Resource-Efficient Transformer Pruning for Finetuning of Large Models

Supplementary Material

6. Anatomy of Transformer-based Models

RECAP is designed as a task-agnostic but also a task-aware framework for finetuning with efficient pruning of transformer-based large models. It is also important to examine the anatomy of these models to better understand how RECAP increases the memory efficiency of the process. Therefore, in this section, we provide more background on transformer-based models in terms of the factors in computation and memory cost. In terms of computation cost, we can categorize the operations in a transformer into three main groups:

- *Tensor contractions*: linear layers and attention heads with matrix multiplications (most expensive)
- *Normalizations*: softmax and layer normalization with reduction operations (less expensive)
- *Element-wise operations*: biases, dropout, activations, and residual connections (least expensive)

Considering the high computation cost of attention and feedforward layer operations, techniques targeting to speed up these executions for inference have been proposed such as flash attention and adaptive token selection. However, as the focus of our study is resolving the issue of GPU memory bottleneck, we further analyze the components of memory footprint during finetuning. There are four main components causing memory footprint during finetuning:

- *Weights*: Model weights have to be stored at GPU during forward and backward passes. The cost is 4 bytes/param at fp32, 2 bytes/param at fp16/bf16, and so on.
- *Optimizer states*: Optimizer states have to be stored at GPU during parameter and optimizer state updating steps after backward passes. At fp32, the cost is 4 bytes/param for optimizers that store one state per parameter such as SGD with momentum, and 8 bytes/param for optimizers that store two states per parameter such as Adam.
- *Gradients*: Gradients of the parameters being updated have to be stored at GPU during parameter updating steps after backward passes. The cost is 4 bytes/param at fp32.
- *Activations*: Certain intermediate activation outputs during the forward pass can be temporarily stored at GPU to prevent re-calculations during the backward pass and high latency. The cost of activations mainly depends on the model architecture, batch size, and input sequence length.

There can be additional costs due to pre-allocations, loaded kernels etc., which may depend on the implementation and the neural network library. We provide further details about implementation in Section 8.2.

7. Details of Weight Grouping for Structured Pruning and Masking

In this section, we provide further details on RECAP in terms of how we group weights for pruning and masking. As explained in Figure 2 in Section 3.1.1, we consider weights coupled in a head as groups for attention modules. Formally, let $\mathbf{W}_q^{(l)}$, $\mathbf{W}_k^{(l)}$, $\mathbf{W}_v^{(l)}$ denote the query, key, value and $\mathbf{W}_o^{(l)}$ the output weight matrices of the l-th transformer module, where $\mathbf{W}_q^{(l)}$, $\mathbf{W}_k^{(l)}$, $\mathbf{W}_v^{(l)} \in \mathbb{R}^{d_a^{(l)} h^{(l)} \times d_i^{(l)}}$ and $\mathbf{W}_o^{(l)} \in \mathbb{R}^{d_o^{(l)} \times d_a^{(l)} h^{(l)}}$. Here, $d_a^{(l)}$ is the number of hidden dimensions at each head, $h^{(l)}$ is the number of heads, and $d_i^{(l)}$, $d_o^{(l)}$ are the input/output dimensions, for the l-th transformer module. Likewise, we also group the feedforward layer weights $\mathbf{W}_{f_1}^{(l)} \in \mathbb{R}^{d_f^{(l)} \times d_o^{(l)}}$, $\mathbf{W}_{f_2}^{(l)} \in \mathbb{R}^{d_o^{(l)} \times d_f^{(l)}}$, but in the neuron-level resolution. Thus, we have the following set of pruning weight groups:

$$
\begin{aligned}
\mathcal{G}^{(p)} = \{ \quad & (\mathbf{W}^{(l)}_{q_{[d_a^{(l)} m_a : d_a^{(l)}(m_a+1),\,:]}}, \mathbf{W}^{(l)}_{k_{[d_a^{(l)} m_a : d_a^{(l)}(m_a+1),\,:]}}, \\
& \mathbf{W}^{(l)}_{v_{[d_a^{(l)} m_a : d_a^{(l)}(m_a+1),\,:]}}, \mathbf{W}^{(l)}_{o_{[:\,, d_a^{(l)} m_a : d_a^{(l)}(m_a+1)]}}) \mid \\
& m_a \in \{1, ..., h^{(l)} - 1\}\}_{l=1}^{L} \cup \\
\{ \quad & (\mathbf{W}^{(l)}_{f_1[m_f : m_f+1,\,:]}, \mathbf{W}^{(l)}_{f_2[:\,, m_f : m_f+1]}) \mid \\
& m_f \in \{1, ... d_f^{(l)} - 1\}\}_{l=1}^{L},
\end{aligned}
$$

and $M = |\mathcal{G}^{(p)}|$ is the number of weight groups for pruning and L is the number of transformer modules. While sorting these weight groups based on importance values and keeping the most important groups, we enforce each module to keep at least one head to prevent layer removal.

For the finetuning masks, we consider grouping at the neuron level and do not enforce coupled pruning in attention modules hence, we have the following set of finetuning weight groups:

$$
\begin{aligned}
\mathcal{G}^{(f)} = \{ \quad & (\mathbf{W}^{(l)}_{q_{[m_a : m_a+1,\,:]}}), (\mathbf{W}^{(l)}_{k_{[m_a : m_a+1,\,:]}}), \\
& (\mathbf{W}^{(l)}_{v_{[m_a : m_a+1,\,:]}}), (\mathbf{W}^{(l)}_{o_{[:\,, m_a : m_a+1]}}) \mid \\
& m_a \in \{1, ..., h^{(l)} - 1\}\}_{l=1}^{L} \cup \\
\{ \quad & (\mathbf{W}^{(l)}_{f_1[m_f : m_f+1,\,:]}), (\mathbf{W}^{(l)}_{f_2[:\,, m_f : m_f+1]}) \mid \\
& m_f \in \{1, ... d_f^{(l)} - 1\}\}_{l=1}^{L},
\end{aligned}
$$

and $N = |\mathcal{G}^{(f)}|$ is the number of weight groups for finetune masking. These weight groupings are then utilized to perform the importance calculations and actual prun-

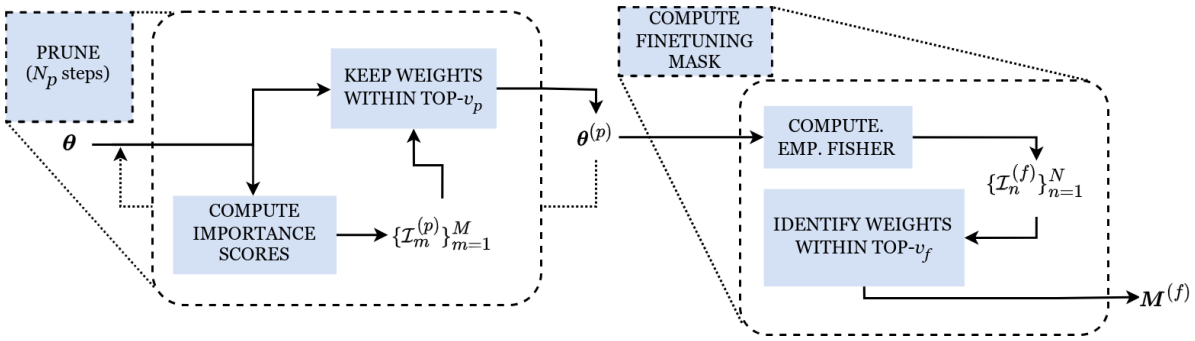

Figure 10. Schematic of the operations in pruning and finetuning mask computing stages.

ing/masking operations. We also provide a flowchart in Figure 10 to illustrate the processes executed at the CPU for pruning and finetuning mask calculations over these weight groups.

8. Experiment Setup

In this section, we provide the details of datasets, preprocessing, and implementation.

8.1. Datasets and Preprocessing

In image classification experiments, we consider common benchmark datasets: CIFAR-100 [22] (60K images, 32x32 resolution, 100 classes) and TinyImageNET (100K images, 64x64 resolution, 200 classes), which is the downsampled and smaller version of ImageNet 2012 [8]. We consider ImageNet-21k (IN21K) as the pre-training dataset in image classification experiments. We follow the data augmentation techniques applied in [17] with center crop and random horizontal flip. We resize images to 224x224 considering the pre-trained model configuration. For semantic segmentation, we use the Cityscapes [7] and KITTI [1] datasets, which contain images from urban areas mostly with cars, pedestrians, roads, etc. Cityscapes has 5000 images with the resolution of 1024x2048 and KITTI has 400 images with the resolution of 375x1242, finely labeled at pixel-level for 19 classes. We hold out 10% of the train-set images for validation. In natural language understanding experiments, we consider the GLUE benchmark, which encapsulates nine datasets in varying dataset sizes for different tasks (sentiment analysis, linguistic acceptability, question answering etc.) [37]. For tokenization, we use the pre-trained tokenizers provided by the open-source HuggingFace library [38].

8.2. Implementation Details

On CIFAR-100 and TinyImageNet datasets, we perform experiments with two variants of vision transformer: ViT-base with 86M parameters and ViT-large with 307M parameters [11]. These models are pretrained with 16x16 patches of the images in Image-Net21K, which is a very large-scale image dataset with 14 million images from 21843 classes at 224x224 resolution. On Cityscapes and KITTI datasets, we use the Mask2Former architecture, which has shown great performance across panoptic/instance/semantic segmentation tasks. This model employs the optimization at the mask level instead of pixel level, and utilizes a pixel decoder and a multi-scale transformer decoder with masked attention. We consider the two versions, pretrained at ImageNet-21K with Swin-base and Swin-large encoder backbones. We train with half-resolution images in Cityscapes and with full resolution in KITTI. We perform full-resolution single-scale inference.

We use the first-order Taylor expansion-based pruning importance criterion and empirical Fisher Information-based finetuning importance criterion for RECAP in all comparison experiments. In all experiments, we operate on fp32-bit precision except optimizer states, which we store with bf16. We use Adam optimizer and perform separate logarithmic grid searches for the learning rate at each setup in the range of {1e-5, 1e-3} and choose the value based on validation performance. We use linear decay for the learning rate during each iteration. We perform two epochs of training at each finetuning stage and repeat five iterations ($K = 5$), which results in ten epochs of finetuning in total. We set the size of the pseudo-dataset used for importance calculations to 100. We utilize gradient accumulation during finetuning. In LoRA, we apply the technique on attention weights. We set the number of dimensions to 8 and the scaling factor to 32. In Pre-FT and Post-FT pruning, we use the same pruning importance criterion and perform ten epochs of finetuning. We report the results for the checkpoint with the best validation performance. We repeat image classification experiments three times and report the mean of the obtained results. For memory measurements, we report the total allocation required for weights, gradients, optimizer states, and activations. Experiments are conducted on a machine with RTX3060 and a 2.9GHz 8-core CPU. We use Pytorch 2.0

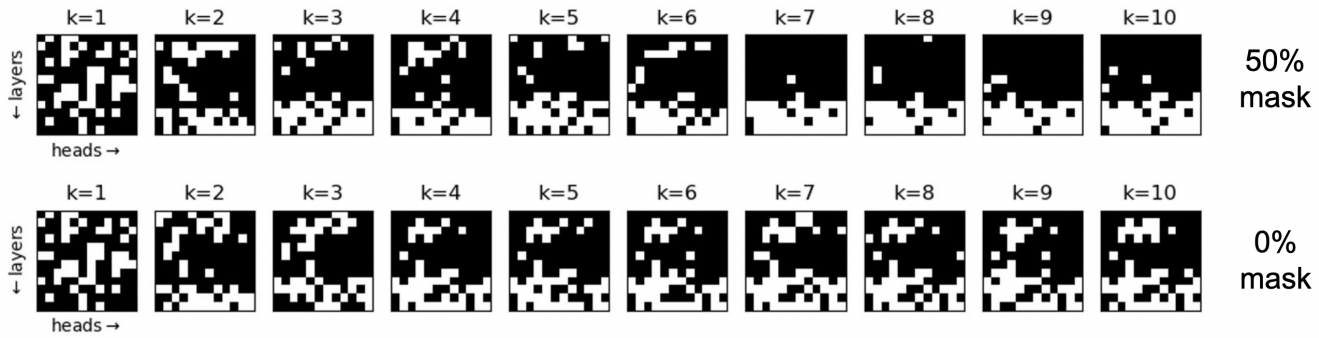

Figure 11. Evolution of pruning masks using RECAP with 50% masking and RECAP without masking for BERT-base @ COLA. We illustrate the pruned heads in attention models with white color (each row as a layer) computed at each iteration.

as the deep learning library, and utilize some functionalities in the NNI compression library. [1] We have also measured the transfer time of tensors between CPU and GPU. In our setup, we measure around 1.24 GB/s for CPU→GPU and 0.56 GB/s for GPU→CPU (e.g. total transfer time per round for weights/opt.states/masks/updates is around 0.33s for ViT-b and 1.22s for ViT-l with RECAP). Lastly, the reduction in the number of parameters and FLOPs of the pruned model is almost linear with the pruning ratio. For instance, the pruned ViT-base/large/huge with 33% pruning has 59M/204M/417M parameters and 12G/42G/118G FLOPs whereas the full variants have 86M/307M/632M parameters and 18G/62G/167G FLOPs respectively.

9. Visualization of the Pruned Model Structure

In this section, we analyze the pruned model structure with RECAP through the iterations and also compare the behavior of RECAP with and without masking. To this end, in Figure 11, we illustrate the pruning masks calculated for BERT-base being finetuned at CoLA and pruned with $r_p = 33\%$. Here, we visualize the pruning masks at each iteration such that each image has layers as rows, heads as columns, and white items are pruned out. We observe that the pruned model structure saturates early at around the third iteration without masking. With masking, we observe a smoother evolution of the pruned model structure.

10. Results with Complementary Techniques

In this section, we analyze the efficiency improvements of RECAP when combined with other efficiency techniques such as quantization and adapters. Since the improvements gained with pruning are orthogonal to these techniques, we observe further improvement in memory footprint reduction with slight performance loss in a complementary manner. We report the results for ViT-base at CIFAR-100 in Table 6. Here, we report the results obtained with RECAP at 16-bit precision, also with LoRA adapters and 8-bit quantized optimizer. We observe that the performance decrease is around 3% except for the adapters, where the higher drops are due to the fact that we disable the masking during finetuning to use adapters instead. More sophisticated methodologies to combine pruning with adapters is a promising future direction and can potentially yield higher efficiency gains. RECAP with quantization only causes a slight performance drop of less than 1%. The fact that the advantages of RECAP can be effectively combined with optimizations from other perspectives such as quantization increases the practical applicability of our approach.

[1] https://github.com/microsoft/nni

Model	Method	**Pruning Ratio** (r_p)		
		16.6%	33.3%	50.0%
ViT-base (86M)	**RECAP**	90.50	88.34	83.33
Full-FT: 91.84%	**RECAP (w/ bf16)**	89.52	88.25	83.10
Head-FT: 82.75%	**RECAP (w/ adapters)**	86.33	84.40	80.75
	RECAP (w/ 8-bit Adam)	89.57	87.68	82.90

Table 6. The results for ViT-base at CIFAR100 with RECAP combined with bf16 quantization, adapters instead of masking during fine-tuning and 8-bit quantized Adam optimizer.

Importance Criterion for		**CIFAR-100**				**TinyImageNet**			
Pruning	Masking	16.6%	33.3%	50.0%	66.6%	16.6%	33.3%	50.0%	
Magnitude	Random	89.65	85.11	70.90	47.10	85.50	79.44	65.77	
	Emp. FI	90.22	85.77	75.87	52.12	86.45	80.76	69.76	
Connection Sensitivity	Random	90.01	87.76	80.41	69.90	84.43	82.50	70.44	
	Emp. FI	90.44	88.10	82.76	72.45	86.88	83.65	73.14	
Taylor-1st	Random	89.92	88.00	80.21	68.99	84.02	82.05	70.19	
	Emp. FI	90.50	88.34	83.33	72.90	86.93	83.83	73.96	
Taylor-2nd	Random	89.90	88.05	80.62	69.11	84.60	82.76	70.02	
	Emp. FI	90.44	88.39	83.55	73.21	86.69	83.76	74.00	

Table 7. RECAP performance on image recognition datasets with ViT-base for different pruning and masking criteria combinations.

11. Effect of the Pruning and Masking Criteria

In this section, we analyze the effect of various criteria in the pruning stage and finetuning mask calculations. As we explain in Sections 3.1 and 3.2, in the main experiments, we use first-order Taylor approximation for pruning importance and empirical Fisher for finetuning mask importance calculations. To analyze the impact of the criteria, we evaluate RECAP with three other pruning criteria. First, we consider magnitude-based pruning [15], where the importance of each weight group is the sum of weight magnitudes. Second, we consider the connection sensitivity-based pruning as suggested in [23], which depends on the scaled gradient magnitudes. Lastly, we also perform experiments with second-order Taylor expansion-based criterion. For masking, we also experiment with random masking without any importance calculations. We report the results on CIFAR-100 and TinyImageNet in Table 7. We observe that Taylor-expansion-based approximation of importance values with empirical Fisher consistently yields the best performance. We observe a slight performance decrease with the connection sensitivity proposed in [23]. Magnitude-based pruning suffers a significant decrease in performance in high pruning ratios. Considering the high cost of second-order Taylor due to Hessian computations, we use the first-order Taylor in RECAP due to its consistent performance and lower computation cost.

12. Analysis of the Iterative Pruning Stage

In this section, we analyze the impact of the pseudo-dataset (sampled from the finetuning dataset) size used for importance calculations in the pruning stage. To this end, we report the performance with ViT-base at CIFAR-100 for various sizes of pseudo-datasets. We also compare the performances with one-shot and iterative approaches during pruning. We report the results in Figure 12. Here, we plot the accuracy obtained with ViT-base on CIFAR-100 at various pseudo-dataset sizes. Each line corresponds to a different pruning ratio. We observe that at every pruning ratio regime, using an excessively small subset of the finetuning dataset hurts the performance as it lowers the quality of importance estimations for pruning and mask computations. In particular, empirical Fisher has limitations in particular when the sample size is small, which can be a factor in low performance when $|\mathcal{D}_s| = 10$. Therefore, we set the size of $\mathcal{D}_s$ to 100, which is the value, where after, we observe no significant improvement in the system performance and the CPU operations (pruning and mask computation) would be unnecessarily more time-consuming.

In addition, in Figure 12, we plot the results with one-shot pruning in the pruning stage ($N_p = 1$) using dashed lines. We normally set $N_p = 1, 2, 3, 4$ for $r_p = 16.6\%, 33.3\%, 50.0\%, 66.6\%$, respectively in all experiments since as shown, we observe $\sim 0.5\%$ accuracy decrease when $N_p = 1$.

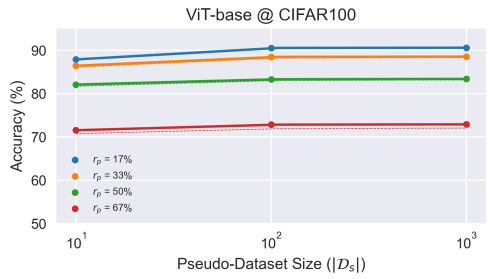

Figure 12. Accuracy for ViT-base on CIFAR-100 with RECAP at different pruning ratios and pseudo-dataset size used for pruning and mask computation operations. Dashed lines represent the results for RECAP with $N_p = 1$.