Dual-Assessment Driven Pruning: Iterative Optimizing Layer-wise Sparsity for Large Language Model

Qinghui Sun
Alibaba Group
Hangzhou, China
yuyang.sqh@alibaba-inc.com

Weilun Wang
Alibaba Group
Hangzhou, China
wangweilun.wwl@alibaba-inc.com

Yanni Zhu
Alibaba Group
Hangzhou, China
tangyue.zyn@alibaba-inc.com

Shenghuan He
Alibaba Group
Hangzhou, China
heshenghuan.hsh@alibaba-inc.com

Hao Yi
Alibaba Group
Hangzhou, China
yihao.yh@alibaba-inc.com

Zehua Cai
Alibaba Group
Hangzhou, China
zehua.czh@alibaba-inc.com

Hong Liu
Alibaba Group
Hangzhou, China
liuhong.liu@alibaba-inc.com

ABSTRACT

Large Language Models (LLMs) have demonstrated efficacy in various domains, but deploying these models is economically challenging due to extensive parameter counts. Numerous efforts have been dedicated to reducing the parameter count of these models without compromising performance, employing a technique known as model pruning. Conventional pruning methods assess the significance of weights within individual layers and typically apply uniform sparsity levels across all layers, potentially neglecting the varying significance of each layer. To address this oversight, we first propose a dual-assessment driven pruning strategy that employs both intra-layer metric and global performance metric to comprehensively evaluate the impact of pruning. Then our method leverages an iterative optimization algorithm to find the optimal layer-wise sparsity distribution, thereby minimally impacting model performance. Extensive benchmark evaluations on state-of-the-art LLM architectures such as LLaMAv2 and OPT across a variety of NLP tasks demonstrate the effectiveness of our approach. When applied to the LLaMaV2-7B model with an overall pruning sparsity of 80%, our method achieves a 50% reduction in perplexity compared to the benchmark. The results indicate that our method significantly outperforms existing state-of-the-art methods in preserving performance after pruning.

CCS CONCEPTS

• **Computing methodologies** → Natural language processing.

KEYWORDS

Pruning, sparsity, large language models, iterative optimization

ACM Reference Format:
Qinghui Sun, Weilun Wang, Yanni Zhu, Shenghuan He, Hao Yi, Zehua Cai, and Hong Liu. 2024. Dual-Assessment Driven Pruning: Iterative Optimizing Layer-wise Sparsity for Large Language Model. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3637528.3671780

KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 979-8-4007-0490-1/24/08
https://doi.org/10.1145/3637528.3671780

1 INTRODUCTION

The emergence of Large Language Models (LLMs) like the Generative Pretrained Transformer (GPT) family[1][2][3][4] has revolutionized the field of Natural Language Processing (NLP), demonstrating unprecedented performance across diverse language tasks. However, due to their billions of parameters, the deployment and application of large language models require high computational costs and GPU resource consumption. Consequently, it becomes imperative to investigate advanced model compression techniques to ensure their sustainable application and development.

In the field of LLMs, attention is increasingly drawn to model compression methods to make these models more accessible. Among various compression strategies, quantization and pruning capture a substantial interest from the community[5][6][7][8][9][10]. Quantization methods reduce the size of models by decreasing the number of bits required to represent each weight, alleviating the demand for storage without reducing the total count of weights. The advantage of quantization is that it offers a structured approach to model compression, making it more storage-efficient and widely applicable. In contrast, pruning strategies represent an alternative method for model compression and are equally worth exploring. Initially proposed in 1989[11], it reduces the model size by removing less significant weights or structures. Although effective in vision and smaller NLP models, pruning poses unique challenges when applied to LLMs. To preserve model performance, traditional pruning methods[12][13] typically necessitate retraining weights with the full volume of original training corpora, a process that

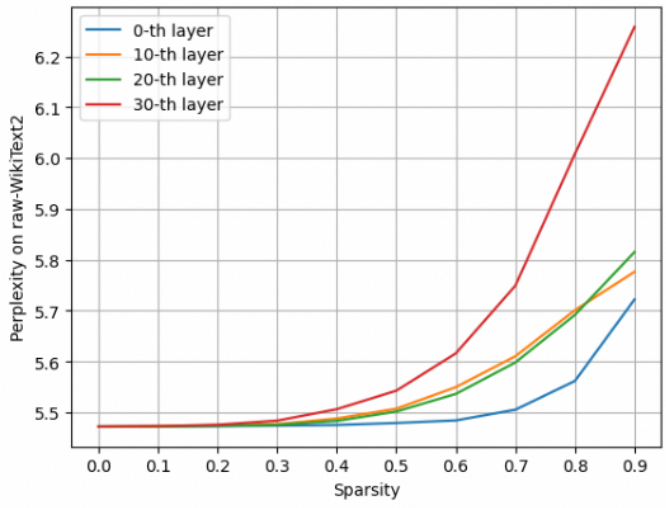


Figure 1: Pruning impact of different transformer layers with different sparsity on LLaMAv2-7B

follows the pruning stage. However, for LLMs, the acquisition of original training corpora and the consumption of training resources are prohibitively expensive. In this context, Post-training pruning methods are gaining popularity because they simplify the pruning process without the need for retraining. Nonetheless, early methods such as Magnitude Pruning[14] proved inadequate because reducing weights by just 10% led to significant performance drops, and a 30% reduction resulted in substantial deterioration of the model's performance.

These results reveal that using only weight magnitude to guide pruning LLMs is insufficient for precisely targeting the weights that should be pruned. Recent advancements show that pruning is possible for LLMs while maintaining performance. SparseGPT [15] leverages second-order derivative information to select which weights to prune, managing to halve the model size with minimal performance loss. Meanwhile, Wanda[16] exploits the novel magnitude features, which are products of weights and input activations, to identify which weights to drop. The results show that it can achieve pruning performance that competes with SparseGPT even without the need to compute second-order derivatives. Both methods provide an intra-layer metric to precisely evaluate the importance of weights within each layer and perform pruning at a fixed ratio for each layer, resulting in uniform sparsity throughout all layers.

While these methods mark significant progress in the field of LLM pruning, in our analysis, we identify two inherent issues that warrant further consideration. First, the specific metrics might not accurately reflect the true impact of a weight on the whole model. A weight that seems unimportant in one layer could be vital in the context of the entire model. To illustrate, consider a weight that is among the top 50% within its layer and thus remains unpruned. From the model's overall perspective, this weight might not be in the global top 50% in terms of importance. Second, from a broader perspective, the fixed sparsity of each layer may not be ideal. To further support our perspective, we conducted an experiment focusing on the impact of pruning across several transformer layers within the LLaMAv2-7B model. From Figure 1, we can observe that when only a small ratio of the weights is pruned, the performance of each layer is almost indistinguishable. However, as the pruning ratio increases, variations between layers start to emerge progressively. The weights of certain layers are highly insensitive to pruning. For example, in layer 0, reducing 70% of the weights has almost no impact on model performance. It indicated that a higher pruning ratio should be allocated to the layer, while the more sensitive layers should be assigned a lower pruning ratio. Our analysis concludes that allocating the same sparsity to each layer is unreasonable, the viewpoint further corroborated by recent advancements. Specifically, OWL[17] introduced a method that employs the proportion of outliers in each layer to determine the sparsity ratio distribution across the model, supporting our conclusion of differential layer sensitivity. However, this approach requires tuning hyperparameters uniquely for each model, which increases complexity and limits its applicability. Furthermore, the reliance on outlier-based sparsity calculations does not precisely capture the importance of each layer.

Contrasting with earlier methods that solely assess the importance of weights within individual layers [15][16][17], our approach not only retains this intra-layer evaluation but also introduces a novel metric to precisely gauge the global importance of each layer. This dual-assessment strategy enables us to identify the optimal sparsity ratio distribution across the entire model. In our work, we first assess the sensitivity of each layer to prune by evaluating the performance loss at different levels of sparsity. This process identifies the degree of pruning that each layer can endure before it significantly impairs the model performance. For example, if testing reveals that performance loss of a layer is tolerable at 50% sparsity, we continue to incrementally increase sparsity to determine the optimal level. Based on the assessment results, we adjust the sparsity levels for each layer to match its pruning tolerance. Layers that demonstrate minor performance degradation under higher sparsity are assigned with a more aggressive pruning ratio, while those more sensitive to sparsity are assigned with a more modest pruning ratio. By matching sparsity to the pruning tolerance of each layer, the method improves model performance compared to the pruning methods with uniform sparsity.

The main contributions of this work are summarized as follows:

(1) We offer a novel global performance metric based on KL divergence, which not only assists in identifying the ideal sparsity level for each layer within the model but also provides a rapid and lightweight evaluation metric to assess the impact on model performance when the model is pruned.

(2) We proposed an iterative algorithm that continuously adjusts the sparsity levels of individual layers, informed by the dual-assessment strategy, which simultaneously considers intra-layer evaluations and global performance assessments. This approach aims to identify an optimal distribution of sparsity across layers, aligning with the overall performance goal.

(3) We present comprehensive benchmark evaluations on leading LLM architectures across various NLP tasks. The results show that our method substantially reduces perplexity and preserves performance on complex language tasks compared with other pruning methods.

2 RELATED WORK

2.1 Pruning and LLM Pruning

Network pruning is a powerful technique that reduces the size or complexity of a model by removing unnecessary weights or redundant components, while maintaining its performance. This technique was proposed relatively early, and in recent decades, a substantial amount of research has emerged[18][19][20][21][22]. However, progress in pruning LLMs has been modest. Traditional pruning methods often necessitate re-training to recover performance, posing difficulties for LLMs. To overcome this, specialized pruning algorithms for LLM compression have emerged. For instance, LLM-Pruner[23] investigated structured sparse LLMs using Taylor pruning to remove entire weight rows, followed by LoRA fine-tuning for knowledge recovery. Subsequently, LoRAPrune[24] and LoRAShear[25] performed optimizations to enhance compatibility and performance. Recent research has made significant progress in unstructured pruning without fine-tuning. SparseGPT[15] utilizes Hessian inverse for pruning and with subsequent weight updates to reduce reconstruction error of dense and sparse weights, it formalizes the problem of pruning LLMs by solving a local layer-wise reconstruction problem, where their pruning metric and weight update procedure is inspired from Optimal Brain Surgeon (OBS)[13]. Wanda[16] use weight and activations as the pruning metric, requiring no gradient computation via back-propagation or any second-order Hessian inverses, and entailing no weight update on pruned networks, it reduces the computational complexity significantly. Traditional LLM pruning methods, including approaches like Wanda and SparseGPT, concentrate on evaluating the significance of weights within individual layers, often overlooking the fact that different layers may contribute unequally to the model's overall performance. OWL[17] explores the important role of non-uniform hierarchical sparsity in LLM pruning and proposed a new technology that uses outlier distribution to guide hierarchical LLM pruning. It introduces a new stratified sparsity ratio, outlier-weighted stratified sparsity, which is proportional to the ratio of outliers observed at each layer.

2.2 Layerwise Sparsity for Pruning

While uniform layerwise sparsity is typically employed in pruning language models, research has also explored non-uniform sparsity for vision models. Mocanu et al.[26] introduced a non-uniform, scale-free topology derived from graph theory that outperformed the dense counterpart in restricted Boltzmann machines. Later developments enhanced this approach's scalability using the Erdos-Renyi graph, extending its use to both fully connected layers (Mocanu et al., 2018[20]) and convolutional layers (Evci et al., 2020[27]; Liu et al., 2022[28]) without relying on data or feedforward. Additionally, other studies have achieved non-uniform sparsity by setting a global threshold across layers (Frankle Carbin, 2019[29]; Lee et al., 2020[30]; Liu et al., 2021[31]). Recently, inspired by the visual model pruning method above, OWL[17] attempts to introduce non-uniform layer sparsity into large language models, which proves to be effective and worthy of further investigation.

3 METHOD

3.1 Preliminary

Intra-layer Weight Importance Metric: "Intra-layer Weight Importance Metric" refers to a method for quantifying the importance of weights within a linear layer, which can be utilized to determine which parameters should be pruned or retained. Our strategy can be compatible with various weight importance metrics, such as Wanda[16], SparseGPT[15]. For Wanda, the metric involves calculating the product of the weight matrix W and the L2 norm of the activation matrix $||X||_2$, which serves as an estimation of each weight's significance:

$$S = |W| \cdot ||X||_2 \tag{1}$$

In the context of SparseGPT, the metric employs the inverse of the Hessian matrix H^{-1} to measure the importance of each weight. Both methods demonstrate effectiveness and high efficiency in evaluating weight importance for large-scale models.

Layer-wise Sparsity Distribution: Language models employing transformer architectures consist of multiple transformer layers, with each transformer layer potentially contributing differently to the overall functionality of the model. In light of this, it may be posited that the information required to be dropped in each transformer layer could vary. The sparsity denotes the proportion of weights that are dropped in each layer. The desired total sparsity is denoted by r, and we introduce a layer-wise sparsity vector $R = \{r_0, r_1, r_2, \ldots, r_N\}$, where r_i represents the sparsity of the i^{th} transformer layer. In the context of unstructured pruning, examples of r_i include 50% and 60%. For N: M structured pruning, possible r_i might be 2:8, 3:8, 4:8, and so forth. Our strategy ensures that the average sparsity across the layers, weighted by the number of parameters in each layer, matches the target global sparsity r. This is expressed by the following conditions:

$$\frac{\sum_{i=0}^{N} a_i \cdot r_i}{\sum_{i=0}^{N} a_i} = r \tag{2}$$

where a_i denotes the number of parameters in the i^{th} transformer layer, and the numerator represents the total count of pruned parameters across the model.

3.2 Pruning Algorithm

In this section, we detail the Dual-Assessment Driven Pruning algorithm. Specifically, in our method, while we preserve the metric at the intra-layer level, we further proposed a global performance metric to gauge the importance of each transformer layer across the model. Based on the dual-assessment strategy, we designed an iterative optimization algorithm to gauge each layer's global importance precisely.

Global Performance Metric: The intra-layer metric is constrained to the assessment of weights' significance within individual layers, neglecting the crucial task of evaluate a layer's overall contribution to the model's architecture. To bridge this deficiency, we present a global performance metric designed specifically to gauge the relative importance of each layer across the entire model. In addition, our proposed metric is both lightweight and efficient,

Algorithm 1 Iterative Optimizing Layer-wise Sparsity Algorithm

Input: The original model M, the pruned model M^*, adjustment step s, the input is x, N is the number of transformer layers in the model, and target sparsity is r. The initial sparsity distribution $R = \{r_0, r_1, r_2, \ldots, r_N\}$.

Output: Sparsity distribution $R = \{r_0, r_1, r_2, \ldots, r_N\}$.

Initialization uniform pruned $R = \{r_i = r, i \leq N\}$ and original model word distribution $q(x) = M(x)$, $p(x) = M^*(x)$. Initialization loss is $L = D_{KL}(p||q)$
while $r_i \geq 0$ and $u \neq d$ and $L^* \leq L$ **do**
  for each $i \in N$ **do**
    $r_i = r_i + s$
    $p(x) = M^*(x)$
    $U_i = D_{KL}(p_i||q)$
    $r_i = r_i - s$
  end for
  Find the smallest layer id $u = \arg\min U_i$.
  $r_u = r_u + s$
  for each $i \in N$ **do**
    $r_i = r_i - s$
    $p(x) = M^*(x)$
    $D_i = D_{KL}(p_i||q)$
    $r_i = r_i + s$
  end for
  Find the largest layer id $d = \arg\min D_i$.
  $r_d = r_d - s$
  Calculate new loss $L^* = D_{KL}(p||q)$
end while

which facilitates expedited iterative pruning, enhancing the optimization process. Recognizing that large language models generate predictions as probability distributions over a vocabulary, we use the distribution of the original model as a standard reference. By comparing this with the distribution of the pruned model, we employ the Kullback-Leibler (KL) divergence to measure the similarity between the two. The pruned model is denoted as M^* and the original model as M. The prediction word distribution for the pruned model is expressed as $p(x) = M^*(x)$, and for the original model as $q(x) = M(x)$, where x is the input to both models. We then calculate the KL divergence as follows:

$$D_{KL}(p \parallel q) = \sum p(x) \log \frac{p(x)}{q(x)} \tag{3}$$

The rationale is simple: the higher the similarity (or the lower the KL divergence), the less the loss in performance due to pruning. This approach allows for a swift assessment of the pruning effect on the vocabulary's probability distributions generated by the model, thus enabling real-time adjustments and optimization during the pruning process. Its computational efficiency makes it particularly well-suited for scenarios that require rapid iteration.

Iterative Sparsity Distribution Optimization: Our algorithm begins by setting a global sparsity target for the model and uniformly pruning each layer to achieve an initial baseline. The objective is to iteratively refine the Layer-wised sparsity distribution to find the optimal pattern R. The optimal pattern is used to minimize the global metric previously introduced, which can be defined as follows:

$$\arg\min_{R \in \boldsymbol{R}} D_{KL}(p||q) \tag{4}$$

To minimize the metric indicating pruning-induced performance loss, we individually increase the sparsity level of each transformer layer by a fixed increment, then calculate the performance loss using our metric, and select the layer with the minimum loss which indicates that the layer has the strongest pruning-tolerance under the current pruning settings. Since our objective is to maintain the overall sparsity of the model, we then decrease sparsity in another transformer layer by an equivalent amount, following a similar process of loss measurement and selection. The algorithm converges when a further reduction in the performance loss is no longer attainable or when the transformer layer selected for increasing sparsity is the same layer chosen for decreasing sparsity. To systematize this process, we structure the algorithm into three principal phases:

Stage 1: During the first phase, we individually increase the sparsity level of each transformer layer by a fixed step size denoted as s, which represents the increment of sparsity adjustment. For each transformer layer indexed by i, with i ranging from 0 to N where N is the number of transformer layers in the model, we calculate the word distribution $p_i(x)$ that corresponds to the pruned model state, denoted by $M^*(x)$. Next, we compute the performance loss for this pruned configuration using the KL divergence metric:

$$U_i = D_{KL}(p_i \parallel q), \quad 0 \leq i \leq N \tag{5}$$

We then identify the index of the transformer layer with the minimum U_i, which is the layer index u:

$$u = \arg\min_i U_i \tag{6}$$

Stage 2: Subsequently, in the second phase, to maintain constant sparsity throughout the pruning, we decrease the sparsity of each transformer layer by s. We compute the loss:

$$G_i = D_{KL}(p_i \parallel q), \quad 0 \leq i \leq N \tag{7}$$

We find the index of the layer with the smallest G_i, denoted as g:

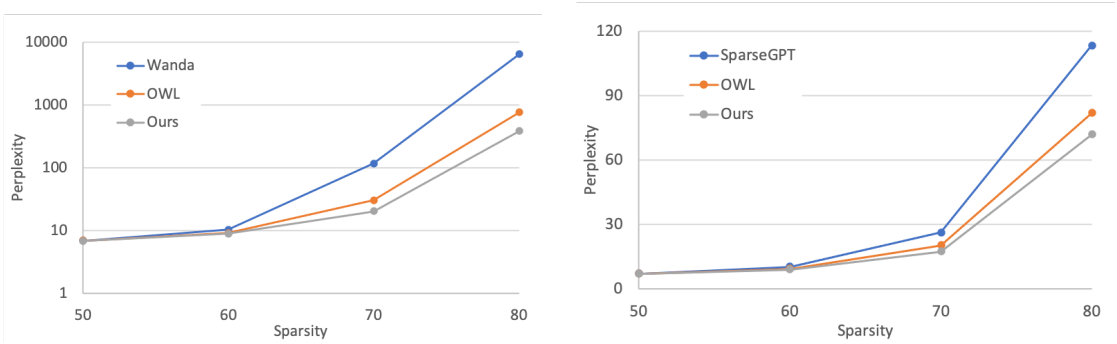
$$g = \arg\min_i G_i \tag{8}$$

Stage 3: Finally, we adjust the sparsity of the u-th layer by increasing it by s and decreasing the sparsity of the g-th layer by s to ensure the global sparsity level remains constant. Following the adjustments made during these three stages, we consider the newly established sparsity distribution as the baseline model and proceed to calculate its loss. This three-stage cycle is repeated until either the condition $u = g$ is satisfied, which indicates that the same layer is selected for both sparsity increase and decrease or until there is no further reduction in loss. The entire algorithmic process is shown in Algorithm 20. Our experimental results demonstrate that this algorithm can significantly reduce errors.

It is noteworthy that our method, while primarily employed to identify the optimal sparsity distribution for transformer layers, is theoretically capable of determining the ideal sparsity distribution across all linear layers globally. This broader applicability underscores the versatility of our approach. However, our focus in this work remains on the sparsity distribution within transformer layers.

Method	Sparsity	OPT Family		LLaMAv2 Family	
		OPT-1.3B	OPT-6.7B	LLaMAv2-7B	LLaMAv2-13B
Dense	0%	14.62	10.86	5.47	4.88
Wanda	50%	18.36	**10.31**	6.89	5.96
OWL	50%	20.02	12.21	6.85	5.93
Ours	50%	**16.65**	11.43	**6.82**	**5.90**
Wanda	60%	26.66	15.01	10.28	8.19
OWL	60%	31.93	15.55	9.17	7.51
Ours	60%	**23.85**	**13.84**	**8.93**	**7.38**
Wanda	70%	87.63	456.99	115.47	53.65
OWL	70%	153.99	43.23	30.14	17.91
Ours	70%	**72.60**	**28.84**	**20.27**	**13.62**
Wanda	80%	2589.50	7569.36	6367.23	1924.06
OWL	80%	2121.91	14015.85	755.06	241.36
Ours	80%	**954.44**	**604.75**	**381.19**	**180.36**

Table 1: WikiText validation perplexity of pruning methods for LLaMAV2-7B/13B and OPT-1.3B/6.7B at 50%-80% sparsity. Our approach and OWL both employ Wanda as the method for intra-layer weight selection. The best performance method is indicated in bold.



(a) Wanda (b) SparseGPT

Figure 2: The WikiText perplexity of our method was applied to SparseGPT and Wanda at various sparsities. In the case of Figure2(a), both our method and OWL utilize Wanda as the method for intra-layer weight selection. For Figure2(b), our approach and OWL employ SparseGPT as the method for intra-layer weight selection.

Additionally, it is important to highlight that our approach involves global performance metrics, which require multiple forward inferences to pinpoint the critical distribution of sparsity. Nevertheless, we argue that the computational efforts invested in forward inferences during the pruning phase constitute just a small portion of the total operational time post-deployment. We assert that the additional investment of time and resources during the pruning stage is justified by the significant improvements in the model's performance.

4 EXPERIMENTS

Large Language Models (LLMs) have demonstrated their efficacy across a spectrum of applications. Recent research[32] indicates that reliance on metrics like perplexity alone is insufficient for accurately gauging the performance of pruned models on diverse tasks. In this section, we establish a range of benchmarks, encompassing both Perplexity for language modeling and a suite of more complex NLP tasks, such as various zero-shot challenges. This will allow us to conduct a thorough comparative analysis of the models in question.

Model and Dataset We evaluate methods on the two prominent LLM model families: OPT-1.3B/6.7B[33] and LLaMAv2-7B/13B[34]. We evaluate the performance of pruned models on language modeling and zero-shot tasks. For language modeling, we measure perplexity on the WikiText[35] test set and the Penn Treebank (PTB)[36] validation set. In terms of zero-shot tasks, we employ the Accuracy metric for zero-shot evaluations on seven benchmarks, including BoolQ[37], RTE[38], HellaSwag[39], WinoGrande[40], ARC-Easy and ARC-Challenge[41] and OpenbookQA[42]. For calibration data, we follow SparseGPT[15], Wanda[16], and OWL[17], which consists of 128 2048-token segments, randomly chosen from the first shard of the C4[43] dataset. For computing the KL divergence, we selected 5 samples from the 128 segments for assessment. This signifies that we utilize a generic internet dataset for pruning without task-specific data, ensuring that the downstream task is

Model	Method	Sparsity	BoolQ	RTE	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Mean
OPT-6.7B	Dense	0%	66.14	55.23	50.50	65.19	65.61	30.54	27.60	51.54
	Wanda	50%	64.58	**53.06**	46.77	62.11	63.67	27.98	**25.20**	49.05
	OWL	50%	63.79	52.35	46.35	61.48	62.96	27.39	23.80	48.30
	Ours	50%	**65.50**	**53.06**	**48.04**	**62.98**	**64.18**	**28.41**	25.00	**49.60**
	Wanda	60%	62.35	**52.70**	43.17	60.22	59.13	**25.85**	23.00	46.63
	OWL	60%	62.72	**52.70**	42.15	59.74	58.37	25.00	22.60	46.18
	Ours	60%	**62.93**	**52.70**	**44.20**	**60.85**	**60.60**	24.48	**23.20**	**46.99**
	Wanda	70%	43.42	50.90	27.56	48.22	36.23	19.96	12.60	34.12
	OWL	70%	62.04	**53.07**	31.43	52.17	45.16	17.58	16.40	39.69
	Ours	70%	**62.17**	52.34	**36.04**	**55.88**	**51.09**	**22.95**	**17.90**	**42.52**
	Wanda	80%	38.47	52.70	26.14	47.98	25.58	**22.09**	13.00	32.28
	OWL	80%	37.83	52.70	25.86	50.67	26.01	21.16	**15.00**	32.74
	Ours	80%	**49.87**	**53.42**	**27.59**	**50.98**	**36.19**	19.36	12.80	**35.74**
LLaMAv2-13B	Dense	0%	80.55	65.34	60.04	72.22	79.38	48.46	35.20	63.02
	Wanda	50%	**81.13**	60.65	57.12	**71.58**	76.43	42.32	31.20	60.06
	OWL	50%	80.67	**64.62**	57.48	71.11	76.39	42.66	31.80	60.68
	Ours	50%	80.18	63.18	**57.62**	70.71	**77.69**	**44.37**	**33.00**	**60.96**
	Wanda	60%	77.77	60.29	49.68	68.19	69.90	38.31	28.80	56.13
	OWL	60%	**79.72**	59.93	51.98	69.53	71.42	39.08	**29.40**	57.29
	Ours	60%	77.83	**63.18**	**52.70**	**69.69**	**71.59**	**39.59**	27.60	**57.45**
	Wanda	70%	62.23	52.71	31.50	55.80	48.02	22.18	18.40	41.55
	OWL	70%	65.04	52.70	38.48	61.09	57.49	26.79	21.80	46.20
	Ours	70%	**67.92**	**53.79**	**42.19**	**65.90**	**59.30**	**31.83**	**23.40**	**49.19**
	Wanda	80%	37.82	52.35	26.43	50.67	27.23	**20.22**	11.40	32.30
	OWL	80%	39.20	**52.70**	27.09	47.82	27.31	19.88	11.20	32.17
	Ours	80%	**59.78**	**52.70**	**27.71**	**51.70**	**31.06**	18.86	**11.80**	**36.23**

Table 2: Accuracies (%) for 7 zero-shot tasks with 50%-80% sparsity using OPT-6.7B and LLaMAv2-13B. Our approach and OWL both employ Wanda as the method for intra-layer weight selection. The best performance method is indicated in bold.

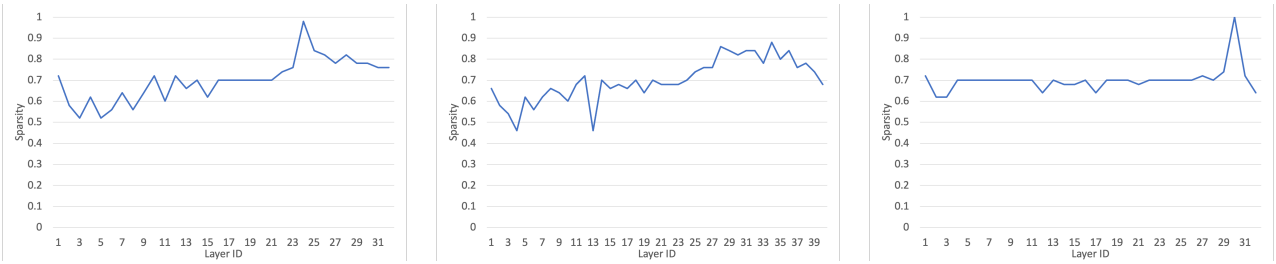
configured as a zero-shot task. For a fair comparison, the same calibration data is employed across all evaluated methods.

Competitors We compare our method with three established approaches. Specifically, we use Wanda[16] and SparseGPT[15] as benchmarks for weight selection due to their status as state-of-the-art methods. Wanda utilizes the product of weights and input activations to identify which weights should be pruned, while SparseGPT employs a computation of the Hessian matrix to inform weight selection and updating. Both Wanda and SparseGPT are uniform pruning methods that estimate the intra-layer weight importance without layer-wise importance. In contrast, the OWL[17] method has recently demonstrated notable success with non-uniform pruning by determining layer-wise sparsity through outlier analysis, which can be incorporated into Wanda and SparseGPT. To align with the configurations used in the LLaMA and OPT models reported in the OWL[17] study, we set M=5 and λ = 8% for LLaMAv2-7B and OPT1.3B, M=7 and λ = 8% for LLaMAv2-13B, M=10 and λ = 8% for OPT6.7B. For our method, under unstructured pruning, we set the adjustment step to 2%. For N:M sparsity, we configure the adjustment step to be 1:8.

Sparsity For all methods, we follow the same pruning settings as SparseGPT[15], Wanda[16], and OWL[17], which is to prune all linear layers without the first embedding layer and the final classification head. Our primary approach to induce sparsity is through unstructured pruning. Considering the potential need for practical speedup, structured N: M sparsity can achieve 1.2x-1.6x[16] speedup compared to the dense model. We also conduct evaluations on structured N: M sparsity[44]. Specifically, we provide comparisons on 4:8, 3:8, and 2:8 sparsity patterns.

4.1 Language Modeling

4.1.1 Unstructured Sparsity. Table 1 lists the performance comparisons on OPT families and LLaMAv2 families with Wanda, Figure 2 illustrates the perplexity results of combining the weights selection of Wanda and SparseGPT using different methods at varying sparsity levels of LLaMAv2-7B. Lower perplexity values indicate better model performance. Several observations can be made: Firstly, our method consistently demonstrates a significant improvement over all competing approaches across various sparsity. From Table 1, when contrasted with Wanda, our method significantly lowers the perplexity for OPT Families by a range of 1.17 to 6964.61, depending on the sparsity degree. Similarly, for the LLaMAv2 family, the PPL reduction by our method spans from 0.06 to 5986.06 when compared to Wanda. Even against the state-of-the-art non-uniform pruning method, OWL, our method stands out by decreasing the PPL for OPT-1.3B by 3.37 to 1167.47 at different sparsity degree. Secondly, the results suggest that higher sparsity levels necessitate a more thoughtful allocation of sparsity to maintain effective weights within the model. As an example, at a sparsity of 70% for



(a) LLaMAv2-7B (b) LLaMAv2-13B (c) OPT-6.7B

Figure 3: The sparsity distribution of the OPT-6.7B and the LLaMAv2-7B/13B model that we have discovered. A higher sparsity indicates that fewer parameters are retained.

the LLaMAv2-7B model, our method notably reduces perplexity by 95.2 when compared to Wanda and by 9.87 when compared to OWL. At a sparsity of 80%, our method further extends its lead, offering even 50% reductions in perplexity. Lastly, from Figure 2, our method's strong performance is evident across Wanda and SparseGPT, demonstrating the robustness and effectiveness of our approach. In summary, the experimental results suggest that our pruning method might offer a more effective balance between model size reduction and performance retention compared to its counterparts.

4.1.2 Structured N: M Sparsity. Unstructured pruning typically requires acceleration from GPU kernels, so we also conducted experiments on structured N: M methods. It is important to highlight that structured N: M methods tend to be more restrictive than unstructured approaches, typically resulting in inferior performance when comparing the same number of parameters. For example, 50% unstructured sparsity and 4:8 structured sparsity have the same amount of parameters, but 50% unstructured sparsity achieves a lower perplexity. We conducted experiments on the LLaMAv2-7B model with sparsity of 4:8, 3:8, and 2:8, respectively. In alignment with OWL's methodology, we adopted a mixed N:8 sparsity configuration. Rather than applying a uniform N value across all layers, we permitted distinct N values for individual transformer layers, ensuring an equivalent total parameter count to enable a fair comparison. The results of our experiments are presented in Table 3. From this result, our proposed method continues to outperform across various N: M values. This was particularly evident at a sparsity level of 2:8, where our method achieved a significant reduction in Perplexity compared to OWL, with a decrease of 267.69 on the WikiText dataset and a decrease of 745.09 on the PTB dataset. Such results indicate that our method can be generalized to structured N: M sparsity pruning.

Method	Sparsity	WikeText	PTB
Wanda	2:8	9367.32	4640.05
OWL	Mixed 2:8	365.00	1443.21
Ours	Mixed 2:8	**97.31**	**698.12**
Wanda	3:8	50.91	345.89
OWL	Mixed 3:8	20.60	173.89
Ours	Mixed 3:8	**16.61**	**150.45**
Wanda	4:8	8.52	44.57
OWL	Mixed 4:8	8.51	**35.05**
Ours	Mixed 4:8	**8.14**	38.12

Table 3: The perplexity of WikiText and PTB at Structured N: M Sparsity of LLaMAv2-7B. Our approach and OWL employ Wanda as the method for intra-layer weight selection.

4.2 Zero-shot Tasks

Although perplexity is a widely used metric in language modeling, it mainly serves as a statistical indicator to measure the credibility of language modeling. To provide a more comprehensive evaluation of the language model's ability in downstream tasks, we conducted experiments on the seven zero-shot task performances of all methods under different sparsity levels in OPT-6.7B and LLaMAv2-13B. The result is shown in Table 2. From the result, our method can improve accuracy under different sparsity and tasks. In OPT6.7B, our method achieved an average accuracy improvement of 0.5 to 8.4 percentage points compared to Wanda. Compared to the OWL method, it improved accuracy by 0.8 to 3 percentage points. For LLaMAv2-13B, our method stands out by improving the average accuracy by 0.9 to 7.64 compared to Wanda and achieving an average accuracy improvement of 0.28 to 4.06 percentage points compared to Wanda. These findings underscore the potential of our method for enhancing performance, even on more challenging zero-shot downstream tasks.

4.3 Discussion

Sparsity Distribution In this section, we will demonstrate and discuss the sparsity distribution of the OPT series model and the LLama series model that we have discovered. This distribution also represents the importance of each layer of transformer in the LLMs. We selected the sparsity distribution of LLaMAv2-13B, LLaMAv2-7B, and OPT-6.7B at 70% sparsity, respectively. A higher level of sparsity indicates that fewer parameters are retained. The result is shown in Figure 3. Analyzing these results, we can derive two key findings. Firstly, we found interestingly that the method tends to retain the parameters of lower-layer transformers and discard higher-layer transformers' parameters while retaining more parameters in the highest layers. This phenomenon is consistent between the OPT family model and the LLaMAv2 family model. We speculate that

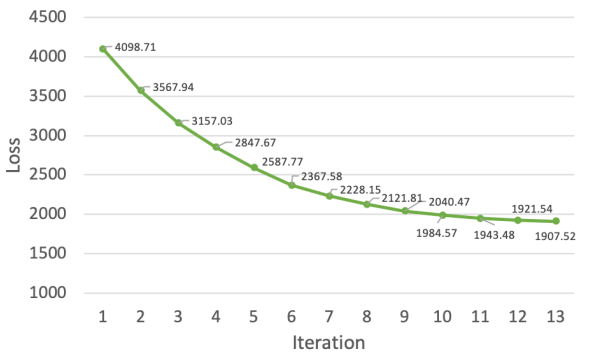


Figure 4: The decline of the total loss following the adjustment of the sparsity of LLaMAv2-7B at 70% sparsity.

the reason for this phenomenon is that the lower-layer transformer is crucial in capturing fine-grained information in the text. The higher-layer model often captures coarse-grained information and may contain more redundant parameters. Secondly, certain layers exhibit very strong pruning resistance, indicating that the model can still work even when most of their weights are pruned, such as what we have observed with the 22nd layer in LLaMAv2-7B and the 33rd layer in LLaMAv2-13B. We hypothesize that, owing to the residual connections, the model is capable of retaining functionality even when some layers are discarded under high sparsity conditions.

Loss Decrease Within our algorithm, it is necessary to evaluate the overall KL divergence of the model. Figure 4 illustrates the decline of the total loss following the adjustment of the sparsity of LLaMAv2-7B at 70% sparsity. The results indicate that as the number of iterations increases, the loss consistently diminishes, exhibiting a trend of rapid decline that gradually slows before reaching a gentle convergence. This demonstrates that our method can effectively reduce the error and enhance the performance of the model.

4.4 Time Efficiency

In this section, we investigate the convergence rate of the LLaMA2-7B model at 70% sparsity under varying step sizes, employing a V100 GPU as the computational environment. The result is shown in Table 4. Observations derived from our experimental outcomes indicate that there exists an inverse relationship between the step size and the convergence rate of the model. For instance, when the step size is set to 20%, the LLaMAv2-7B model approximately requires 54 minutes to converge. In contrast, with a reduced step size of 2%, the model necessitates an extended duration of 280 minutes to reach convergence. Furthermore, it is imperative to underscore that our methodology integrates global performance metrics which necessitate numerous forward inferences to accurately identify the pivotal sparsity distribution. We contend that the computational expenditure for forward inferences within the pruning phase represents a mere fraction of the aggregate operational duration subsequent to deployment. We assert that the additional investment of time and resources during the pruning stage is justified by the significant improvements in the model's performance.

Step	20%	10%	8%	4%	2%
Time	54	90	130	190	280
PPL	28.01	23.12	21.58	22.67	20.27

Table 4: The perplexity and time cost (minutes) of WikiText of LLaMAv2-7B at 70% sparsity.

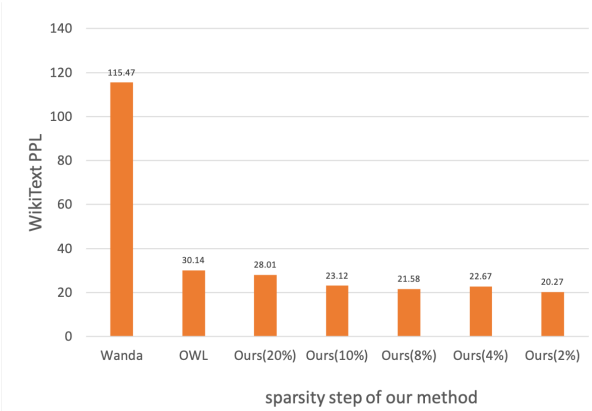


Figure 5: The effects of the hyperparameter 'step' of LLaMAv2-7B at 70% sparsity, which proves the robustness of our method

4.5 Hyperparameter Sensitivity

In our algorithm, the hyperparameter 'step' defines the magnitude of sparsity adjustments made during each iteration. A smaller step size allows the algorithm to fine-tune the sparsity distribution more precisely, but this results in slower convergence. Conversely, a larger step size leads to a coarser sparsity distribution and faster convergence. As illustrated in Figure 5, we demonstrate the varying effects of step sizes on the performance of LLaMAv2-7B at 70% sparsity. Overall, our algorithm achieves a sparsity distribution that surpasses those obtained by both Wanda and OWL, thereby underscoring the robustness of our approach.

5 CONCLUSION

In this paper, we propose a dual-assessment strategy that employs both intra-layer metric and global performance metric to comprehensively evaluate the impact of pruning. And we designed an iterative optimization algorithm to gauge each layer's global importance precisely. We conduct a thorough evaluation of our method on the LLaMAv2 family and OPT family on various language benchmarks and many zero-shot tasks. Our innovative approach significantly outperforms the established baseline and stands competitive against the state-of-the-art method. At the same time, this paper focuses on exploring the roles played by different transformer layers, and there are still many questions that can be explored, such as the importance of attention mechanisms in different layers. We believe that conducting these explorations can not only contribute to model compression but also help us better understand the operational mechanisms of LLMs.

REFERENCES

[1] Radford, Alec, and et al. Improving language understanding by generative pre-training. *OpenAI blog*, June 2018.

[2] Radford, Alec, and et al. Improving language understanding by generative pre-training. *OpenAI blog*, 1(8), Feb 2019.

[3] Brown, Tom, and et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33, July 2020.

[4] OpenAI. Training language models to follow instructions with human feedback. *arXiv preprint*, March 2022.

[5] Frantar Elias and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. *Advances in Neural Information Processing Systems*, 35, 2022.

[6] Dettmers, Tim, and et al. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint*, 2022.

[7] Xiao, Guangxuan, and et al. Smoothquant: Accurate and efficient post-training quantization for large language models. *International Conference on Machine Learning*, 2023.

[8] Frantar, Elias, and et al. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint*, March 2023.

[9] Dettmers, Tim, and et al. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint*, 2023.

[10] Chee, Jerry, and et al. Quip: 2-bit quantization of large language models with guarantees. *arXiv preprint*, March 2024.

[11] LeCun, Yann, and et al. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4), 1989.

[12] LeCun, Yann, and et al. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

[13] Hassibi, Stork, and et al. Optimal brain surgeon and general network pruning. *IEEE international conference on neural networks*, 1993.

[14] Han, Song, and et al. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

[15] Frantar Elias and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. *International Conference on Machine Learning*, 2023.

[16] Sun, Mingjie, and et al. A simple and effective pruning approach for large language models. *arXiv preprint*, 2023.

[17] Lu Yin, You Wu, and et al. Outlier weighed layerwise sparsity (owl): A missing secret sauce for pruning llms to high sparsity. *arXiv preprint*, 2023.

[18] Michael C Mozer and Paul Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. *Advances in neural information processing systems*, 1, 1988.

[19] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

[20] Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1):2383, 2018.

[21] Wei Wen, Yuxiong He, Samyam Rajbhandari, Minjia Zhang, Wenhan Wang, Fang Liu, Bin Hu, Yiran Chen, and Hai Li. Learning intrinsic sparse structures within long short-term memory. *arXiv preprint arXiv:1709.05027*, 2017.

[22] Shaohui Lin, Rongrong Ji, Chenqian Yan, Baochang Zhang, Liujuan Cao, Qixiang Ye, Feiyue Huang, and David Doermann. Towards optimal structured cnn pruning via generative adversarial learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2790–2799, 2019.

[23] Xinyin Ma, Gongfan Fang, and et al. Llm-pruner: On the structural pruning of large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[24] Mingyang Zhang, Hao Chen, and et al. Loraprune: Pruning meets low-rank parameter-efficient fine-tuning. *arXiv preprint*, 2023.

[25] Tianyi Chen, Tianyu Ding, and et al. Lorashear: Efficient large language model structured pruning and knowledge recovery. *arXiv preprint*, 2023.

[26] Decebal Constantin Mocanu, Elena Mocanu, and et al. A topological insight into restricted boltzmann machines. *Machine Learning*, 104:243–270, 2016.

[27] Utku Evci, Trevor Gale, and et al. Rigging the lottery: Making all tickets winners. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 2943–2952. PMLR, 13–18 Jul 2020.

[28] Shiwei Liu, Tianlong Chen, and et al. The unreasonable effectiveness of random pruning: Return of the most naive baseline for sparse training. In *International Conference on Learning Representations*, 2022.

[29] Jonathan Frankle, Michael Carbin, and et al. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

[30] Jaeho Lee, Sejun Park, and et al. Layer-adaptive sparsity for the magnitude-based pruning. In *International Conference on Learning Representations*, 2021.

[31] Shiwei Liu, Tianlong Chen, and et al. Sparse training via boosting pruning plasticity with neuroregeneration, 2022.

[32] Ajay Jaiswal, Zhe Gan, Xianzhi Du, Bowen Zhang, Zhangyang Wang, and Yinfei Yang. Compressing llms: The truth is rarely pure and never simple, 2023.

[33] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. Opt: Open pre-trained transformer language models, 2022.

[34] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[35] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

[36] Mitchell P. Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. The penn treebank: Annotating predicate argument structure. In *Human Language Technology, Proceedings of a Workshop held at Plainsboro, New Jerey, USA, March 8-11, 1994*. Morgan Kaufmann, 1994.

[37] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions, 2019.

[38] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding, 2019.

[39] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019.

[40] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019.

[41] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018.

[42] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering, 2018.

[43] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023.

[44] Itay Hubara, Brian Chmiel, Moshe Island, Ron Banner, Seffi Naor, and Daniel Soudry. Accelerated sparse neural training: A provable and efficient method to find n:m transposable masks, 2021.