
Supplemental Material: Computation and Memory-Efficient Model Compression with Gradient Reweighting

This appendix can be divided into 8 parts. To be precise,

1. Section A provides the experimental configuration.
2. Section B describes our algorithm of distributed sparse training.
3. Section C describes the analysis of variance reduction.
4. Section D elaborates on the computation of the projection operator.
5. Section E presents the detailed proof of Theorem 1.
6. Section F talks about the limitation of this paper.
7. Section G discusses the broader impact of the LLM pruning.
8. Section H presents the generations of pruned Llama2-7B.

A Experiments Configuration.

A.1 LLM Configuration

Baselines. LLM-Pruner adopts structured pruning that selectively removes non-critical coupled structures based on gradient information. SliceGPT replaces each weight matrix with a smaller (dense) matrix, reducing the embedding dimension of the network. Wanda-sp prunes weights with the smallest magnitudes multiplied by the corresponding input activations within a module. FLAP employ adaptive global model compression strategies.

Platform. PyTorch [39] framework is used for all experiments. We used only one 40 GB A100 GPU for LLM experiments, and the actual pruning process utilized approximately 6 GB of memory.

Experimental Hyperparameters. Here is a detailed presentation of the experimental setup for pruning in Llama2-7B. Please see the Table 8 for detailed configuration.

Table 8: Hyperparameter of Llama2-7B. Other models hyperparameter configurations are the same.

Epochs	Train Set	Batch Size	Sequence Length	Optimizer	Learning Rate	ϵ	M	τ	Test Sequence Length
1	120K(C4)	8	128	decoupled_Adamw	5e-3	0.2	8	10	2048

Evaluations. In our evaluation, the structural parameters s of each module are sorted, and the bottom ρK are discarded to meet the pruning rate ρ . Perplexity is calculated as:

$$\text{PPL}(X) = \exp \left\{ -\frac{1}{t} \sum_i^t \log p_W(x_i | x_{<i}) \right\},$$

where $X = (x_0, x_1, \dots, x_t)$ is the tokenized sequence, $p_W(x_i | x_{<i})$ is the log-likelihood of the i -th token conditioned on the preceding tokens $x_{<i}$ according to our model.

Following [34], we evaluate the zero-shot tasks by using the LM Evaluation Harness [16], specifically selecting the following five tasks: PIQA, HellaSwag, ARC-e, ARC-c, and OBQA.

A.2 Implementation for Algorithm 1

There is a slight difference between the actual implementation and Algorithm 1 for fair comparison. For any $(t, i, r) \in T \times M \times \tau$, the mini-batches are distinct, and each mini-batch is utilized only once, ensuring $T \times M \times \tau = \# \text{mini-batch}$, i.e., epoch=1 and the numbers of forwards are equal for each method. Specifically, in Step 3, we instantiate M sub-models, each assigned with τ unique mini-batches. These sub-models are used in Steps 4-9 to perform τ updates to parameter s . We select $M = 8$ through the experiments below, as too large M improves stability but reduces the total number of updates (i.e., $T \times \tau$) due to the efficient implementation, causing performance degradation. The effect of M can be seen in Table 9. Under these different numbers of sub-models, our method consistently outperforms the baseline.

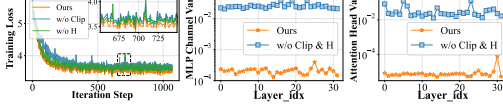


Figure 3: Ablation study on variance reduction.

Number of sub-models (M)	2	4	8	16
WikiText2 / Ptb Perplexity	36.88 / 77.82	31.27 / 60.74	26.83 / 44.23	28.01 / 48.80

Table 9: M sub-models on 50% Llama2-7B.

A.3 Sparse Training for Visual Models

Our algorithm can also be applied to sparse training scenarios. We conducted training tasks with extremely high sparsity on various vision models, including ResNet [22] and VGG [44], using the CIFAR10 and CIFAR100 datasets. During the structured pruning of the model, we simultaneously updated the model parameters, as shown in Eqn.(7). Due to the instantiation of sub-models in our algorithm, the computational resources required for the backpropagation process of parameter updates are greatly reduced. Table 11 shows that, even under extremely high sparsity, our algorithm allows the pruned models to maintain high accuracy. Table 12 demonstrates that our sparse training algorithm results in significant resource savings during the training process. Table 13 shows that our algorithm performs excellently in sparse training of vision models while significantly saving training resources.

Platform. PyTorch framework is used for all experiments. Our all sparse training experiments on visual models were conducted on a single RTX 4090 GPU.

Experimental Hyperparameters. Detailed configuration for sparse training is shown in Table 10. We employ the SGD optimizer for weight updates.

Table 10: Hyperparameters of ResNet-32. Other models hyperparameter configurations are the same.

Epochs	Train Set	Batch Size	Weight Optimizer	Weight lr	Structure Optimizer	Structure lr	Instan Num(M)	Instan Freq(τ)
300	CIFAR100/10	256	SGD	0.1	Adam	12e-3	2	20

Train-cost savings. Our calculation of Train-cost savings follows the approach used in [67], including both the forward and backward propagations. The forward pass computes the model’s loss function, while the backward pass calculates the gradients with respect to the weights and activations of each layer. In terms of FLOPs, the backward generally requires 2–3 times the computational cost of the forward; for simplicity, we approximate it as 2 times in our calculations. The forward propagation of dense network is f_D . Since we instantiated sub-models, both the forward and backward propagations are fully sparse. Thus, the forward FLOPs is f_S , and the backward FLOPs is $2 \times f_S$. The forward propagation has to be computed two times as we instantiate 2 sub-models. Therefore the train-cost saving is computed as $\frac{f_D + 2 \times f_D}{2 \times f_S + 2 \times f_S} = \frac{3}{4 f_S / f_D}$. Furthermore, $f_S / f_D \approx (1 - \rho)^2$, allowing us to achieve approximately second-order train-cost savings.

A.4 Distributed Sparse training Configuration

Baselines. Dense non-distributed performs non-distributed training on a single GPU. Dense distributed performs distributed training on four GPUs. Local SGD performs local updates on its data

Table 11: Accuracy of different models on the CIFAR10 under extremely high sparsity.

Model Structure	Dense	70%	80%	90%	95%	98%
ResNet20	94.77	93.16	93.01	91.13	89.12	83.89
ResNet32	94.83	94.32	93.60	92.03	90.45	86.82
VGG16	94.34	93.54	93.08	91.65	90.33	85.04

Table 12: Performance and train cost savings of ResNet32 on the CIFAR100 under sparse training.

Prune rate	Dense	70%	80%	90%	95%	98%
Accuracy(%) \uparrow	75.59	71.20	69.05	67.99	64.89	57.55
FLOPs(%) \downarrow	100	30.67	23.30	12.46	5.9	1.3
Train-cost saving(\times) \uparrow	1	2.41	3.04	5.96	12.13	56.25

Table 13: Comparison of vision model with other baselines.

Dataset	Model	Method	Val Acc(%)	Params(%)	FLOPs(%)	Train-Cost Savings(\times)
CIFAR-10	ResNet-20	Original	94.77	100	100	1 \times
		L1-Pruning [30]	90.9	55.6	55.4	-
		Provable [33]	90.8	37.3	54.5	-
		GrowEfficient [61]	90.91	35.8	50.2	1.13 \times
		Ours	93.16	30.0	34.1	2.37\times
ImageNet-1K	ResNet-50	Original	77.0	100	100	1 \times
		L1-Pruning	74.7	85.2	77.5	-
		Provable	75.2	65.9	70.0	-
		GrowEfficient	75.2	61.2	50.3	1.10 \times
		Ours	75.7	50.0	47.8	1.40\times

and then merges the updated weights of different nodes together. PowerSGD performs distributed training with sparsified data communication. TSNNs performs distributed weight-level sparse training developed for multi-layer-perceptron and we extend it to our experiments by performing sparse training especially on linear layers. EffTrain is the base solver of our method and performs non-distributed sparse training.

Platform. We perform experiments on PyTorch with *gloo* as the distributed backend. We use V100 in ImageNet experiments. Distributed training on ImageNet is performed on a cluster of 8 nodes (each equipped with 1 GPU). Worker nodes are connected via a 40 Gbps (5000 Mb/s) Ethernet interface.

Time Calculations. Our computational time is the sum of forward and backward computational costs. For distributed algorithms, communication time is the time for passing weights and gradients. For sparse distributed algorithms, another important time quantity is also counted: model instantiation, since the subnetwork is evolving through the training process.

Experimental Hyperparameters. We present the details of hyperparameters in Table 14.

Table 14: Hyperparameters of ResNet-50, MobileNet-V1 and DeiT-Base on ImageNet-1K. The left term of "/" denotes the hyperparameter for non-distributed training while the right term denotes that of distributed training. R-50, M-V1 and D-Base stands for ResNet-50, MobileNet-V1 and DeiT-Base.

Model Structure	Nodes	Epochs	Batch Size	Weight Optimizer	Weight lr	Structure Optimizer	Structure lr	Label Smoothing	Instan Freq(τ)
R-50, M-V1	1/8	100	128/1024	SGD	0.256	Adam	12e-3	0.1	20
D-Base	1/8	300	128/1024	AdamW	5e-4	Adam	12e-3	0.1	20

B Algorithm of Sparse Distributed Training

Our distributed sparse training algorithm is illustrated in Figure 4, with detailed implementation provided in Algorithm 2. The real-time and infrequent update/communications are colored in **red** and **blue**, respectively. The computation in all clients are sparse for both forward and backward operations as they only work on a small subnetwork. Eqn.(6), the formulation of \mathbf{g}_s^{vr} shows that by passing only two real numbers, i.e., $\omega(\mathbf{m}^{(i)})$ and $\mathcal{L}_B(\mathbf{W}_{\mathbf{m}^{(i)}})$ for each client, we can update \mathbf{s} in the server based on the latest results in the clients and inform the clients with the essential structure information in real time. The reweighting technique enables us to reduce the synchronization frequency of \mathbf{W} to every τ iterations. And even in these synchronization steps, we only need to pass the sparse parameters $\mathbf{W}_{\mathbf{m}^{(i)}}^{(i)}$ instead of the full model. Thus both the overall communication and instantiation costs can be effectively saved.

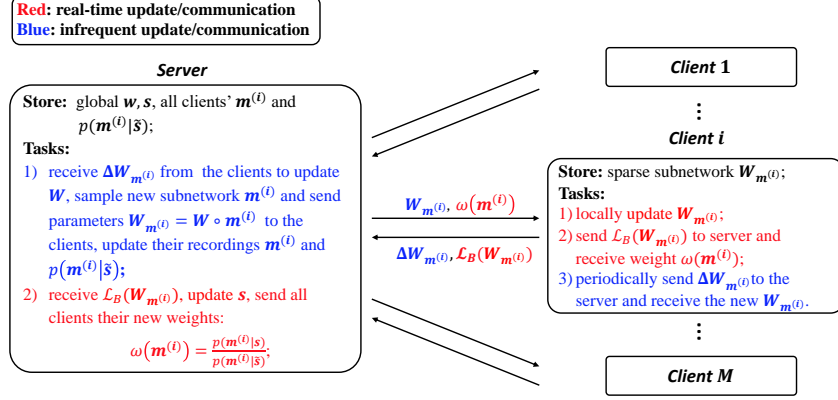


Figure 4: Extend to distributed sparse training. Only two scalars, $\omega(\mathbf{m}^{(i)})$ and $\mathcal{L}_B(\mathbf{W}_{\mathbf{m}^{(i)}})$, need to be transmitted in real-time. This significantly reduces the communication cost.

Algorithm 2 Efficient Sparse Collaborative Training: Distributed Training Version

Input: prune rate ρ , structural parameters \mathbf{s} , inner steps τ , dense DNN \mathbf{W} , learning rates η_1 and η_2 .

- 1: **for** $t = 1, 2, \dots, T$ **do**
- 2: Set $\tilde{\mathbf{s}} = \mathbf{s}$, server samples $\{\mathbf{m}^{(i)}\}_{i=1}^M$ from $p(\mathbf{m}|\tilde{\mathbf{s}})$, sends $f(\cdot; \mathbf{W}_{\mathbf{m}^{(i)}})$ to each worker.
- 3: **for** $r = 1, 2, \dots, \tau$ **do**
- 4: Each worker i sample mini batch $\mathcal{B}^{(i)} = \left\{ \left(\mathbf{x}_1^{(i)}, \mathbf{y}_1^{(i)} \right), \dots, \left(\mathbf{x}_B^{(i)}, \mathbf{y}_B^{(i)} \right) \right\}$.
- 5: Each worker i sends $\mathcal{L}_{\mathcal{B}^{(i)}}(\mathbf{W}_{\mathbf{m}^{(i)}})$ to server and requests $\omega(\mathbf{m}^{(i)})$ from server, updates its own $\mathbf{W}_{\mathbf{m}^{(i)}}$ based on the gradient calculated as:

$$\mathbf{W}_{\mathbf{m}^{(i)}} \leftarrow \mathbf{W}_{\mathbf{m}^{(i)}} - \eta_1 \text{clip}(\omega(\mathbf{m}^{(i)})) \nabla_{\mathbf{W}_{\mathbf{m}^{(i)}}} \mathcal{L}_{\mathcal{B}^{(i)}}(\mathbf{W}_{\mathbf{m}^{(i)}}).$$
- 6: Server collects the loss from M workers and update \mathbf{s} based on the gradient calculated by:

$$\mathbf{s} \leftarrow \text{proj}_{\mathcal{S}}(\mathbf{s} - \eta_2 \mathbf{g}_s^{vr}).$$
- 7: **end for**
- 8: Server collect the $\Delta \mathbf{W}_{\mathbf{m}^{(i)}}$ from M workers and update the whole weights by:

$$\mathbf{W} \leftarrow \mathbf{W} + \frac{1}{M} \sum_{i=1}^M \Delta \mathbf{W}_{\mathbf{m}^{(i)}}.$$
- 9: **end for**
- 10: **return** Pruned DNN $\mathbf{W}_{\mathbf{m}}$, mask \mathbf{m} is sampled from the distribution $p(\mathbf{m}|\mathbf{s})$.

C Variance Reduction Analysis

The impact of variance reduction on training stability is extensively studied, especially SVRG. Roughly, we minus $\mathcal{L}_B(\mathbf{m}) \nabla_{\mathbf{s}} \log(p(\mathbf{m}|\mathbf{s}))$ by a highly correlated and zero mean item in form of $\mathcal{L}_B(\mathbf{m}') \nabla_{\mathbf{s}} \log(p(\mathbf{m}|\mathbf{s}))$ to reduce the variance, which is a big idea in statistics [56]. Figure 3 compares the w/o clip & H setting, showing our method's effectiveness. It can be observed that our variance reduction technique effectively reduces the variance of stochastic gradients, including both Attention heads and MLP channels, and leads to a more stable training process. This provides a practical foundation for our application of policy gradient methods.

D Computation of the Projection Operator.

Following 3, we denote the feasible region of \mathbf{s} in problem (1) as \mathcal{S} , that is $\mathcal{S} = \{\mathbf{s} : \|\mathbf{s}\|_1 \leq (1 - \rho)K \text{ and } \mathbf{s} \in [0, 1]^K\}$. The theorem below shows that the projection of a vector onto \mathcal{S} can be calculated efficiently, which makes the sparsity constraint would always be satisfied during our training process.

Theorem 2. For each vector \mathbf{z} , its projection \mathbf{s} in the set \mathcal{S} can be calculated as follows:

$$\mathbf{s} = \text{proj}_{\mathcal{S}}(\mathbf{z}) = \min(1, \max(0, \mathbf{z} - v_2^* \mathbf{1})), \quad (8)$$

where $v_2^* = \max(0, v_1^*)$ with v_1^* being the solution of the following equation

$$\mathbf{1}^\top [\min(1, \max(0, \mathbf{z} - v_1^* \mathbf{1}))] - (1 - \rho)K = 0. \quad (9)$$

The Eqn.(9) can be solved by bisection method efficiently. Now we can apply policy gradient descent (PGD) to solve problem (1) directly on probability space with explicit sparsity constraint. Theorem 2 is a special case of the problem proposed in [52]. A detailed proof is not the main focus of this paper and can be found in [68].

E Proof of Theorem 1

Notations: For any random variable $X(\mathbf{m})$, we denote $\mathbb{E}X(\mathbf{m})$ to be the expectation over the current distribution $p(\mathbf{m}|\mathbf{s})$, i.e., $\mathbb{E}X(\mathbf{m}) := \mathbb{E}_{\mathbf{m} \sim p(\mathbf{m}|\mathbf{s})} X(\mathbf{m})$; while let $\tilde{\mathbb{E}}X(\mathbf{m})$ be the expectation over the out-dated distribution $p(\mathbf{m}|\tilde{\mathbf{s}})$, i.e., $\tilde{\mathbb{E}}X(\mathbf{m}) := \mathbb{E}_{\mathbf{m} \sim p(\mathbf{m}|\tilde{\mathbf{s}})} X(\mathbf{m})$.

E.1 Unbiasedness of The Gradient

In our method, with the masks \mathbf{m}_i sampled from the out-dated distribution $p(\mathbf{m}_i|\tilde{\mathbf{s}})$, $i = 1, \dots, M$, the variance reduced gradient \mathbf{g}_s^{vr} is defined as follows:

$$\mathbf{g}_s^{vr} = \frac{1}{M-1} \sum_{i=1}^M \text{clip}(\omega(\mathbf{m}^{(i)})) \left(\mathcal{L}_{\mathcal{B}}(\mathbf{m}^{(i)}) - \frac{1}{M} \sum_{j=1}^M \mathcal{L}_{\mathcal{B}}(\mathbf{m}^{(j)}) \right) H^\alpha(\mathbf{s}) \nabla_{\mathbf{s}} \log p(\mathbf{m}^{(i)}|\mathbf{s}).$$

Note that $H^\alpha = \text{diag}(\mathbf{s} \circ (1 - \mathbf{s}))^\alpha$ with $\alpha \in [\frac{1}{2}, 1)$ is a diagonal preconditioning matrix, which plays a role of adaptive step size. $\text{clip}(\cdot)$ is adopted to stabilize the training iterations, which is widely adopted in neural network training. Both $\text{clip}(\cdot)$ and H^α , therefore, to verify \mathbf{g}_s^{vr} is unbiased they need to be ignored. Hence, we define

$$\tilde{\mathbf{g}}_s^{vr} = \frac{1}{M-1} \sum_{i=1}^M \omega(\mathbf{m}^{(i)}) \left(\mathcal{L}_{\mathcal{B}}(\mathbf{m}^{(i)}) - \frac{1}{M} \sum_{j=1}^M \mathcal{L}_{\mathcal{B}}(\mathbf{m}^{(j)}) \right) \nabla_{\mathbf{s}} \log p(\mathbf{m}^{(i)}|\mathbf{s}),$$

and verify it is an unbiased estimation of the gradient $\nabla_{\mathbf{s}} \mathbb{E}_{\mathbf{m} \sim p(\mathbf{m}|\mathbf{s})} \mathcal{L}_{\mathcal{B}}(\mathbf{W}, \mathbf{m})$.

We have the following propositions.

Lemma 1. Suppose the masks \mathbf{m}_i are sampled independently from the out-dated distribution $p(\mathbf{m}|\tilde{\mathbf{s}})$, then $\tilde{\mathbf{g}}_s^{vr}$ is an unbiased stochastic gradient of $\mathbb{E}_{\mathbf{m} \sim p(\mathbf{m}|\mathbf{s})} \mathcal{L}(\mathbf{W}, \mathbf{m})$ when \mathcal{B} is sampled uniformly from the training dataset.

Proof. We rewrite $\tilde{\mathbf{g}}_s^{vr}$ as

$$\tilde{\mathbf{g}}_s^{vr} = \frac{1}{(M-1)M} \sum_{i=1}^M \sum_{j \neq i}^M \omega(\mathbf{m}^{(i)}) \left(\mathcal{L}_{\mathcal{B}}(\mathbf{m}^{(i)}) - \mathcal{L}_{\mathcal{B}}(\mathbf{m}^{(j)}) \right) \nabla_{\mathbf{s}} \log p(\mathbf{m}^{(i)}|\mathbf{s}). \quad (10)$$

Due to the linearity of expectation, we have

$$\begin{aligned} \tilde{\mathbb{E}} \tilde{\mathbf{g}}_s^{vr} &= \frac{1}{(M-1)M} \sum_{i=1}^M \sum_{j \neq i}^M \tilde{\mathbb{E}}_{\mathbf{m}_i, \mathbf{m}_j} \omega(\mathbf{m}^{(i)}) \left(\mathcal{L}_{\mathcal{B}}(\mathbf{m}^{(i)}) - \mathcal{L}_{\mathcal{B}}(\mathbf{m}^{(j)}) \right) \nabla_{\mathbf{s}} \log p(\mathbf{m}^{(i)}|\mathbf{s}) \\ &= \frac{1}{(M-1)M} \sum_{i=1}^M \sum_{j \neq i}^M \tilde{\mathbb{E}}_{\mathbf{m}_i} \omega(\mathbf{m}^{(i)}) \mathcal{L}_{\mathcal{B}}(\mathbf{m}^{(i)}) \nabla_{\mathbf{s}} \log p(\mathbf{m}^{(i)}|\mathbf{s}) \end{aligned} \quad (11)$$

$$\begin{aligned} &\stackrel{(4)}{=} \frac{1}{(M-1)M} \sum_{i=1}^M \sum_{j \neq i}^M \nabla_{\mathbf{s}} \mathbb{E}_{\mathbf{m}_i} \mathcal{L}_{\mathcal{B}}(\mathbf{m}^{(i)}) \\ &= \nabla_{\mathbf{s}} \mathbb{E}_{\mathbf{m} \sim p(\mathbf{m}|\mathbf{s})} \mathcal{L}_{\mathcal{B}}(\mathbf{m}). \end{aligned} \quad (12)$$

The Eqn.(11) holds since when $i \neq j$, we have

$$\begin{aligned}
& \tilde{\mathbb{E}}_{\mathbf{m}_i, \mathbf{m}_j} \omega(\mathbf{m}^{(i)}) \mathcal{L}_{\mathcal{B}}(\mathbf{m}^{(j)}) \nabla_{\mathbf{s}} \log p(\mathbf{m}^{(i)} | \mathbf{s}) \\
&= \tilde{\mathbb{E}}_{\mathbf{m}_j} \mathcal{L}_{\mathcal{B}}(\mathbf{m}^{(j)}) \tilde{\mathbb{E}}_{\mathbf{m}_i} \omega(\mathbf{m}^{(i)}) \nabla_{\mathbf{s}} \log p(\mathbf{m}^{(i)} | \mathbf{s}) \\
&= \tilde{\mathbb{E}}_{\mathbf{m}_j} \mathcal{L}_{\mathcal{B}}(\mathbf{m}^{(j)}) \tilde{\mathbb{E}}_{\mathbf{m}_i} \frac{1}{p(\mathbf{m}^{(i)} | \tilde{\mathbf{s}})} \nabla_{\mathbf{s}} p(\mathbf{m}^{(i)} | \mathbf{s}) \\
&= \tilde{\mathbb{E}}_{\mathbf{m}_j} \mathcal{L}_{\mathcal{B}}(\mathbf{m}^{(j)}) \nabla_{\mathbf{s}} \mathbb{E}_{\mathbf{m}_i} 1 \\
&= 0.
\end{aligned}$$

Therefore, $\tilde{\mathbf{g}}_s^{vr}$ is an unbiased stochastic gradient of $\mathbb{E}_{\mathbf{m} \sim p(\mathbf{m} | \mathbf{s})} \mathcal{L}_{\mathcal{B}}(\mathbf{W}, \mathbf{m})$ as well as $\mathbb{E}_{\mathbf{m} \sim p(\mathbf{m} | \mathbf{s})} \mathcal{L}(\mathbf{W}, \mathbf{m})$. \square

E.2 Bounded Variance

Theorem 1. Suppose the masks $\{\mathbf{m}^{(i)}\}_{i=1}^M$ are independently sampled from the Bernoulli distribution $p(\mathbf{m} | \tilde{\mathbf{s}})$, then for any $\alpha \in [\frac{1}{2}, 1)$ and $\mathbf{s} \in [0, 1]^K$, the variance \mathbf{g}_s^{vr} is bounded.

Before giving the proof, we need to present the following two empirically verified assumptions/properties in the work [67].

Property 1. Given the probability \mathbf{s} and the weights \mathbf{W} for an overparameterized deep neural network, then for two independent masks \mathbf{m} and \mathbf{m}' sampled from $p(\cdot | \mathbf{s})$, $\mathcal{L}(\mathbf{W}, \mathbf{m}) - \mathcal{L}(\mathbf{W}, \mathbf{m}')$ is always small. That is

$$V(\mathbf{s}) := \mathbb{E}_{\mathbf{m} \sim p(\cdot | \mathbf{s})} \mathbb{E}_{\mathbf{m}' \sim p(\cdot | \mathbf{s})} (\mathcal{L}(\mathbf{m}) - \mathcal{L}(\mathbf{m}'))^2 \quad (13)$$

is small.

Property 2. Given the probability \mathbf{s} and the weights \mathbf{w} for an overparameterized deep neural network, consider a mask \mathbf{m} sampled from $p(\cdot | \mathbf{s})$, if we flip one component of \mathbf{m} , then the network would not change too much. Combined with Property 1, this can be stated as: for any $j \in \mathcal{C}$, we denote \mathbf{m}_{-j} and \mathbf{s}_{-j} to be all the components of \mathbf{m} and \mathbf{s} except the j -th component, and define

$$V_{\max}(\mathbf{s}) := \max_{\mathbf{m}_j \in \{0,1\}, j \in \mathcal{C}} \mathbb{E}_{\mathbf{m}_{-j} \sim p(\cdot | \mathbf{s}_{-j})} \mathbb{E}_{\mathbf{m}' \sim p(\cdot | \mathbf{s})} (\mathcal{L}(\mathbf{m}) - \mathcal{L}(\mathbf{m}'))^2,$$

then

$$V_{\max}(\mathbf{s}) \approx V(\mathbf{s}). \quad (14)$$

Proof. of Theorem 1:

The total variance of \mathbf{g}_s^{vr} is

$$\text{Var}(\mathbf{g}_s^{vr}) = \tilde{\mathbb{E}} \|\mathbf{g}_s^{vr}\|_2^2 - \|\tilde{\mathbb{E}} \mathbf{g}_s^{vr}\|_2^2. \quad (15)$$

Thus, we only need to prove $\tilde{\mathbb{E}} \|\mathbf{g}_s^{vr}\|_2^2$ is bounded.

Similar with Eqn.(10), we rewrite \mathbf{g}_s^{vr} as

$$\mathbf{g}_s^{vr} = \frac{1}{(M-1)M} \sum_{i=1}^M \sum_{j \neq i} \underbrace{\text{clip}(\omega(\mathbf{m}^{(i)})) \left(\mathcal{L}_{\mathcal{B}}(\mathbf{m}^{(i)}) - \mathcal{L}_{\mathcal{B}}(\mathbf{m}^{(j)}) \right) H^\alpha(\mathbf{s}) \nabla_{\mathbf{s}} \log p(\mathbf{m}^{(i)} | \mathbf{s})}_{:= \tilde{\mathbf{g}}^{ij}}.$$

Let

$$\tilde{\mathbf{g}}^{ij} = \omega(\mathbf{m}^{(i)}) \left(\mathcal{L}_{\mathcal{B}}(\mathbf{m}^{(i)}) - \mathcal{L}_{\mathcal{B}}(\mathbf{m}^{(j)}) \right) H^\alpha(\mathbf{s}) \nabla_{\mathbf{s}} \log p(\mathbf{m}^{(i)} | \mathbf{s}),$$

we know that

$$\|\tilde{\mathbf{g}}^{ij}\|^2 \leq \|\mathbf{g}^{ij}\|^2.$$

Therefore, we only need to prove $\tilde{\mathbb{E}}\|\mathbf{g}^{ij}\|^2$ is bounded for any $i \neq j$. We analyze the k -th component of \mathbf{g}_k^{ij} as follows:

$$\begin{aligned} \tilde{\mathbb{E}}\|\mathbf{g}_k^{ij}\|^2 &= \tilde{\mathbb{E}}\omega^2(\mathbf{m}^{(i)}) \left(\mathcal{L}_{\mathcal{B}}(\mathbf{m}^{(i)}) - \mathcal{L}_{\mathcal{B}}(\mathbf{m}^{(j)}) \right)^2 [H^\alpha(\mathbf{s}) \nabla_{\mathbf{s}} \log p(\mathbf{m}^{(i)}|\mathbf{s})]_k^2 \\ &= \tilde{\mathbb{E}}\omega^2(\mathbf{m}^{(i)}) \left(\mathcal{L}_{\mathcal{B}}(\mathbf{m}^{(i)}) - \mathcal{L}_{\mathcal{B}}(\mathbf{m}^{(j)}) \right)^2 \left(s_k^{2(\alpha-1)} (1-s_k)^{2(\alpha-1)} (\mathbf{m}_k - s_k)^2 \right) \\ &= \tilde{\mathbb{E}}_{\mathbf{m}_k^{(i)}} \tilde{\mathbb{E}}_{\mathbf{m}_{-k}^{(i)}} \tilde{\mathbb{E}}_{\mathbf{m}^{(j)}} \frac{\omega(\mathbf{m}^{(i)})}{\omega(\mathbf{m}^{(j)})} \omega(\mathbf{m}^{(i)}) \omega(\mathbf{m}^{(j)}) \left(\mathcal{L}_{\mathcal{B}}(\mathbf{m}^{(i)}) - \mathcal{L}_{\mathcal{B}}(\mathbf{m}^{(j)}) \right)^2 \\ &\quad \left(s_k^{2(\alpha-1)} (1-s_k)^{2(\alpha-1)} (\mathbf{m}_k^{(i)} - s_k)^2 \right) \\ &\leq r_{\max} V_{\max}(\tilde{\mathbf{s}}) \tilde{\mathbb{E}}_{\mathbf{m}_k^{(i)}} \frac{s_k^{\mathbf{m}_k^{(i)}} (1-s_k)^{(1-\mathbf{m}_k^{(i)})}}{\tilde{s}_k^{\mathbf{m}_k^{(i)}} (1-\tilde{s}_k)^{(1-\mathbf{m}_k^{(i)})}} \left(s_k^{2(\alpha-1)} (1-s_k)^{2(\alpha-1)} (\mathbf{m}_k^{(i)} - s_k)^2 \right) \\ &= r_{\max} V_{\max}(\tilde{\mathbf{s}}) \tilde{\mathbb{E}}_{\mathbf{m}_k^{(i)}} \left(s_k^{2(\alpha-1)} (1-s_k)^{2(\alpha-1)} (\mathbf{m}_k^{(i)} - s_k)^2 \right) \\ &\leq r_{\max} |\mathcal{C}| V_{\max}(\tilde{\mathbf{s}}), \end{aligned}$$

where $r_{\max} := \max \frac{\omega(\mathbf{m}^{(i)})}{\omega(\mathbf{m}^{(j)})}$. Note that r_{\max} is the ratio of two reweightings of two masks sampled from the same distribution $p(\cdot|\mathbf{s})$ independently, which are always close and thus we can regard the ratio as a bounded value. Hence $\tilde{\mathbb{E}}\|\mathbf{g}_k^{ij}\|^2$ is bounded and thus $\tilde{\mathbb{E}}\|\mathbf{g}^{ij}\|^2$ is bounded. \square

F Limitation

Compared to compressing conventional vision models, pruning methods for LLMs including our approach typically result in a larger performance drop after pruning. The reasons could be: 1) resources for pruning are not on par with those used in training, while traditional vision models use comparable resources for both; 2) the redundancy in LLMs may be relatively low, which is unclear with the current insufficient resources for pruning. In light of this, developing computation-efficient approaches like ours is particularly valuable, as only such methods can enable sufficient optimization under constrained resources.

G Broader Impacts

Model pruning algorithms inherently represent a socially beneficial research direction, as the resulting compact models significantly reduce carbon emissions during both inference and training phases. This effectively mitigates the environmental impact of AI development. Moreover, these efficient models lower the computational resource requirements, enabling wider participation in AI innovation across research institutions and individual developers.

As demonstrated by our key contribution, e.g., minimizing resource consumption during pruning while maintaining performance, we provide the community with more efficient pruning algorithms and better-performing models. We believe resource-constrained downstream users will substantially benefit from this advancement, which will effectively accelerate progress in related fields.

H Generations from Compressed Llama2-7B

Table 15 presents the generation results of the model at different prune rates using our algorithm. We used Llama2-7B for generation, and the results show that even at high sparsity levels (50%), the

pruned model can still generate reasonable outputs. Notably, at a 30% prune rate, the model’s output appears more logical, such as providing a detailed list of 10 suggestions. This suggests our pruning algorithm removes redundant parameters, possibly improving the model’s reasoning ability.

Table 15: Generation results from the Llama2-7B at 0%, 30%, 50% prune rates. We set the generation sequence length to 128. The underlined words are the input prompts, and the rest are generated by the pruned models.

Prune Rate	Generation Sentences
0%	I believe the meaning of life is to get everything we can out of life. If we think of God as the Great Unmoved Mover, then who moves us? There is an inherent danger in thinking that our suffering is meaningless. Suffering and meaninglessness are not two sides of the same coin. It is a fact that some suffering is meaningless.
	Simply put, the theory of relativity states that 1) the speed of light is a constant for all observers, and 2) the effects of gravity and acceleration on time are the same in the laboratory and in a moving frame of reference. The relativistic velocity transform law defines the relationship of inertial reference systems. In general relativity, the theory of relativity is used to explain gravitation and other phenomena involving acceleration by replacing classical mechanics with the theory of general relativity (see below). General relativity is a theory of gravitation in accordance with Newton’s laws.
	Building a website can be done in 10 simple steps: 1. Set Your Domain, Buy Your Domain, and Set Up Your Website The first step is to decide on a domain name that will make sense for your website. As far as the domain goes, you can do one of three things. You can use a web host to buy your domain. If you decide to host on a site like GoDaddy.com, your domain will cost around \$9.99 a year. Or you can host with your domain yourself. Hosting with a domain yourself will cost around \$10 a year.
30%	I believe the meaning of life is the following: we have no control over the circumstances that we find ourselves in. The only way to make an effort to keep things from getting worse is to make an effort to improve them. It is my understanding that the people who achieve this level of happiness are doing all that they can. There will be highs and lows in all that they attempt to create an environment and situation for themselves that is safe and enjoyable, no matter what happens. You cannot control what happens to you, however you can choose what you do. Inside our mind are those that are positive, self
	Simply put, the theory of relativity states that 1) all frames are equivalent and therefore frames and time are related and 2) inertial mass is equivalent to the gravitational mass of a body and thus accelerating mass frames (at least on a point of view) could be treated as frames. Except of course, this theory was already devised by many other scientists long before Newton did so. But Newton’s Law of Inertial Mass and Frames was only one part of Newton’s work. Newton did two major discoveries in physics. Newton postulated the laws of mechanics
	Building a website can be done in 10 simple steps: 1) Buy domain name. 2) Pay for site hosting. 3) Build web layout. 4) Buy images. 5) Set site’s colors. 6) Create content. 7) Select layout. 8) Pick logo and/or add new content. 9) Build HTML and CSS. 10) Build content. As you can see, these are just some basic steps involved in creating a website. The process of building a website becomes a lot easier for people without programming or technical knowledge.
50%	I believe the meaning of life is the perception that it can be lived. If we think of life as the time we have on this earth, then we have nothing. Therefore, it is the quality of our lives. If we live it such as is and have the potential to see and experience things, we will find this to be a correct statement. Our lives can be lived or lived based on our decisions. If we live it in love, life, and joy, we will experience life as it is lived, and it will be lived. If we live it in joy and
	Simply put, the theory of relativity states that 5/30% of the Universe is composed of matter, while 5/30% of that Universe is composed of energy and space. These observations from the same stars are not confirmed by the physics of quantum. The physics of quantum have been proved to be true for the Universe to be composed of energy and space and not of time and energy. If you wish to obtain fuller descriptions concerning the Universe, be sure to have access to these laws. If you wish to observe the Universe and its laws, you
	Building a website can be done in 10 simple steps:-Designing a template: a template or layout to convey the content (i. Preparedesign.com) -An example layout for this template includes designing a template layout that incorporates all features of your template layout-Designing a template that is designed for you to use in a social media platform (social media included) -Revisiting your website template (as above) and re-designing layout template to incorporate your new layout layout, then re-designing layout template