Structured Pruning for Efficient Generative Pre-trained Language Models

Chaofan Tao[1], **Lu Hou**[2], **Haoli Bai**[2], **Jiansheng Wei**[2],
Xin Jiang[2], **Qun Liu**[2], **Ping Luo**[1], **Ngai Wong**[1]
[1]The University of Hong Kong [2]Huawei Noah's Ark Lab
cftao@connect.hku.hk, pluo@cs.hku.hk, nwong@eee.hku.hk
{houlu3, baihaoli, weijiansheng, jiang.xin, qun.liu}@huawei.com

Abstract

The increasing sizes of large generative Pre-trained Language Models (PLMs) hinder their deployment in real-world applications. To obtain efficient PLMs, previous studies mostly focus on pruning the attention heads and feed-forward networks (FFNs) of the Transformer. Nevertheless, we find that in generative PLMs, the hidden dimension shared by many other modules (e.g., embedding layer and layer normalization) contains persistent outliers regardless of the network input. In this study, we propose SIMPLE, a new structured pruning framework for generative PLMs that comprehensively investigates all the above compressible components. To identify redundant network structures, we assign learnable masks over compressible components followed by sparse training. Various sizes of PLMs can be flexibly extracted via different thresholds, and are then task-specifically fine-tuned for further improvement. Extensive experiments on language modeling, summarization and machine translation validate the effectiveness of the proposed method. For example, the pruned BART brings 1.51x/6.96x inference speedup on GPU/CPU with 67% size reduction, and can be further combined with quantization for more than 25× compression.

1 Introduction

Large-scale generative pretrained language models (PLMs) (Radford and Narasimhan, 2018; Brown et al., 2020; Lewis et al., 2020; Raffel et al., 2020) show remarkable performance on various tasks. However, their increasing sizes also lead to expensive memory and computation, hindering their deployment in real applications.

Recent attempts (Tao et al., 2022; Frantar et al., 2022; Dettmers et al., 2022; Xiao et al., 2022; Wang et al., 2021) propose to compress generative PLMs models by quantization. However, hardware-dependent low-bit kernels need to be specially developed for real inference speedup. Compared to quantization, structured pruning methods prune parts of the model structures without requiring designing extra operators to achieve inference speedup and run-time memory saving. Recently, Anonymous (2023) show that the feed-forward networks (FFNs) of GPT models can be pruned to smaller widths, and Li et al. (2022) compress the BART models by combining layer pruning and model quantization for a higher compression rate. However, these models consider only limited components for pruning, i.e., the FFNs or Transformer layers, which can be restrictive for various deployment requirements.

In this work, we propose a new structured pruning framework named SIMPLE (**S**parsity-**I**nduced **M**ask learning for **P**runing generative pre-trained **L**anguage mod**E**ls), which offers a wider range of compressible components. Aside from the attention heads and the width of FFNs commonly used for structured pruning of discriminative PLMs, we also propose to prune the hidden dimension, to further push the trade-off between performance and model size. It is motivated by the observation that persistent outliers exist in the hidden dimension of both decoder-only and encoder-decoder generative PLMs. The observation implies that the hidden dimension may be slimmed sharply with a slight performance drop. Additionally, as the dimension of the hidden state is shared by many modules, e.g., embedding layer, attention heads, FFN and layer normalization, the model size thus can be collectively slimmed for further compression.

The crux of pruning lies in ranking the importance of different compressible components. Towards that end, we assign learnable masks over the output of all compressible components. These masks are optimized in an end-to-end fashion together with a sparsity-induced penalty. Unlike conventional pruning criteria based on magnitudes (He et al., 2018) or gradients (Voita et al., 2019), these masks can be mathematically interpreted to prune

Findings of the Association for Computational Linguistics: ACL 2023, pages 10880–10895

away components shrinking towards zero during the training. Moreover, the learning of masks is one-shot, i.e., one can flexibly obtain various pruned models by different thresholds over the learned mask values. To mitigate the accumulated compression error, a causal distillation objective is proposed to fine-tune the pruned networks for further improvement on downstream tasks.

We verify the efficacy of SIMPLE on various generation tasks (i.e., language modeling, summarization and machine translation) and network architectures (i.e., GPT2 and BART/mBART). Empirical results show that the proposed SIMPLE outperforms other pruning methods, significantly speeds up the inference, and can be combined with quantization for more aggressive compression.

2 Related Work

2.1 Structured Pruning of Transformers

Pruning away unimportant parameters is a widely-used approach to compress neural networks. Unlike unstructured pruning which removes individual weights, structured pruning prunes parts of network structures, and achieves inference speedup/runtime memory saving without designing extra operators. Two commonly used structured components in Transformers are attention heads and FFN neurons (Michel et al., 2019; Hou et al., 2020; McCarley, 2019; Anonymous, 2023). Depending on how the importance of the compressible components is determined, current structured pruning methods for Transformer based methods can be divided into two categories. The first category uses heuristics to calculate the importance. For instance, magnitude-based method (He et al., 2018) uses the component's magnitude as its importance, while the loss-aware method (Voita et al., 2019) measures the importance of a component based on the variation in the training loss if it is removed (Hou et al., 2020; Michel et al., 2019). The other category considers the changes in the component's weights during training, and the importance is not determined until the training is finished. For instance, movement pruning (Sanh et al., 2020; Lagunas et al., 2021) captures the changes in the weights during fine-tuning as the dynamic importance metric and prunes weights that shrink during fine-tuning.

2.2 Compression of Generative PLMs

In contrast to the extensive research on compressing the discriminative PLMs like BERT (Sanh et al., 2019; Lan et al., 2019; Jiao et al., 2020), the study on the compression of Generative PLMs still at its early stages. Early attempts try to compress generative PLMs by tensor decomposition (Edalati et al., 2021) and knowledge distillation (Song et al., 2020), but suffer from severe performance drop. Some recent attempts propose to apply weight quantization (Tao et al., 2022; Frantar et al., 2022; Dettmers et al., 2022; Xiao et al., 2022) to reduce the storage and run-time memory of generative PLMs, but require hardware-dependent low-bit quantization kernels to achieve inference speedup. Recently, DQ-BART (Li et al., 2022) combines weight quantization and layer pruning to compress BART models on abstractive summarization tasks. GUM (Anonymous, 2023) proposes to prune the neurons in the feed-forward network of GPT models based on importance measured by uniqueness and sensitivity. In addition to these neurons, our proposed SIMPLE also prunes the attention heads and the hidden state dimension, to better explore the trade-off between performance and model size.

3 Methodology

In this section, we elaborate on the proposed SIMPLE for structurally pruning generative PLMs. In Section 3.1, we include the hidden dimension as compressible components in addition to the attention heads and FFN neurons, due to the existence of persistent outliers. Then, we introduce sparse learning of masks together with their mathematical interpretations in Section 3.2. The structurally pruned models are followed by task-specific fine-tuning, as detailed in Section 3.3. An overall framework of SIMPLE is depicted in Figure 1.

3.1 Pruning Granularities

3.1.1 Preliminary on Pruning PLMs

A standard Transformer layer has a multi-head attention (MHA) layer and a feed-forward network (FFN). Previous works (McCarley, 2019; Hou et al., 2020) show that the width of a Transformer layer can be reduced by pruning the heads of MHA and the neurons in the intermediate layer of FFN.

Specifically, suppose the input of MHA is $\mathbf{X} \in \mathbb{R}^{n\times d}$ where n and d are the sequence length and hidden state size, respectively. The computation of the MHA can be reformulated as the summation of all N_H attention heads(Hou et al., 2020). For the h-th head, denote its projection matrices as $\mathbf{W}_h^Q, \mathbf{W}_h^K, \mathbf{W}_h^V, \mathbf{W}_h^O \in \mathbb{R}^{d\times d_h}$ where

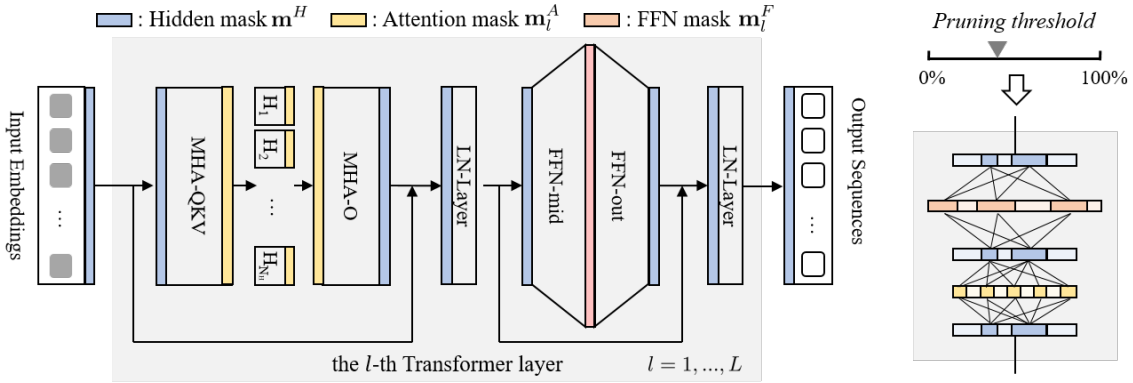


Figure 1: The left part shows the overall framework of the proposed SIMPLE. The stripes with the same color represent the pruning dimension shared with the same mask, i.e., the hidden dimension (blue), the attention head (yellow) and the intermediate dimension of FFN layers (red). The right part demonstrates that after sparse learning of masks, one can flexibly extract various Transformer sizes with different pruning thresholds on the mask.

$d_h = d/N_H$. The softmax function is applied to the scaled dot product of queries and keys to get attention scores $\text{Attn}^h_{\mathbf{W}_h^Q,\mathbf{W}_h^K,\mathbf{W}_h^V,\mathbf{W}_h^O}(\mathbf{X}) = \text{Softmax}(\frac{1}{\sqrt{d}}\mathbf{X}\mathbf{W}_h^Q\mathbf{W}_h^{K\top}\mathbf{X}^\top)\mathbf{X}\mathbf{W}_h^V\mathbf{W}_h^{O\top}$. Suppose the mask of attention heads is $\mathbf{m}_l^A \in \mathbb{R}^{N_H}$ for the l-th layer, the output of the MHA is:

$$\text{MHAttn}_{\mathbf{W}^Q,\mathbf{W}^K,\mathbf{W}^V,\mathbf{W}^O}(\mathbf{X}) = \sum_{h=1}^{N_H}[\mathbf{m}_l^A]_h \times \text{Attn}^h_{\mathbf{W}_h^Q,\mathbf{W}_h^K,\mathbf{W}_h^V,\mathbf{W}_h^O}(\mathbf{X}). \quad (1)$$

We use $\mathbf{m}^A = \{\mathbf{m}_l^A\}_{l=1}^L$ to denote the masks of MHA over all L Transformer layers.

For FFN, denote d_{ff} as the number of neurons in the intermediate layer of FFN, and the weights and biases of two linear layers are $\mathbf{W}^1 \in \mathbb{R}^{d\times d_{ff}}, \mathbf{b}^1 \in \mathbb{R}^{d_{ff}}$ and $\mathbf{W}^2 \in \mathbb{R}^{d_{ff}\times d}, \mathbf{b}^2 \in \mathbb{R}^d$, respectively. The output of FFN can also be reformulated as the summation of computations of d_{ff} neurons. With a slight abuse of notation, we still use $\mathbf{X} \in \mathbb{R}^{n\times d}$ to denote the input of FFN. Suppose the mask of these neurons is $\mathbf{m}_l^F \in \mathbb{R}^{d_{ff}}$ for the l-th Transformer layer, the output of the FFN is computed as:

$$\text{FFN}_{\mathbf{W}^1,\mathbf{W}^2,\mathbf{b}^1,\mathbf{b}^2}(\mathbf{X}) = \sum_{i=1}^{d_{ff}}[\mathbf{m}_l^F]_i \times \text{Act}(\mathbf{W}^1_{:,i}+b_i^1)\mathbf{W}^2_{i,:}+\mathbf{b}^2. \quad (2)$$

We use $\mathbf{m}^F = \{\mathbf{m}_l^F\}_{l=1}^L$ to denote the masks of FFNs over all L Transformer layers.

3.1.2 Persistent Outliers and Hidden Dimension Pruning

In McCarley (2019), it is shown that for discriminative BERT-like models, the cross-layer coupling caused by skip connections makes the hidden dimension d is far more difficult to prune than the attention heads or the FFN neurons.

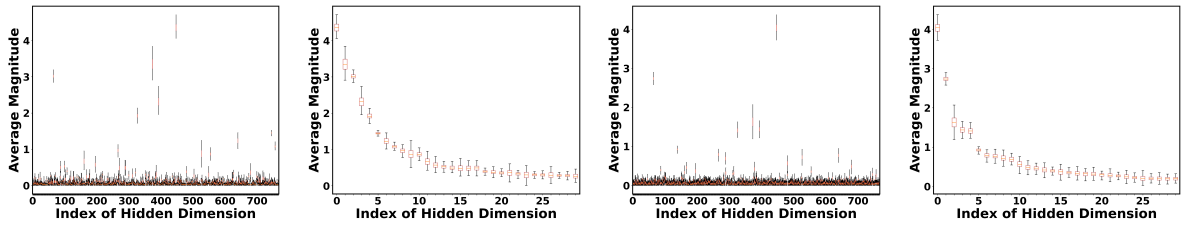
However, as shown in Figure 2 and Figure 7 in Appendix B.3, for both decoder-only and encoder-decoder generative PLMs, large-magnitude outliers appear only in a small fraction of the hidden dimensions. Moreover, if one dimension has an outlier, it almost persistently appears for all tokens, i.e., the variance of all tokens in a particular index is often smaller than the variance among different indices. Similar observations are also found in (Xiao et al., 2022; Dettmers et al., 2022), and are used to guide weight quantization, *i.e.*, assign the data in outlier dimension with high bit-width. If the magnitude is used as the importance metric (i.e., magnitude-based pruning methods) (Gordon et al., 2020; Shen et al., 2022), the high sparsity of these outliers indicate that a large fraction of the hidden dimensions can be pruned away.

Similar to the attention heads and the FFN neurons, we also set a mask on the hidden dimension. Since the hidden dimensions at adjacent residual blocks are connected with an identity skip connection, we use a shared mask $\mathbf{m}^H \in \mathbb{R}^d$ across all Transformer layers. For ease of notation, we use $\mathbf{h}$ as a general term denoting the output of $\text{MHAttn}(\mathbf{X})$ in Eq.(1), $\text{FFN}(\mathbf{X})$ in Eq.(2), the embedding layer and layer normalization layer. The output $\hat{\mathbf{h}} \in \mathbb{R}^d$ of each hidden dimension is then computed as

$$\hat{\mathbf{h}} = \mathbf{m}^H \times \mathbf{h}. \quad (3)$$

3.2 Sparsity-induced Mask Learning

After setting these masks over the compressible components (i.e., attention heads, FFN neurons and hidden dimension). Determining which component to prune away is then equivalent to ranking their learned mask values. To reduce the ac-



(a) Layer 1 (all). (b) Layer 1 (top-30). (c) Layer 6 (all). (d) Layer 6 (top-30).

Figure 2: Boxplots of the magnitudes of the hidden dimensions in Layer 1 and 6 of a 12-layer GPT-2 fine-tuned on the dataset PTB. For each layer, we show both (i) magnitudes of all 768 dimensions; and (ii) the top-30 magnitudes. The other layers follow similar patterns.

curacy drop caused by pruning, it is desired to learn sparse masks. In this section, we propose a sparsity-inducing objective to learn the masks. For a pruned student model, we use the cross-entropy loss (Hinton et al., 2015) between teacher and student model's output logits to guide the learning of these masks. Specifically, for the i-th token t_i, suppose the logits of the student and teacher network are $\mathbf{z}^s_{t_i}, \mathbf{z}^t_{t_i} \in \mathbb{R}^{|V|}$, the loss is

$$\ell = -\sum\nolimits_{i=1}^{n} \mathbf{z}^t_{t_i} \log(\mathbf{z}^s_{t_i}). \tag{4}$$

Denote the learnable masks as $\mathbf{m} = \{\mathbf{m}^A, \mathbf{m}^F, \mathbf{m}^H\}$. Inspired by (Liu et al., 2017; Chavan et al., 2022), we use the ℓ_1 regularizer over $\mathbf{m}$ to promote sparsity, and the optimization objective is:

$$\ell_{mask} = \ell + \lambda \|\mathbf{m}\|_1. \tag{5}$$

The learnable mask values are initialized to 1, and is updated with gradient descent during training. After learning the mask, these masks are binarized according to a threshold determined by a given sparsity (Section 3.3). $\lambda > 0$ is a penalty parameter that controls the weights of the two loss terms

The learning stage is computation-efficient as *we only need to learn the masks once*, and then we can extract sub-networks given any required sparsity, by simply binarizing masks according to the desired threshold.

Method Interpretation. Intuitively, consider one attention head with output $\hat{\mathbf{h}} = m\mathbf{h}$, the gradient with respect to its mask m[1] can be computed as $\frac{\partial \ell_{mask}}{\partial m} = \frac{1}{m}\mathbf{h}^\top \frac{\partial \ell}{\partial \mathbf{h}} + \lambda \mathbf{sign}(m)$. For $m > 0$, the magnitude of m is increasing when $\frac{\partial \ell_{mask}}{\partial m} < 0$; while for $m < 0$, the magnitude of m is increasing when $\frac{\partial \ell_{mask}}{\partial m} > 0$. Thus the magnitude of m increases if

$$\begin{cases} \frac{1}{m}\mathbf{h}^\top \frac{\partial \ell}{\partial \mathbf{h}} + \lambda < 0, & m > 0 \\ \frac{1}{m}\mathbf{h}^\top \frac{\partial \ell}{\partial \mathbf{h}} - \lambda > 0, & m < 0 \end{cases}$$

Then we get

$$\mathbf{h}^\top \frac{\partial \ell}{\partial \mathbf{h}} < -\lambda |m| < 0, \tag{6}$$

which means $\frac{\partial \ell}{\partial h_i} h_i < 0$ dominates in the term $\frac{\partial \ell}{\partial \mathbf{h}}\mathbf{h} = \sum_{i=1}^{n} \frac{\partial \ell}{\partial h_i} h_i$. Note that $\frac{\partial \ell}{\partial h_i} h_i < 0$ happens in two cases:

1. $\frac{\partial \ell}{\partial h_i} < 0$ and $h_i > 0$; or
2. $\frac{\partial \ell}{\partial h_i} > 0$ and $h_i < 0$.

The above two cases mean h_i is increasing while being positive or is decreasing while being negative. Thus (6) is equivalent to saying that m is increasing when most entries in the output $\mathbf{h}$ are moving away from 0. Inversely, m is decreasing when most entries in $\mathbf{h}$ are shrinking towards 0. A similar analysis also holds for the masks $\mathbf{m}^F$ for the neurons in the intermediate layer of FFN and $\mathbf{m}^H$ for the hidden dimension. In addition, (6) also shows that when m gets larger, $\mathbf{h}^\top \frac{\partial \ell}{\partial \mathbf{h}}$ also needs to be further away from 0 to make m even larger.

As can be seen from (5), the gradient w.r.t. m keeps track of the output h during the learning stage. This is similar to the movement pruning (Sanh et al., 2020) which considers the changes in weights during fine-tuning. The difference is in two aspects.1) Movement assigns the weights of high magnitude with high importance, while SIMPLE considers the output $\mathbf{h}$ in each module, which is directly connected to the final output. 2) Instead of learning a specific sparse network with a given sparsity like movement, SIMPLE learns the importance of each compressible component in a separable mask learning stage, therefore numerous sub-neworks can be obtained during the fine-tuning stage, given different pruning configurations.

[1]For simplicity of notation, we drop the superscript A and omit the case when m equals 0.

3.3 Fine-tuning

After learning the importance metric during the first stage, we fine-tune the model to the required sparsity with the masks fixed. Since generative models compute tokens in left-to-right order, the compression error incurred in previous tokens will pass on to future tokens, making the learning signal noisier over time.

To alleviate this problem, we design a novel causal loss to guide the fine-tuning. Specifically, denote the input sequence of the self-attention layer in an auto-regressive Transformer as $\{\mathbf{x}_i\}_{i=1}^n$. At the i-th time step, given the query $\mathbf{q}_i = \mathbf{W}^Q\mathbf{x}_i$, the model first accesses the current key memory $\mathbf{K}_i = [\mathbf{K}_{i-1}, \mathbf{k}_i]$ and generate the output $\mathbf{y}_i$ by retrieving a convex combination of the value memory slots $\mathbf{V}_i = [\mathbf{V}_{i-1}, \mathbf{v}_i]$ as $\mathbf{y}_i = \mathbf{V}_i\text{softmax}(\mathbf{K}_i, \mathbf{q}_i)$. Therefore, historical key and value memory slots $\mathbf{K}_{i-1}, \mathbf{V}_{i-1}$ affect the generation of the current token. Thus we propose a causal loss ℓ_{causal} to align the distribution of key and value in the teacher and student models. With a slight abuse of notation, for the h-th head, denote $\mathbf{K}_h^s, \mathbf{V}_h^s$ and $\mathbf{K}_h^t, \mathbf{V}_h^t$ as the key and memory slots over all tokens of the pruned student and full teacher models, respectively. The causal distillation loss for each Transformer layer is computed over the remaining heads as:

$$\ell_{causal} = \mathbf{m}_h^A \times [\text{MSE}(\mathbf{K}_h^t, \mathbf{K}_h^s) + \text{MSE}(\mathbf{V}_h^t, \mathbf{V}_h^s)],$$

where $\text{MSE}(\cdot)$ is the mean square error.

Besides the causal distillation loss, we also adopt the conventional logits distillation loss in (4) and the hidden state distillation loss ℓ_{hidden} (Jiao et al., 2019). The overall objective during fine-tuning is:

$$\ell_{finetune} = \ell + \lambda_1\ell_{causal} + \lambda_2\ell_{hidden}. \quad (7)$$

To make the pruning process smooth, progressive pruning is applied during fine-tuning. The sparsity is linearly increased to the target value in the first half of the training steps and then fixed.

Finally, we remark that SIMPLE can be easily extended to other pruning granularities such as block pruning and unstructured pruning by simply adjusting the corresponding masks (Appendix B.1).

4 Experiments

4.1 Setup

Tasks and Models. We evaluate the effectiveness of the proposed method on three types of generative tasks, i.e., language modeling, abstractive summarization and machine translation with two types of network architectures, i.e, the decoder-only model GPT-2 and encoder-decoder model BART. More datasets and details can be found in Appendix A.1, and Appendix A.3, respectively.

Implementation Details. Note that once the masks are learned, the network can be pruned to any desired sparsity level. We empirically evaluate the method by reporting results on three different sparsity levels. For all compressible components, we set the compression ratio as 1.2x, 1.5x, and 2x respectively. For each compression ratio r, we retain $\lfloor N_H/r \rfloor$ attention heads and $\lfloor d_{ff}/r \rfloor$ neurons in the width of FFN in each Transformer layer and reduce the hidden dimension to $\lfloor d/r \rfloor$.

These compression levels result in 26%, 48%, and 67% total parameters reduction. We replace the original GeLU activation with ReLU, which speeds up the inference with comparable performance.

Compared Methods. To comprehensively evaluate SIMPLE across different tasks and network architectures, we re-implement three state-of-the-art pruning techniques: magnitude-based pruning (He et al., 2018), loss-aware pruning (Voita et al., 2019), and movement pruning (Sanh et al., 2020) [2] given the lack of baselines. For a fair comparison, we keep the same experimental setting, including the distillation loss, progressive pruning, and hidden dimension pruning, where the importance of each hidden dimension is averaged from all Transformer layers. We also list the comparisons with the existing records once their tasks and models are aligned. For instance, GUM (Anonymous, 2023) prunes FFN in GPT models on language modeling. DQ-BART (Li et al., 2022) combines layer pruning and quantization for abstract summarization.

4.2 Main Results

4.2.1 Language Modeling

We perform language modeling on WikiTxt2 (Merity et al., 2016), PTB (Mikolov and Zweig, 2012) and WikiTxt103 (Merity et al., 2016). This task predicts the probability distributions of a sequence of tokens. Perplexity (PPL) is used to evaluate performance. The results under three different sparsity levels are shown in Table 1. For all pruning methods and datasets, the performance drops as the sparsity level increases. Among the three comparison

[2] https://github.com/huggingface/block_movement_pruning

Method	Params (M)	WikiTxt2 PPL (↓)	PTB PPL (↓)	WikiTxt103 PPL (↓)
Full Model	124.4	14.49	14.67	14.23
Magnitude	91.9 $_{\downarrow 26\%}$	16.12	15.19	15.35
Loss-aware	91.9 $_{\downarrow 26\%}$	15.91	15.27	15.28
Movement	91.9 $_{\downarrow 26\%}$	16.10	15.35	15.32
SIMPLE	91.9 $_{\downarrow 26\%}$	**15.77**	**15.01**	**15.10**
Magnitude	64.1 $_{\downarrow 48\%}$	17.42	16.18	16.77
Loss-aware	64.1 $_{\downarrow 48\%}$	17.37	16.45	16.72
Movement	64.1 $_{\downarrow 48\%}$	17.35	16.36	16.75
SIMPLE	64.1 $_{\downarrow 48\%}$	**17.14**	**16.02**	**16.51**
Magnitude	41.0 $_{\downarrow 67\%}$	19.47	17.65	19.42
Loss-aware	41.0 $_{\downarrow 67\%}$	19.54	18.17	19.41
Movement	41.0 $_{\downarrow 67\%}$	20.08	17.84	19.85
SIMPLE	41.0 $_{\downarrow 67\%}$	**19.22**	**17.46**	**19.08**

Table 1: Results of language modeling on the test set of WikiTxt2, PTB and WikiTxt103 datasets with GPT-2. "↓ " denotes the percentage of reduced parameters.

	Params (M)	WikiTxt2 PPL (↓)	PTB PPL (↓)	WikiTxt103 PPL (↓)
Full Model	124.4	14.49	14.67	14.23
GUM	96.1 $_{\downarrow 22.7\%}$	-	-	17.44
Ours	96.1 $_{\downarrow 22.7\%}$	15.30	14.73	**14.73**

Table 2: Comparison with GUM which prunes the FFN of the GPT-2 on language modeling tasks.

methods, loss-aware pruning performs better than magnitude-based and movement pruning when the compression ratio is relatively small. However, when the compression becomes more aggressive, loss-aware pruning's performance becomes similar or even worse compared to the other two methods. In contrast, our proposed method, SIMPLE, consistently outperforms all three baseline pruning methods at each of the three sparsity levels.

Comparison with GUM (Anonymous, 2023). In Table 2, we compare our proposed SIMPLE method with GUM, the latest work that considers pruning the width of FFN in GPT-like models. The GUM does not prune other modules like attention heads or hidden dimension, and the reduction in model size is only 22.7% even when half of the FFN neurons are pruned away. For a fair comparison, we also prune away half of the FFN neurons. As shown, the proposed SIMPLE improves the language modeling compared to GUM by a significant margin. In addition, considering also the attention heads and hidden stage dimension as compressible components enables SIMPLE to achieve a better tradeoff between model size and performance (refer to Section 4.3.1).

4.2.2 Abstractive Summarization

In our experiments, we use the XSum (Narayan et al., 2018) and CNN/DailyMail (See et al., 2017) datasets for evaluating the summarization performance of the pruned BART model. The ROUGE 1, 2, L metrics are used as the evaluation metric. The results are presented in Table 3.

As can be seen, our proposed SIMPLE achieves the best performance on both datasets under all compared sparsity levels. The performance gain over other pruning baselines becomes more pronounced as the sparsity level increases. In particular, SIMPLE reduces 68% of the parameters with less than 2.5 Rouge 1 drop on the XSum dataset. This demonstrates the effectiveness and efficiency of our proposed pruning method for abstractive summarization tasks.

Comparison with DQ-BART (Li et al., 2022). DQ-BART (Li et al., 2022) applies weight quantization and layer pruning to compress the BART model. To compare with it, we also use quantization on the pruned model. Specifically, we first prune 48% parameters from the BART model with our proposed method, and then we quantize the pruned model with the same quantization method as DQ-BART. Table 4 shows the comparison between our proposed method and DQ-BART in terms of the performance of the quantized models. As can be seen, our method outperforms DQ-BART under both 8-bit and 2-bit quantization settings by a significant margin, under similar model sizes. This indicates that the pruned models by SIMPLE are also quantization-friendly. In particular, the ROUGE 1 score decreases by only 3 points only with over 25x model size reduction.

4.2.3 Machine Translation

The WMT16 En-Ro dataset (Bojar et al., 2016), which is used for machine translation of English to Romanian, is used to evaluate the performance of the model pruned from a 24-layer mBART model (Liu et al., 2020). The BLEU metric is used as the evaluation metric. The results are shown in Table 5. Our proposed SIMPLE method preserves good translation performance at different sparsity levels, demonstrating its effectiveness in compressing and accelerating the Transformer-based neural network for machine translation tasks.

Method	Params (M)	XSum R1 (↑)	XSum R2 (↑)	XSum RL (↑)	CNN/DailyMail R1 (↑)	CNN/DailyMail R2 (↑)	CNN/DailyMail RL (↑)
Full Model	139.4	41.81	19.07	33.92	44.73	22.15	41.83
Magnitude	102.4 $_{\downarrow 27\%}$	41.42	18.82	33.65	42.55	19.83	39.70
Loss-aware	102.4 $_{\downarrow 27\%}$	41.41	18.85	33.70	42.44	19.86	39.56
Movement	102.4 $_{\downarrow 27\%}$	41.34	18.67	33.44	43.03	20.30	40.13
SIMPLE	102.4 $_{\downarrow 27\%}$	**41.70**	**19.09**	**33.93**	**43.34**	**20.58**	**40.39**
Magnitude	70.9 $_{\downarrow 49\%}$	40.02	17.49	32.29	41.25	18.66	38.44
Loss-aware	70.9 $_{\downarrow 49\%}$	40.65	18.08	32.76	41.68	19.09	38.70
Movement	70.9 $_{\downarrow 49\%}$	40.10	17.67	32.48	42.40	19.71	39.48
SIMPLE	70.9 $_{\downarrow 49\%}$	**40.89**	**18.30**	**33.18**	**42.81**	**20.05**	**39.91**
Magnitude	44.9 $_{\downarrow 68\%}$	35.62	13.92	28.57	39.33	16.73	36.49
Loss-aware	44.9 $_{\downarrow 68\%}$	39.01	16.86	31.62	40.82	18.32	37.79
Movement	44.9 $_{\downarrow 68\%}$	38.05	16.04	30.73	41.05	18.44	38.11
SIMPLE	44.9 $_{\downarrow 68\%}$	**39.45**	**17.08**	**31.95**	**42.14**	**19.48**	**39.15**

Table 3: Results of abstractive summarization on the test set of the XSum and CNN/DailyMail dataset with BART.

Method	Bit-width (W-E-A)	Size (MB)	XSum R1 (↑)	XSum R2 (↑)	XSum RL (↑)	CNN/DailyMail R1 (↑)	CNN/DailyMail R2 (↑)	CNN/DailyMail RL (↑)
Full Model	32-32-32	532.0	41.81	19.07	33.92	44.73	22.15	41.83
DQ-BART(E3D1)	8-8-8	72.4 $_{\downarrow 86\%}$	36.39	15.29	29.91	41.18	18.75	38.58
SIMPLE	8-8-8	69.8 $_{\downarrow 87\%}$	**40.31**	**17.92**	**32.73**	**42.02**	**19.40**	**39.11**
DQ-BART(E3D1)	2-2-8	22.8 $_{\downarrow 96\%}$	29.04	9.56	23.47	39.00	16.73	36.42
SIMPLE	2-2-8	19.9 $_{\downarrow 96\%}$	**38.84**	**16.56**	**31.35**	**41.54**	**18.95**	**38.58**

Table 4: Results of the pruned BART combined with quantization on the test set of the XSum and CNN/DailyMail dataset, compared with DQ-BART (Li et al., 2022). "W-E-A" denotes the bit-width for weight, embedding layer and activation. "E3D1" denotes the pruned subnet has 3 encoder layers and 1 decoder layer. Although DQ-BART uses a stronger full model as the teacher, the proposed SIMPLE outperforms DQ-BART by a clear margin.

Method	Params (M)	WMT16 En-Ro BLEU (↑)
Full Model	610.9	26.09
Magnitude	392.1 $_{\downarrow 37\%}$	24.79
Loss-aware	392.1 $_{\downarrow 37\%}$	24.76
Movement	392.1 $_{\downarrow 37\%}$	24.88
SIMPLE	392.1 $_{\downarrow 37\%}$	**25.09**
Magnitude	299.2 $_{\downarrow 51\%}$	24.20
Loss-aware	299.2 $_{\downarrow 51\%}$	24.05
Movement	299.2 $_{\downarrow 51\%}$	24.10
SIMPLE	299.2 $_{\downarrow 51\%}$	**24.37**
Magnitude	217.3 $_{\downarrow 64\%}$	22.63
Loss-aware	217.3 $_{\downarrow 64\%}$	22.74
Movement	217.3 $_{\downarrow 64\%}$	**23.02**
SIMPLE	217.3 $_{\downarrow 64\%}$	22.90

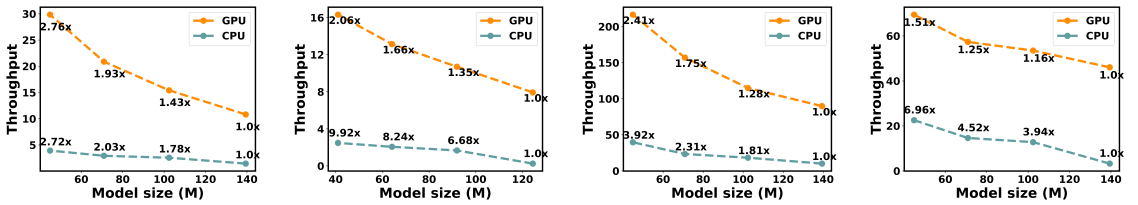
Table 5: Results of machine translation on the test set of the WMT16 En-Ro dataset with mBART.

4.3 Ablation Study

4.3.1 Pruning the Hidden Dimension

In Figure 4(a), we study the effect of whether to prune the hidden dimension (Hid) using PTB dataset on the GPT-2. The stage of mask learning is shared for all these pruned sub-networks. Under the same sparsity level, the pruning considering the hidden dimension has lower perplexities. As the compression ratio increases, the performance gain becomes even more pronounced. This indicates that *the hidden dimension of generative PLMs contains a non-negligible amount of redundancy*. By considering pruning hidden dimensions and other dimensions (attention heads, width of FFN) jointly can the model achieve a much higher compression rate under a similar performance drop.

Note that the importance of different compressible components only needs to be computed once for our proposed method and the magnitude-based and loss-aware method. To demonstrate that our sparsity-induced mask learning offers a good model initialization and importance calculation, we compare the initial perplexity before fine-tuning in Figure 4(b). Movement pruning is not compared as the metric is evaluated during fine-tuning. As can be seen, the proposed method has the lowest perplexity, since SIMPLE captures the dynamics of activations when evaluating the importance metric. In contrast to movement pruning that converges to a certain sparsity, SIMPLE provides a good sparsity-



(a) GPT-2 (Length 1). (b) GPT-2 (Length 8). (c) BART (Length 1). (d) BART (Length 8).

Figure 3: Throughput (sequence/second) comparison during generation. We vary the length of the generated text in {1,8} with beam size as 6. We use Intel Xeon Gold 6278C CPU and Nvidia V100 GPU devices, respectively.

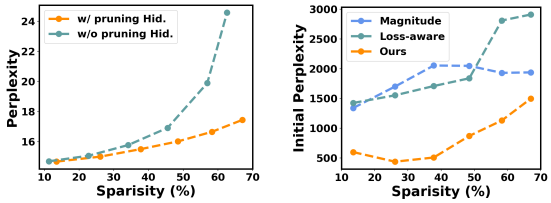


Figure 4: **(a)** The effect of pruning the hidden dimension. **(b)** The initial perplexity before fine-tuning.

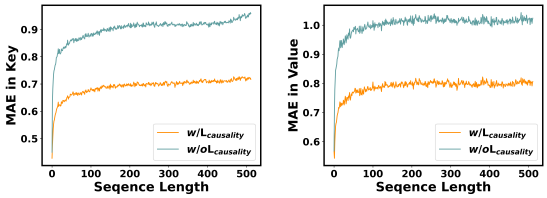


Figure 5: Visualization of the Mean Absolute Error (MAE) of the key and value in GPT.

agnostic model initialization.

4.3.2 Fine-tuning with Casual Distillation

Table 6 illustrates the effect of the ℓ_{causal} loss function on fine-tuning the pruned model for language modeling tasks. As can be seen, the proposed ℓ_{causal} consistently reduces the perplexity across different datasets. In Figure 5, we visualize the Mean Absolute Error (MAE) of the key and value in the last Transformer layer between the full model and the pruned model on the PTB dataset. The error is computed along the sequence length. The errors of the key and value both increase as the sequence length increases, which illustrates that the missed information in the pruned generative model accumulates along the sequence length, in line with the auto-regressive nature of GPT. However, by adopting the ℓ_{causal} loss function, the errors on the key and value are decreased, which improves the causality in the compressed generative model. It's worth noting that maintaining causality in compressed generative pre-trained language models is a topic that deserves further research.

	WikiTxt2	PTB	WikiTxt103
SIMPLE	19.22	17.46	19.08
w/o ℓ_{causal}	19.35	17.75	19.17

Table 6: Ablation study on the ℓ_{causal}.

4.4 Inference Speedup

Finally, we study the practical speed-up of the pruned models on both CPU and GPU in Figure 3. The batch size is set to 4 and 32 for GPT and BART, respectively. The length of the source text is fixed at 512. We vary the length of the generated text in {1,8}.

Single token generation can be used in scenarios like deterministic tasks (Conneau et al., 2018). When the length of the generation is 1, the speedup of GPT-2 is at most 2.76x/2.72x on GPU/CPU, and the speedup of BART is at most 2.06x/9.92x on GPU/CPU. When the length of the generation is larger than 1 (e.g., 8), the key and value of source text and generated history are cached, which causes the inference to move from computation-bound to memory-bound gradually. The proportion of time spent on data movement becomes greater, resulting in a lower speedup than that of single token generation. To further speed up the inference, one could combine the proposed method with other memory-saving techniques (Dao et al., 2022).

5 Conclusions

We propose SIMPLE, an efficient method to structurally prune generative pre-trained language models. SIMPLE includes attention heads, the neurons in the intermediate layer of the feed-forward network, and the hidden dimension as compressible components. By learning the masks of these compressible components through a sparsity-induced objective, different sized pruned models can be obtained, and further fine-tuned with a causal distillation objective for better performance. Empirical evaluations on different *generative tasks*, *model architectures*, and *pruning configurations* demonstrate the efficacy of the proposed method.

Acknowledgements

This research was partially supported by ACCESS – AI Chip Center for Emerging Smart Systems, sponsored by Innovation and Technology Fund (ITF), Hong Kong SAR, and partially by the Research Grants Council of the Hong Kong SAR (Project Numbers 17209721 & T45-701/22-R).

Limitations

Although SIMPLE achieves great performance with size reduction and generation speedup on various generative language tasks, it is interesting to explore combining the stage of mask learning during the pre-training. Then, one pre-trained model can be applied to downstream tasks with any required sparsity with a stage of fine-tuning. In addition, the pruning of the larger generative pre-trained language models during the fine-tuning is also worth trying. In the future, we would like to investigate the generation ability of the compressed models with more pre-trained data and larger models.

Ethics Statement

During the pre-training process, the information from the trained corpus may contain part of improper expressions like violence or discrimination. In addition, the compression of the generative pre-trained language models may result in relatively weak sentence generation.

Hence, the trained models are likely to be confronted with some potential risks in the large language models as mentioned in (Weidinger et al., 2021). With the tools proposed in (Thoppilan et al., 2022), the harmful training data can be removed to make the trained model conform to the norms of society. It is also noteworthy that the safety check is necessary before we deploy the generative language models.

References

Anonymous. 2023. What matters in the structured pruning of generative language models? *International Conference on Learning Representations (ICLR)*.

Ondvrej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, et al. 2016. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 131–198.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*.

Arnav Chavan, Zhiqiang Shen, Zhuang Liu, Zechun Liu, Kwang-Ting Cheng, and Eric P Xing. 2022. Vision transformer slimming: Multi-dimension searching in continuous optimization space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4931–4941.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel R. Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. Xnli: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Tri Dao, Daniel Y Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *arXiv preprint arXiv:2205.14135*.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. Technical Report arXiv:2208.07339.

Ali Edalati, Marzieh Tahaei, Ahmad Rashid, Vahid Partovi Nia, James J Clark, and Mehdi Rezagholizadeh. 2021. Kronecker decomposition for gpt compression. In *Advances in Neural Information Processing Systems*.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*.

M. A. Gordon, K. Duh, and N. Andrews. 2020. Compressing bert: Studying the effects of weight pruning on transfer learning. Preprint arXiv:2002.08307.

Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. 2018. Soft filter pruning for accelerating deep convolutional neural networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 2234–2240. ijcai.org.

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7).

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

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*.

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 *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174.

François Lagunas, Ella Charlaix, Victor Sanh, and Alexander M Rush. 2021. Block pruning for faster transformers. *arXiv preprint arXiv:2109.04838*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Annual Meeting of the Association for Computational Linguistics*.

Zheng Li, Zijian Wang, Ming Tan, Ramesh Nallapati, Parminder Bhatia, Andrew Arnold, Bing Xiang, and Dan Roth. 2022. DQ-BART: Efficient sequence-to-sequence model via joint distillation and quantization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 203–211, Dublin, Ireland. Association for Computational Linguistics.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. 2017. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision*, pages 2736–2744.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

J. S. McCarley. 2019. Pruning a bert-based question answering model. Preprint arXiv:1910.06360.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? *Advances in neural information processing systems*, 32.

Tomas Mikolov and Geoffrey Zweig. 2012. Context dependent recurrent neural network language model. In *IEEE Spoken Language Technology Workshop*, pages 234–239. IEEE.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

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*.

Victor Sanh, Thomas Wolf, and Alexander Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning. *Advances in Neural Information Processing Systems*, 33:20378–20389.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Maying Shen, Pavlo Molchanov, Hongxu Yin, and Jose M Alvarez. 2022. When to prune? a policy towards early structural pruning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12247–12256.

Kaitao Song, Hao Sun, Xu Tan, Tao Qin, Jianfeng Lu, Hongzhi Liu, and Tie-Yan Liu. 2020. Lightpaff: A two-stage distillation framework for pre-training and fine-tuning. Preprint arXiv:2004.12817.

Chaofan Tao, Lu Hou, Wei Zhang, Lifeng Shang, Xin Jiang, Qun Liu, Ping Luo, and Ngai Wong. 2022. Compression of generative pre-trained language models via quantization. *arXiv preprint arXiv:2203.10705*.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. 2022. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*.

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. *arXiv preprint arXiv:1905.09418*.

Hanrui Wang, Zhekai Zhang, and Song Han. 2021. Spatten: Efficient sparse attention architecture with cascade token and head pruning. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 97–110. IEEE.

Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, et al. 2021. Ethical and social risks of harm from language models. *arXiv preprint arXiv:2112.04359*.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Julien Demouth, and Song Han. 2022. Smoothquant: Accurate and efficient post-training quantization for large language models. *arXiv preprint arXiv:2211.10438*.

A Implementation Details

A.1 Dataset Splits

The train/val/test splits for different datasets are shown in Table 7. We adopt the default data splits for these datasets.

Dataset	Training	Validation	Test
WikiTxt2	36,717	3,760	4,358
PTB	42,068	3,370	3,761
WikiTxt103	1,801,350	3,760	4,358
XSum	204,045	11,332	11,334
CNN/DailyMail	287,113	13,368	11,490
WMT16 En-Ro	610,320	1,999	1,999

Table 7: Data splits of different datasets.

A.2 Dataset Description

WikiTxt2 is a compilation of corpus sourced from Wikipedia's verified Good and Featured articles.

Penn Treebank (PTB) is a widely recognized and utilized corpus for evaluating the capability of language modeling.

WikiTxt103 is another compilation of corpus origin from Wikipedia, and has more data volume than WikiTxt2.

XSum is used to evaluate abstractive single-document summarization systems. The objective is to generate a concise, one-sentence summary that accurately captures the main idea of the article.

CNN/DailyMail is another dataset for abstractive summarization. The articles that needed to be summarized are from the stories on CNN and Daily Mail website.

WMT16 En-Ro is a translation dataset with the source language as English and the target language as Romanian. It is released by the Workshops on Statistical Machine Translation (WMT) (Bojar et al., 2016).

A.3 Hyper-parameters Setting

For all tasks, the coefficients of the mask regularizer are set as 2e-4, 5e-5 and 1e-4 for $\mathbf{m}^A$, $\mathbf{m}^F$ and $\mathbf{m}^H$, respectively.

Language Modeling. We use the 12-layer GPT-2 (Radford et al., 2019) as the backbone. The sequence length is set as 512. We initialize the learning rate as 5e-4 and linearly decay it to 0 with AdamW optimizer (Loshchilov and Hutter, 2017). The overall batch size is 32. λ_1 and λ_2 are both set to 0.001. In addition, language modeling loss with coefficient 1 is added for WikiTxt103 during fine-tuning, which improves performance. For datasets WikiTxt2/PTB/WikiTxt103, we set the learning epochs as 40/100/2 and fine-tuning epochs as 120/300/8, respectively.

Summarization. We use the 12-layer BART (Lewis et al., 2020) as the backbone. For XSum and CNN/DailyMail datasets, the length of the source sequence is set as 512/512 and the length of the target sequence is set as 64/128, respectively. the beam size is utilized to generate summaries with size 4/6 for XSum and CNN/DailyMail datasets, respectively, following the default hyperparameters in BART. Both λ_1 and λ_2 are set to 0.1. The learning rate is initialized as 2e-4/2e-5 for XSum and CNN/DailyMail datasets with a linear scheduler. The AdamW optimizer is used with batch size 96. We set the learning epochs as 3 and fine-tuning epochs as 12 for both XSum and CNN/DailyMail datasets.

Machine Translation. We use the 24-layer multilingual BART (M-BART)[3] (Liu et al., 2020) as the backbone due to the lack of 12-layer open-sourced pre-trained M-BART. For the WMT16 En-Ro dataset, the maximum length of both the source sequence and target sequence is set as 128.

[3] https://huggingface.co/facebook/mbart-large-en-ro

The beam size is set as 5, λ_1, λ_2 to 0.05 and 0.001, respectively. In addition, language modeling loss with coefficient 1 is added during fine-tuning, which boosts translation performance. The learning rate is initialized as 5e-5 with a linear scheduler. We learn the model for 1 epoch and fine-tune it for 3 epochs, with the batch size 32.

A.4 Training Budget

In stage 1, the masks need to be learned once in a relatively short time. For example, it requires 1.0/2.3 hours only for GPT-2 on the PTB/WikiTxt103 dataset with 8 cards. And this cost can be amortized by N times for N different pruned neural networks, since all the pruned networks shared the masks learned in stage 1. Then, the model is fine-tuned with the masks fixed in stage 2.

A.5 Compression Ratio vs. Total Parameter Reduction

In practice, we assign the same compression ratio to all the compressible components, the attention heads, the width of FFN and the hidden dimensions for ease. The aforementioned compression ratio is not equal to the ratio of total parameter reduction.

For instance, in the case with a compression ratio 2x, assuming a matrix M with the shape m*(nd), reducing both the hidden dimension (m) and the number of attention heads (n) to m/2, n/2 respectively will lead to a 75% reduction. On the other hand, for the embedding layer and layer normalization, the setting of compression ratio 2x leads to 50% parameter reduction. Overall, the setting of compression ratio 2x results in approximately 67% total parameter reduction.

B More Experiments

B.1 Extension to Block Pruning and Unstructured Pruning

Extension to Block Pruning. Block pruning (Lagunas et al., 2021) is an extension of the common structured pruning, which balances the fine-grained model sparsification and dense hardware optimization. In block pruning, the weights are divided into different blocks that can be computed efficiently via appropriate GPU kernels. Pruning algorithms are expected to keep the important blocks and remove the redundant ones.

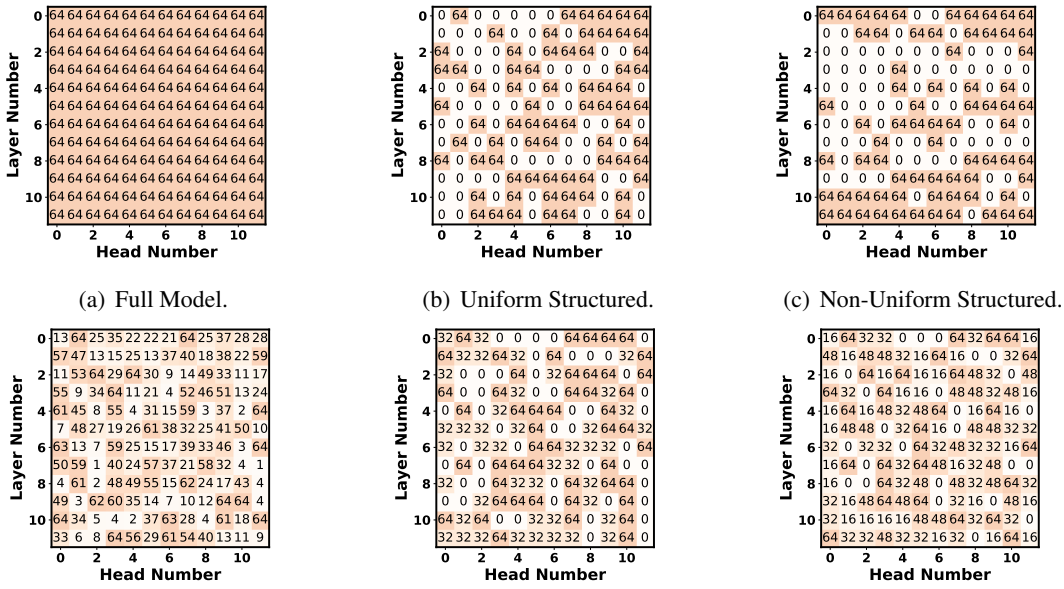
The proposed SIMPLE can be easily combined with block pruning by adapting the size of learnable masks according to the block size. Assume the weight matrix requires to be divided into multiple blocks with block size (M, N), the shapes of $\mathbf{m}^H$, $\mathbf{m}^A$ and $\mathbf{m}^F$ are $R^{d/M}$, $R^{N_H \times d/(N_H N)}$ and $R^{d_{ff}/N}$, respectively. The learnable masks score these blocks and select the important ones.

	WikiTxt2	PTB	WikiTxt103
Uniform Structured	19.22	17.46	19.08
Non-uniform Structured	20.79	18.75	19.45
Unstructured	19.18	17.10	23.03
Block size (32,32)	19.43	17.43	19.00
Block size (16,16)	19.51	17.41	19.02

Table 8: Ablation of sub-net configurations.

Extension to Unstructured Pruning. In SIMPLE, the granularity of pruning can be easily controlled by the shape of the mask. For example, the shape of the attention mask $\mathbf{m}^A$ can be set as $R^{N_H \times d}$ to perform unstructured pruning on the attention module. Here, unstructured pruning does not refer to removing part of connections in any neuron, but removing different dimensions in different attention heads, similar to ViT-slim (Chavan et al., 2022). However, this does not directly accelerate the model inference. By default, we rank the learned masks in each Transformer layer locally to ensure that the number of attention heads in the Multi-Head Attention (MHA) and the width of the Feed-Forward Network (FFN) are equal in each Transformer layer. This is referred to as uniform structured pruning. If we rank the masks globally in each compressible component, then the number of attention heads in the MHA and the width of the FFN vary in different Transformer layers, which is referred to as non-uniform structured pruning.

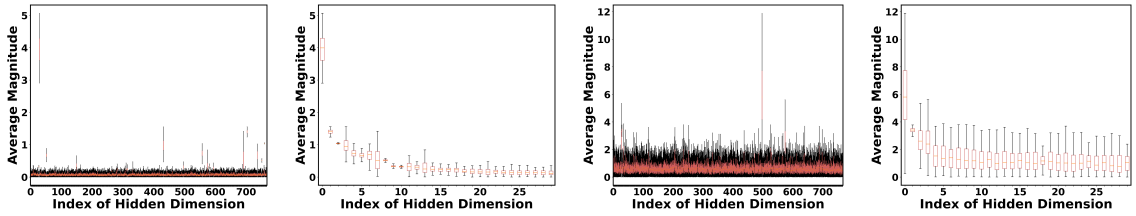
In Table 8, we report the performance of different variants of SIMPLE by changing the shape of the mask or the range to rank the mask. The attention module of the learned subnets is visualized in Figure 6. Interestingly, the setting of uniform structured pruning not only performs competitively against non-uniform pruning and unstructured pruning but also requires no specific hardware design for speedup. Additionally, when using non-uniform structured pruning, attention heads in the shallow and deep layers are more likely to be retained, while those in the intermediate layers are considered relatively less important and are discarded. Furthermore, SIMPLE can be extended to block pruning by setting the shape of the trained masks to adapt to the assigned block size. However,



(a) Full Model. (b) Uniform Structured. (c) Non-Uniform Structured.

(d) Unstructured. (e) Block Pruning (32,32). (f) Block Pruning (16,16).

Figure 6: Illustration of adapting different pruning configurations using SIMPLE from the perspective of MHA: (a)-(f) Dimension per attention head in different pruning configurations based on the SIMPLE on the 12-layer GPT-2, which has 12 heads in each attention module and the dimension of a head is 64 before compression.



(a) Layer 1 (all). (b) Layer 1 (top-30). (c) Layer 6 (all). (d) Layer 6 (top-30).

Figure 7: Boxplots of the magnitudes of the hidden dimensions in Layer 1 and 6 of a BART fine-tuned on the dataset XSum. For each layer, we show both (i) magnitudes of all 768 dimensions; and (ii) the top-30 magnitudes. The other layers follow similar patterns.

it should be noted that block pruning can improve performance but may not bring real speedup when attention heads are not entirely removed (Lagunas et al., 2021).

B.2 Comparison with Un-pruned Small Models

Method	Params (M)	WikiTxt103 PPL (↓)
24-layer GPT	354.8	12.60
pruned 24-layer GPT	125.3 $\downarrow 65\%$	**13.87**
un-pruned 12-layer GPT	124.4	14.23

Table 9: Comparison between 1) compression from a large fine-tuned model by SIMPLE; 2) pre-training a small model and fine-tuning it on the target dataset.

It is desirable for a small model pruned from a large model to outperform a similar-sized un-pruned model trained from scratch. To evaluate this, we compare the performance of a model pruned from a 24-layer GPT to a similar-sized un-pruned 12-layer model in Table 9. As is shown, the pruned model has a lower perplexity than the un-pruned pre-trained small model, demonstrating the effectiveness of SIMPLE. Additionally, by pre-training one large model and then compressing it to various sizes during task-specific fine-tuning, we save the effort required for pretraining various-sized small models.

B.3 Visualization of the Persistent Outliers on BART

From Figure 7, we visualize the distribution of the magnitude of the hidden dimensions on the

fine-tuned BART, which is averaged on the XSum dataset. As is shown in the visualization, a few hidden dimensions have large magnitudes while most of the hidden dimensions maintain a small vibration, despite the input data samples. Similar to the observation on GPT, it indicates that the hidden dimension of BART has redundancy, which motivates us to prune the hidden dimension with a sharp compression rate.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Section of Limitation

☑ A2. Did you discuss any potential risks of your work?
Section of Ethics Statement

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Section of Abstract and Introduction

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☐ Did you use or create scientific artifacts?

Not applicable. Left blank.

☐ B1. Did you cite the creators of artifacts you used?
Not applicable. Left blank.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Not applicable. Left blank.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Not applicable. Left blank.

☑ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Section A.1 and A.2 in the Appendix

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Not applicable. Left blank.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Sectoin A.1 and A.2 in the Appendix

C ☒ Did you run computational experiments?

Left blank.

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 4

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section A.3 in the Appendix

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 3.1

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 4

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Not applicable. Left blank.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Not applicable. Left blank.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Not applicable. Left blank.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. Left blank.