig. 3(b), respectively. After pruning redundant heads, the values of attention weight matrix changes slightly, which leads to the slight change of the head attention map. This illustrates that different optimizers (weight and gate optimizers) work simultaneously by changing different parameters (weight and gate parameters) in the training loop towards the improvement of model accuracy and head sparsity.

5.2 Head Functionality vs. Pruning

The role of the attention heads varies in different downstreaming tasks. Our results show that BERT model can be over-parameterized for a specific task in GLUE and head pruning actually controls model complexity and regularize the learning process, where AutoAttention automatically prunes a proper number of redundant heads by directly penalizing the head-cardinality and achieves a better fine-tune performance. In Table 3, for some less complex tasks such as WNLI, we can achieve over 99% head sparsity, while for comparatively complex and data hungry tasks such as RTE and MRPC, our derived model keeps more heads un-pruned, which further reflects the consistency between task complexity and model complexity.

We discover natural head redundancy differences and potentially head functional differences across different layers in model fine-tuning. As shown in Fig. 3(b), more heads in the last several layers are pruned. In other words, the last several layers are experiencing greater attention head structural changes during the pruning incorporated fine-tune process. This evidence conceptually matches the

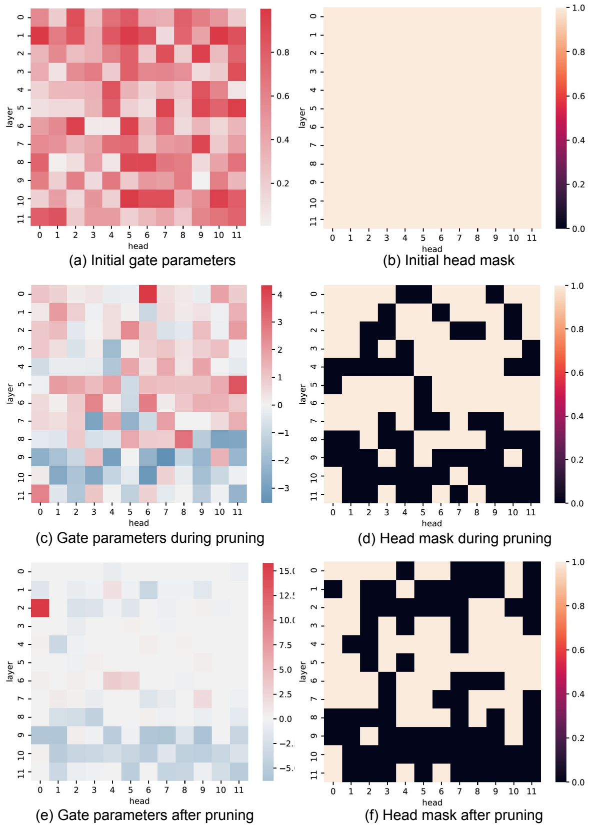


Figure 5: Head mask change during pruning on CoLA dataset with BERT model

discovery in (Kovaleva et al., 2019) which compares the cosine similarity of the flattened layer-wise attention weights between pre-trained and fine-tuned BERT model. Similarly, the attention weights of last several layers change the most, in which further indicates that last several layers encode more task-specific information while the earlier layers are mainly providing comparatively general low-level representations.

5.3 Heads Distribution During Pruning

Different heads have different functionalities and thus have different level of importance for the pruned model. The importance calculation and ranking of attention heads could benefit: a) the pruning process by removing the less important heads, b) the model structure design by arranging different heads in different layers, and c) the interpretability of the multi-heads mechanism and even deep neural networks.

Different from the quantified importance scores of heads (Michel et al., 2019) for pruning, we use learnable gate parameters to determine the retention of the heads. If the gate parameter is larger than 0, the corresponding head will be retained. Otherwise, the head will be pruned based on Eq. 2.

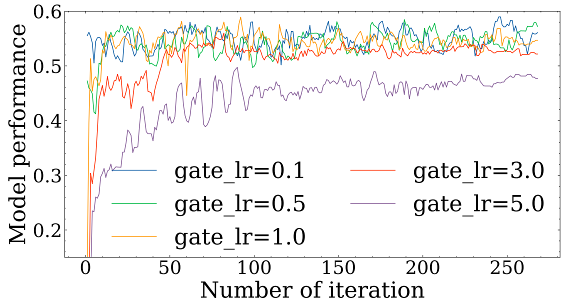
Comparing with the static head pruning method through attention head importance ranking (Michel et al., 2019), AutoAttention enables a larger head structure search space and a more direct pruning objective through automatic differentiable head structure learning. Fig. 5 shows the update process of the auxiliary parameter W' and head masking gate status $g(W')$ of the 144 attention heads jointly trained with the model weights W. We present their values changes in three different training stages:

- In the initialization stage, the auxiliary parameters are initialized by following truncated normal distribution with all values greater than 0 (shown in Fig. 5(a)) and all the corresponding pruning gates are open in (shown in Fig. 5(b)). In this way, our AutoAttention starts with the full number of unpruned attention heads.

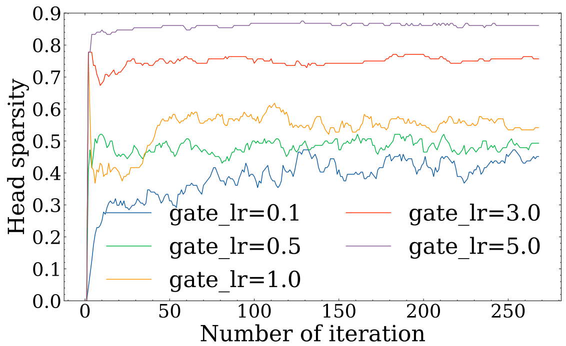
- In the intermediate stage, with continuous penalizing the total number of opened gates in the loss function in Eq. 9, the gate auxiliary parameters corresponding to less important heads are receiving negative gradients. After epochs of training, part of the auxiliary parameter values are dropping below zero (denoted in cold colored boxes in Fig. 5(c)), which leads to the closure of the corresponding gates and the pruning of the attention heads (denoted in dark boxes in Fig. 5(d)).

- In the final stage, in Fig. 5(e), more auxiliary parameter values drop below zero which leads to a higher pruning ratio of the attention heads. The optimization converges when the model accuracy component and sparsity component in the objective function Eq. 9 are competing with each other, in which a more head-sparse transformer model structure is difficult to be learned without largely sacrificing the model accuracy.

More importantly, comparing Fig. 5(d) and Fig. 5(f), not all of the intermediately pruned heads remain pruned in the final stage, which proves the recoverability of our differentiable pruning approach. When some temporally less important attention heads are later discovered to be important according to the current model states, the closed gates will be re-opened through automatic promoting their corresponding auxiliary parameter values through differentiable training. In this way, the model head structure is updated together with

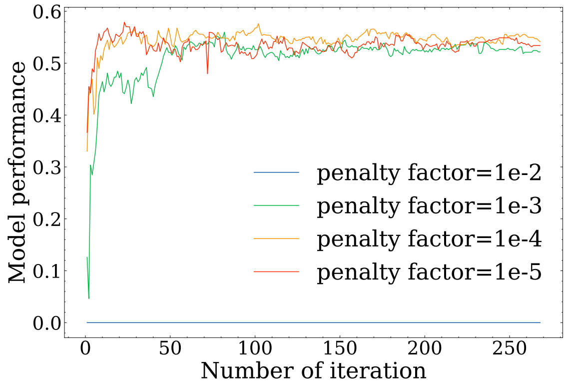


(a) Model Performance (mcc for CoLA)

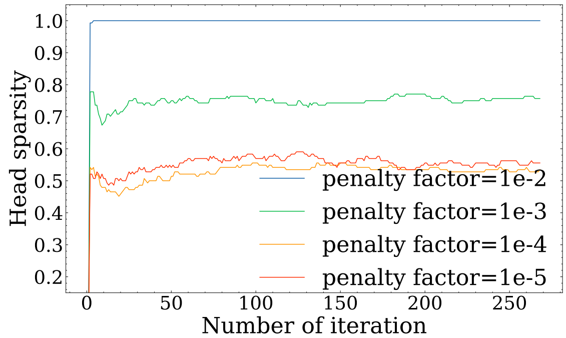


(b) Head Sparsity

Figure 6: Ablation study: gate pruning optimizer with different learning rates on CoLA dataset.



(a) Model Performance (mcc for CoLA)



(b) Head Sparsity

Figure 7: Ablation study: gate pruning optimizer with different penalty factors on CoLA dataset.

the model weights automatically through back-propagation to potentially locate a better local optima with larger searching space.

6 Ablation Study

In this section, we perform ablation study over several hyper-parameters when doing automatic gate pruning with BERT model.

Gate Pruning Learning Rate. To solve the optimization problem in Eq. 7, different optimizers are utilized to update weight and gate parameters. To update W, we use the default initial learning rate (3e-5). For the update of W', larger initial learning rate leads to faster convergence and higher sparsity. While increasing the initial learning rate from 0.1 to 5.0, we could increase the sparsity from 53% to 79% with only 0.03 performance (mcc for CoLA dataset) drop as shown in Fig. 6. We observe the obvious compete between accuracy and sparsity increase since weight and gate optimizers tend to reduce the loss function in different directions.

Penalty Factor. The penalty factor, μ, in Eq. 7 can be chosen to change the balance between the prediction loss and sparsity loss. Larger μ means the higher penalty for sparsity and could leads to the higher model sparsity. As shown in Fig. 7, We test different μ from 1e-2 to 1e-5. When μ is less than 1e-3, larger μ leads to larger sparsity. When μ is larger than 1e-2, we observe the sudden model performance drop. In our tests, we fix the penalty factor as 1e-3 and adjust the gate pruning learning rate to obtain higher model sparsity. , since the gate optimizer is much more robust to find a better local optima than changing the total loss function.

7 Conclusion

In this work, we propose a novel automatic differentiable head pruning method. We reform the pruning loss function with the $L0$ regularizer applied to attention heads by utilizing straight through estimators (STEs). Then the differentiable optimization solution is proposed by designing separate optimizers to update weight parameter and gate parameter (which determines the pruning status of attention heads). We significantly remove the attention head redundancy and visualize the detail information (head pruning status, model weight update, and model attention map) before and after pruning. Our results outperform the state-of-the-art pruning results and validate the effectiveness of our method.

References

Bang An, Jie Lyu, Zhenyi Wang, Chunyuan Li, Changwei Hu, Fei Tan, Ruiyi Zhang, Yifan Hu, and Changyou Chen. 2020. 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.

Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Jianfeng Gao, Songhao Piao, Ming Zhou, et al. 2020. Unilmv2: Pseudo-masked language models for unified language model pre-training. In *International Conference on Machine Learning*, pages 642–652. PMLR.

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

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Zihan Chen, Hongbo Zhang, Xiaoji Zhang, and Leqi Zhao. 2018. Quora question pairs. *URL https://www. kaggle. com/c/quora-question-pairs*.

Gonçalo M Correia, Vlad Niculae, and André FT Martins. 2019. Adaptively sparse transformers. 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 2174–2184.

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

Bill Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*. Asia Federation of Natural Language Processing.

Mitchell Gordon, Kevin Duh, and Nicholas Andrews. 2020. Compressing bert: Studying the effects of weight pruning on transfer learning. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 143–155.

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

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

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling bert for natural language understanding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 4163–4174.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. 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.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

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

Hector Levesque et al. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

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. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 3187–3199.

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

Christos Louizos, Max Welling, and Diederik P Kingma. 2018. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*.

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

Sai Prasanna, Anna Rogers, and Anna Rumshisky. 2020. When bert plays the lottery, all tickets are winning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3208–3229.

Alessandro Raganato, Yves Scherrer, and Jörg Tiedemann. 2020. 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.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

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

Jesse Vig. 2019. A multiscale visualization of attention in the transformer model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 37–42, Florence, Italy. Association for Computational Linguistics.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. 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.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. 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.

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

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2018. Neural network acceptability judgments.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

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

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

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

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

Tianyun Zhang, Xiaolong Ma, Zheng Zhan, Shanglin Zhou, Minghai Qin, Fei Sun, Yen-Kuang Chen, Caiwen Ding, Makan Fardad, and Yanzhi Wang. 2020. A unified dnn weight compression framework using reweighted optimization methods. *arXiv preprint arXiv:2004.05531*.

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