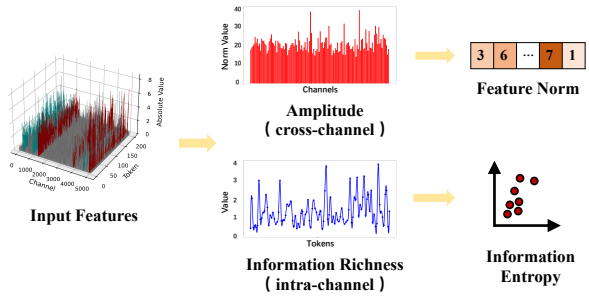
E-Sparse: Boosting the Large Language Model Inference through Entropy-based N:M Sparsity

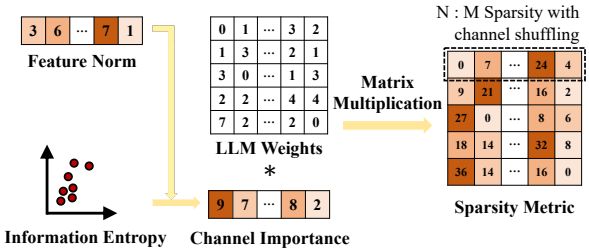
Anonymous ACL submission

Abstract

Traditional pruning methods are known to be challenging to work in Large Language Models for Generative AI because of their unaffordable training process and large computational demands. For the first time, we introduce the information entropy of hidden state features into a pruning metric design, namely `E-Sparse`, to improve the accuracy of N:M sparsity on LLMs. `E-Sparse` employs the information richness to leverage the channel importance, and further incorporates several novel techniques to put it into effect: (1) it introduces information entropy to enhance the significance of parameter weights and input feature norms as a novel pruning metric, and performs N:M sparsity without modifying the remaining weights. (2) it designs global naive shuffle and local block shuffle to quickly optimize the information distribution and adequately cope with the impact of N:M sparsity on LLMs' accuracy. `E-Sparse` is implemented as a Sparse-GEMM on FasterTransformer and runs on NVIDIA Ampere GPUs. Extensive experiments on the LLaMA family and OPT models show that `E-Sparse` can significantly speed up the model inference over the dense model (up to 1.53×) and obtain significant memory saving (up to 43.52%), with acceptable accuracy loss.



(a) Evaluate the input features from both cross-channel and intra-channel dimensions.



(b) The entropy-based sparsity metric of `E-Sparse`.

Figure 1: Overview of the proposed `E-Sparse`. It first introduces entropy to quantify the information richness within each channel (intra-channel) of the input features, and adopts it to enhance the feature norms (cross-channel) as a metric to evaluate parameter importance. Furthermore, it proposes Channel Shuffle to reorder the information distribution in LLMs to obtain N:M Sparsity with less information loss.

1 Introduction

Large language models (LLMs), such as GPT-3(Brown et al., 2020), LLaMA(Touvron et al., 2023), Bloom(Scao et al., 2022), and others, have recently exhibited outstanding performance across a wide range of tasks, including but not limited to social systems, intelligent conversation, content generation, code creation, etc. However, deploying LLMs poses significant challenges due to their substantial computational demands and high memory requirements. For instance, the most powerful variant, the Bloom model with 176 billion parameters, necessitates a minimum of 350 GB of storage in half-precision (FP16) format. When configured with a batch size of 1 and a sequence length of 128, Bloom-176B inference demands a formidable ensemble of 16 NVIDIA A10 GPUs, each equipped with 24GB memory. Consequently, optimizing these models through compression and pruning has emerged as a critical strategy to reduce parameter counts, thereby decreasing computational overhead and conserving memory resources.

In order to harness the acceleration and memory reduction potential offered by sparse neural networks, GPU manufacturers have introduced

architectural enhancements. Specifically, the invention of Sparse Tensor Core (a10, 2020; h10, 2023; Yao et al., 2019; Cao et al., 2019) technology has been pivotal in capitalizing on weight sparsity within Deep Neural Network (DNN) models. This innovation employs a fine-grained structural pruning technique, involving a 2-out-of-4 pruning approach within each partitioned sub-vector. This method effectively balances the computational workload while maximizing parallelism within the dot-product unit.

While there has been substantial research on compressing LLMs using low-precision quantization(Xiao et al., 2023; Dettmers et al., 2022; Frantar et al., 2022), relatively little effort has been dedicated to fully exploiting Sparse Tensor Core technology for accelerating LLMs. Some prior work, exemplified by Wanda(Sun et al., 2023), has proposed the application of a 2-out-of-4 pruning pattern for LLMs. This approach determines channel importance by evaluating input feature norms and weights them against standard parameter magnitudes as pruning metrics. In this studyhe, we introduce an Entropy-based pruning algorithm that builds upon these principles. Our research showcases a remarkable **1.32** LLaMA perplexity improvement over state-of-the-art techniques and delivers a **19.6%-34.8%** speedup on an A100 GPU, demonstrating the effective and efficient utilization of Sparse Tensor Core hardware.

Our work is grounded in two crucial observations. Firstly, we note that **the richness of information among channels exhibits significant variation**. Even within the same batch of tokens, the entropy of elements within each channel differs considerably, despite some sharing the same input feature norm. Secondly, we observe that **channels with close entropy values tend to exhibit relatively concentrated distributions**. These observations naturally inspire us to leverage channel-specific information in order to enhance LLMs inference using $N : M$ sparsity.

Our proposal We propose entropy-based sparsity (`E-Sparse`), a novel method to prune LLMs without modifying the remaining weights. Figure 1 shows the key idea of one-shot `E-Sparse`.

Firstly, inspired by Observation 1, we introduce a novel metric to assess the importance of weights. This metric employs information entropy to quantify the amount of information within each channel of the hidden state features in LLMs. We enhance the significance of parameter weights and input feature norms by incorporating information entropy as a metric for evaluating parameter importance.

Secondly, we implement a channel shuffling mechanism to ensure a more equitable distribution of information among the channels in the hidden features (Figure 3). As Observation 2 reveals, the information distribution across channels tends to be highly concentrated, which can impede the accuracy of $N : M$ sparsity due to the need to remove N elements from adjacent M elements. Channel shuffling is instrumental in preserving a greater number of elements within information-rich channels, thereby mitigating the impact of parameter pruning on LLMs accuracy.

Lastly, with the robust support of NVIDIA's cuSPARSE(cuS, 2023a) and cuSPARSELt(cuS, 2023b) libraries, we have crafted an efficient `E-Sparse` GEMM designed explicitly for LLMs inference and integrated it into FasterTransformer.

`E-Sparse` enables the N:M sparsity of weights for all the matrix multiplications in LLMs, including the LLaMA family, and OPT. The results show that `E-Sparse` outperforms the performance of the state-of-the-art training-free sparsity methods (Frantar and Alistarh, 2023; Sun et al., 2023) for LLMs. It has also been demonstrated that `E-Sparse` can achieve a 1.24–1.53$\times$ speedup and a 42.64%–43.52% memory saving for LLMs with negligible loss in accuracy.

2 Inspiration from Observations

It has been found that a small subset of hidden state features (named "outlier") in LLMs are exceptionally large in magnitude (Dettmers et al., 2022; Xiao et al., 2023), and these features are important for LLMs compression (Sun et al., 2023). Then, we visualize the input activations of linear layers in LLMs and find several key observations about these activations that motivate our method:

• **The information richness between channels varies greatly.** A recent work (Sun et al., 2023) found that the norm of activation in LLMs can be used to measure channel importance. In addition to the same finding, we also observed that the information entropy between channels also varies greatly. To facilitate observation, we first sort the channels according to the norm value and then compare the entropy of each channel feature according to the same index sorted by the norm in Figure 2a and Figure 2b. We find that the entropy of different channels differ considerably, despite some shar-

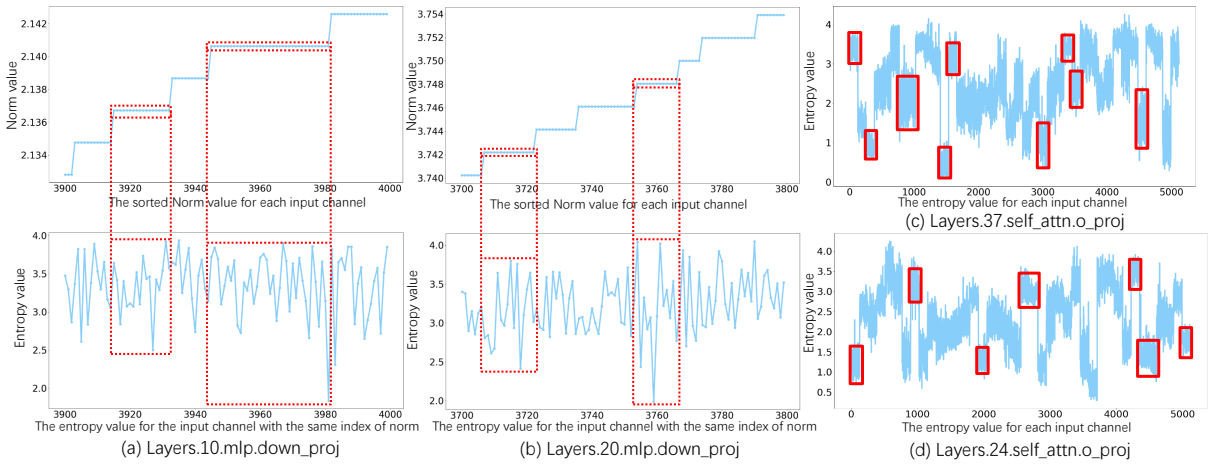


Figure 2: The visualization of the hidden activations in LLMs. The data for each subfigure comes from the activation of the corresponding layer of LLaMA-13B. For clarity, we only capture the norm and entropy values for the 100 channels after norm sorting in (a) and (b). We show the entropy values of all channels in (c) and (d).

ing the same input feature norm. The observation above motivates us to enhance evaluation metrics through information richness.

• **The entropy values of adjacent channels are relatively close.** As shown in Figure 2c and Figure 2d, channels with close entropy tend to exhibit relatively concentrated distributions. However, N:M sparsity forces the model to prune N values out of M consecutive values in the channel dimension, which makes us inevitably need to prune in M consecutive informative channels and damage the accuracy of LLMs. This observation straightforwardly motivates us to shuffle the channels to preserve a greater number of elements within information-rich channels, thereby mitigating the impact of N:M sparsity on accuracy.

3 Method

3.1 Method Overview

`E-Sparse` proposes a new entropy-based metric to evaluate the parameter importance in LLMs, and introduces channel shuffling to minimize the information loss brought by N:M sparsity. The key advantages of `E-Sparse` include: 1) Sparse the LLMs without modifying the remaining weights. In contrast to channel-by-channel parameter sparse and update (Frantar and Alistarh, 2023), `E-Sparse` augments the parameter weights with the information richness and the amplitude of the feature as an evaluation metric, and then adopts it to sparse the weights of a layer at once. 2) More fine-grained importance evaluation of hidden state channels. Apart from the global information (channel amplitude), `E-Sparse` introduces entropy to measure the local information of channels (information richness), thereby comprehensively measuring the importance of channels. 3) More flexible sparse mode. Traditional N:M sparsity forces pruning of N out of M consecutive values, `E-Sparse` introduces channel shuffle mechanism, which is more adaptable to the feature information distribution of LLMs and reduces accuracy loss.

3.2 Information Richness - Entropy

The observation in Section 2 motivates us to enhance the evaluation metrics of LLMs pruning through information richness. Entropy (Shannon, 1948) is a key indicator in the field of information theory to measure the amount of information and uncertainty. The larger the entropy, the higher the information richness. Therefore, we introduce entropy to evaluate the channel information of activation for augmenting the standard weight magnitude and channel norm as a novel pruning metric.

Let $X \in R^{o \times C}$ denote the hidden feature of a fully connected layer in LLMs, where C is the number of channels, and o is the dimension of each channel. To compute the entropy, we first divide it into K different bins and then calculate the probability of an element in the channel falling into each bin. Then, the information richness (entropy) of channel c can be formulated as:

$$\mathcal{IR}_c = -\sum_{k=1}^{K} p_k^c log\left(p_k^c\right) \quad (1)$$

in which, p_k^c is the probability of bin k in channel

c, and $IR_c \in [0, +\infty)$. We set K to 100 empirically, which can achieve good results. Information entropy can be used as a good fine-grained metric to evaluate information richness. The larger IR_c value means higher information richness.

Next, regarding coarse-grained evaluation, we follow (Sun et al., 2023) and adopt the input feature norm to measure the amplitude:

$$\mathcal{AM}_c = \|X_c\|_2 \tag{2}$$

where $\|X_c\|_2$ represents the L^2 norm of the channel X_c.

Finally, to comprehensively evaluate the importance of channels and obtain more reasonable weight evaluation metric, we integrated the fine-grained indicator and the coarse-grained indicator above to get the following evaluation metric for pruning redundant weights in LLMs:

$$\xi_{cj} = |w_{cj}| \cdot (\mathcal{IR}_c + \alpha \cdot \mathcal{AM}_c) \tag{3}$$

in which, w_{cj} is the j-th element in channel c of the fully connected layer in LLMs, and ξ_{cj} is the final important score of w_{cj} in the sparsity metric. The larger ξ_{cj} value means higher importance of the element in this layer.

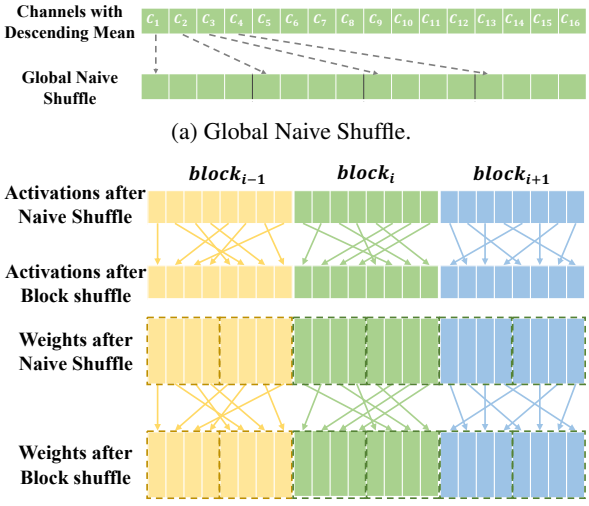
3.3 Information Reorder - Channel Shuffle

Inspired by the observation in Section 2, `E-Sparse` implements a channel shuffling mechanism to ensure a more equitable distribution of information among the channels in the hidden features. By reordering the channel index of the hidden state feature and the layer parameter, `E-Sparse` aims to make the channels with higher information richness distributed more evenly, thus minimizing the information loss caused by N:M sparsity.

First, the N:M sparsity can be formulated as a constrained optimization problem:

$$\mathcal{O} = \min_{\theta} \frac{1}{2} \left\| Y - W^{\theta}_{N:M} \cdot X \right\|_F^2 \tag{4}$$

in which, X and Y are the input and original output of a fully connected layer, respectively. θ is the index order of channels, and $W^{\theta}_{N:M}$ is the weight after performing N:M sparsity on W under the current index order. We are committed to finding an optimal channel order θ, which can minimize the output loss caused by M:N sparsity. However, directly optimizing the above problems in LLMs will bring a large computational overhead. Considering



(a) Global Naive Shuffle.

(b) Local Block Shuffle.

Figure 3: Channel Shuffle of `E-Sparse`. Take 2:4 sparsity as an example. `E-Sparse` first sorts the channels **globally** according to the channel mean of the sparsity metric, and then divides the channels with close mean into different groups, which is **coarse-grained but faster**. Then, `E-Sparse` splits the channel into multiple blocks and performs channel shuffle within the blocks, which is slightly slower than the global shuffling but more accurate.

that the importance metric in (3) contains the information from both weights and activation, we simplify the above problem to minimizing the sparse loss of ξ_{cj}:

$$\acute{\mathcal{O}} = \max_{\theta} \sum_{c=1}^{C} (\xi_{cj})^{\theta}_{N:M} \tag{5}$$

in which, $(\xi_{cj})^{\theta}_{N:M}$ is the evaluation metric after N:M sparsity under the channel permutation θ. Compared to (4), there is no need to repeatedly perform matrix multiplication to calculate the feature map Y and the sparse feature map.

Although the optimization problem above has been greatly simplified, performing channel shuffle in LLMs is non-trivial. The large channel size of LLMs results in a big search space, which in turn brings huge computational and time overhead. For a fully connected layer with C channels, there are $C!$ different orderings of channels. For instance, a layer with 1024 channels has a channel ordering of 10^{2640}. In LLMs, the maximum number of channels can reach more than 10,000, which brings huge resistance to obtaining the optimal permutation.

To deal with the issue above, `E-Sparse` introduced the channel shuffle, which consists of two steps: *global naive shuffle* and *local block shuffle*.

Global Naive Shuffle. To reduce the complexity of channel shuffle in LLMs as much as possible, E-Sparse first performs a fast global channel shuffle. For the sparsity metric $\xi \in R^{o \times C}$, the mean value of each channel is calculated, and based on which the channels are shuffled in descending order. As shown in Figure 3a, according to the sparsity pattern (M), E-Sparse shuffles the channels with close means into different sparse groups. Global naive shuffle can achieve fast coarse-grained information reordering.

Local Block Shuffle. To further minimize the information loss caused by N:M sparsity, E-Sparse introduces local block shuffle. First, E-Sparse divided the ξ after global naive shuffle into n blocks, and each block contains m channels ($C = m \cdot n$), as shown in Figure 3b. We use $m = 256$ unless otherwise specified, thus the channel search space is reduced from $C!$ to $n \cdot 256!$, making the number of unique permutations can be completed in an acceptable amount of time. Then, E-Sparse performs channel shuffling in each small block by adapting the classic greedy search algorithm (Ji et al., 2018; Pool and Yu, 2021).

Combining global naive shuffle and local block shuffle, E-Sparse can realize a fast optimization for information distribution and well cope with the challenge of large channel dimensions in LLMs.

4 Efficient Sparse-GEMM Implementation

To deploy the proposed method in actual application scenarios, we implemented E-Sparse as a sparse engine for efficient LLMs inference. We choose FasterTransformer(Fas, 2023) as the backend and implemented the sparse general matrix multiplication (Sparse-GEMM) of E-Sparse for LLMs inference. Taking 2:4 sparsity as an example, the sparse deployment of Sparse-GEMM mainly includes three steps. (1) E-Sparse first compresses the sparse weights $W_{2:4} \in R^{o \times C}$ into a compressed format, which includes the non-zero weights $W_{2:4} \in R^{o \times \frac{C}{2}}$ and the indices of these non-zero data values. (2) With the support of NVIDIA's cuSPARSE and cuSPARSELt, E-Sparse searches for the optimal matrix multiplication algorithm according to the shape of each sparse weights tensor in LLMs and saves them. (3) Integrates E-Sparse into FasterTransformer for LLMs inference. Based on the saved optimal matrix multiplication algorithm, LLMs can skip 50% of matrix multiplication operations and perform faster inference. The experiments in Section 5.4 have shown that such a design can bring **19.6%–34.8%** latency reduction and **42.64%–43.52%** memory saving.

5 Experiments

5.1 Experimental Environments

Setup. In our experimental framework, we primarily target the LLaMA model family (LLaMA-7B/13B/30B/65B) and OPT models (OPT-6.7B/30B). To demonstrate the comprehensiveness of E-Sparse, we further extends it to the OPT and BLOOM models. All models are from the HuggingFace Transformers library (Wolf et al., 2019). We choose two SOTA methods as our baselines: SparseGPT and Wanda. Following the one-shot sparsity setting of Wanda, we sample the same 128 sequences from C4 (Raffel et al., 2020) training data as calibration dataset. All our experiments only need read right on the models without modifying the remaining weights. In addition, we demonstrate the real-world inference acceleration of 4:8 and 2:4 sparsity patterns on NVIDIA Ampere Architecture (a10, 2020).

Datasets & Evaluation. As perplexity is a stable and robust metric to measure the capabilities of LLMs. Importantly, lower perplexity values indicate better model performance. We reported our results on the WikiText (Merity et al., 2016) validation dataset, based on the perplexity metric. To further demonstrate the efficiency of our method, we also present the zero-shot performance of the pruned networks. Notably, higher values are indicative of superior model performance. Our evaluation rely on the widely-acknowledged EleutherAI LM Harness benchmark (Gao et al., 2021). The zero-shot evaluation benchmark mainly includes the following datasets: HellaSwag (Zellers et al., 2019), OpenbookQA (Mihaylov et al., 2018), PiQA (Bisk et al., 2020), SciQ (Pedersen et al., 2020) and LogiQA (Liu et al., 2020).

5.2 Pruning Results on LLMs

To demonstrate the pruning performance of E-Sparse, we conduct a series of experiments to evaluate its efficacy across various model sizes within the LLaMA model family. Similar to Wanda and SparseGPT, we evaluate the perplexity of WikiText validation on structured 4:8 and 2:4 sparsity. As Table 1 shows, our E-Sparse achieves significant improvements compared with the strong

Table 1: E-Sparse's perplexity performance on LLaMA model family. The results show that E-Sparse can outperform state-of-the-art methods by a large margin without updating the remaining weights. As for the more constrained and challenging 2:4 sparsity, E-Sparse can obtain an 8.26 perplexity for LLaMA-13B, which is **1.32 better than Wanda and 0.85 better than SparseGPT**.

Methods	N:M sparsity	LLaMA-7B	LLaMA-13B	LLaMA-30B	LLaMA-65B
FP16	-	5.68	5.09	4.10	3.56
Magnitude	2:4	42.53	18.36	7.62	7.11
SparseGPT		11.00	9.11	7.16	6.28
Wanda		11.53	9.58	6.90	6.25
E-Sparse		**10.56**	**8.26**	**6.56**	**5.69**
Magnitude	4:8	16.83	13.86	9.11	6.35
SparseGPT		8.61	7.40	6.17	5.38
Wanda		8.56	7.40	5.97	5.30
E-Sparse		**8.29**	**6.92**	**5.74**	**5.09**

Table 2: Accuracy of LLaMA under 2:4 sparsity patterns on different Zero-Shot tasks. It shows that E-Sparse consistently outperforms SparseGPT and Wanda, especially in terms of overall average accuracy across five tasks.

Params	Method	HellaSwag	PiQA	OpenBookQA	SciQ	LogiQA	Avg.
7B	FP16	56.41	78.29	28.20	89.6	21.81	54.86
	Magnitude	41.98	68.00	22.00	74.00	21.00	45.60
	Sparse GPT	42.95	70.78	19.80	**85.00**	**23.34**	48.37
	Wanda	41.82	70.13	21.60	83.90	21.96	47.68
	E-Sparse	**43.59**	**72.03**	**23.00**	84.10	22.27	**49.00**
13B	FP16	59.08	78.89	30.60	93.40	26.57	59.77
	Magnitude	45.06	71.27	23.20	82.80	25.80	57.71
	Sparse GPT	47.34	74.48	24.00	**88.00**	21.35	51.03
	Wanda	45.99	73.55	**25.40**	87.90	**23.04**	51.16
	E-Sparse	**49.40**	**75.24**	24.80	87.80	19.81	**51.41**
30B	FP16	62.64	81.55	29.06	92.50	28.41	58.83
	Magnitude	51.10	77.36	24.40	90.10	22.42	53.08
	Sparse GPT	52.60	**78.40**	28.20	93.30	25.96	55.69
	Wanda	53.74	77.96	27.40	92.90	27.80	56.00
	E-Sparse	**56.41**	77.36	**28.80**	**93.80**	**29.03**	**57.08**
65B	FP16	62.64	81.55	29.60	92.50	28.41	58.94
	Magnitude	57.07	77.36	30.00	90.10	23.65	55.64
	Sparse GPT	55.23	78.40	27.60	93.30	24.42	55.79
	Wanda	55.76	77.96	29.00	92.90	**26.72**	56.47
	E-Sparse	**58.46**	**78.56**	**31.60**	**93.80**	23.04	**57.09**

Table 3: E-Sparse's perplexity performance on OPT models. The results reveal that E-Sparse achieves higher performance than Magnitude and Wanda on both 2:4 and 4:8 patterns, which demonstrates the good generalization of E-Sparse.

Methods	OPT-6.7b(2:4)	OPT-30b(2:4)	OPT-6.7b(4:8)	OPT-30b(4:8)
FP16	10.86	9.56	10.86	9.56
Magnitude	264.14	1980.71	196.18	563.72
Wanda	15.89	13.42	13.56	10.87
E-Sparse	**14.90**	**12.35**	**13.12**	**10.75**

baselines. It is noteworthy that E-Sparse does not require weight updates, yet it outperforms the reconstruction-based SparseGPT across all variants within the LLaMA model family. At the largest LLaMA-65B, the performance of E-Sparse is close to the FP16 baseline. For instance, 4:8 sparsity achieves a perplexity loss of only 1.53 more than FP16. The results indicate that our entropy-based metric and channel shuffle mechanism plays a critical role in N:M sparsity.

To assess the generalization of our method, we conduct experiments on OPT model family, which is one of the most representative LLMs prior to the release of the LLaMA. We choose two models of varying sizes, specifically the OPT-6.7B and OPT-30B, for our experiments. According to the result in Table 3, it is evident that the implementation of E-Sparse can lead to a substantial enhancement in WikiText validation. For instance, E-Sparse can achieve a perplexity score of 14.9 at 2:4 sparsity, markedly outperforming Wanda baseline, which registers at 15.89.

To provide further evidence of our method's performance, we also present results on several Ze-

Table 4: Ablation study on the pruning metric and channel shuffle. Let $Norm$ denote the input feature norm (baseline). $Entropy$ indicates the information entropy. GNS means the Global Naive Shuffle, and LBS is the Local Block Shuffle. The results show that both the proposed entropy strategy and two shuffling methods can bring noteworthy performance gains.

Techniques				LLaMA-7B	LLaMA-13B	LLaMA-30B	LLaMA-65B
$Norm$	$Entropy$	GNS	LBS				
✓	✗	✗	✗	11.53	9.58	6.90	6.25
✓	✓	✗	✗	11.42	8.82	6.80	6.05
✓	✓	✓	✗	10.98	8.58	6.62	5.78
✓	✓	✓	✓	**10.56**	**8.26**	**6.56**	**5.69**

Table 5: GEMM Speedup of E-Sparse after 2:4 sparsity on LLMs. The inputs and weights are all in half-precision (FP16) format, and the latency is evaluated on a single NVIDIA A100 40GB GPU.

	Layer	Input	Weights	Dense GEMM	E-Sparse GEMM	Latency Reduction
Context-stage	Q/K/V	16384×14336	14336×5376	8.452ms	5.815ms	**31.2%**
	Att_Out	16384×1792	1792×14336	3.488ms	2.540ms	**27.2%**
	FFN-1	16384×14336	14336×7168	11.487ms	8.073ms	**29.7%**
	FFN-2	16384×7168	7168×14336	11.478ms	8.958ms	**21.9%**
Decoder	Q/K/V	16×14336	14336×5376	0.122ms	0.098ms	**19.6%**
	Att_Out	16×1792	1792×14336	0.046ms	0.030ms	**34.8%**
	FFN-1	16×14336	14336×7168	0.160ms	0.112ms	**30.0%**
	FFN-2	16×7168	7168×14336	0.158ms	0.109ms	**31.0%**

Table 6: Memory saving of E-Sparse on LLaMA family.

Models	Dense (FP16)	Sparse (FP16)	Memory Saving
LLaMA-7B	9.85GB	5.65GB	**42.64%**
LLaMA-13B	19.11GB	10.89GB	**43.01%**
LLaMA-30B	47.99GB	27.17GB	**43.38%**
LLaMA-65B	96.50GB	54.50GB	**43.52%**

roShot tasks for LLaMA under 2:4 sparsity. The comprehensive results have been tabulated in Tab 2. It can be observed that our E-Sparse consistently exhibits an edge, particularly evident from the superior average accuracy metrics amassed across the quintet of Zero-Shot tasks when compared with other established baseline methods. E-Sparse outperforms Wanda by a margin of 3% and exceeds SparseGPT by 1% on average accuracy for LLaMA-7B. Despite the 2:4 pruning being the most constrained sparsity pattern, our method achieves enhanced performance for all model size on HellaSwag. Additionally, our approach either matches or surpasses the performance of Wanda and SparseGPT on the other four datasets.

5.3 Ablation Study

The good performance of E-Sparse is mainly attributed to the proposed entropy-based pruning metric and two channel shuffle strategies. To validate the effectiveness of these strategies, we conduct a series of ablation studies on LLaMA models in 2:4 sparse pattern. We take the input feature norm ($Norm$ (Sun et al., 2023)) as the baseline strategy.

The results are shown in Table 4. Firstly, it shows that simply introducing *Entropy* to build the pruning metric can bring up to 0.76 perplexity improvement, demonstrating the effectiveness of information entropy on LLM pruning. Then, the introduction of the global naive shuffle and the local block shuffle successively brought the perplexity gains of up to 0.44 and 0.42 respectively, which reveals that GNS and LBS are two complementary channel shuffle strategies. The results above prove that the three proposed new techniques are efficient and effective.

5.4 Speedup and Memory Saving

In this section, we show the measured speedup and memory saving of E-Sparse integrated into FasterTransformer.

Speedup. With the E-Sparse integrated into FasterTransformer, we measure the latency of GEMM in the Context-stage and the Decoder for a batch of 4 and a sequence length of 1024. Due to the lack of support for 4:8 sparsity pattern in NVIDIA Ampere architecture, we only measure the latency of GEMM with 2:4 sparsity on a single A100 40GB GPU. As shown in Table 5, E-Sparse is consistently faster than the dense FP16 GEMM baseline, delivering up to 34.8% latency reduction.

It shows that E-Sparse can work well on both the context-stage and the decoder in LLMs.

Memory Saving. In Table 6, we give the memory saving brought by E-Sparse on LLaMA family. The results reveal that it can save 42.64%–43.52% memory usage on LLaMA models. We can also see a trend that the larger the model, the more significant the memory saving.

6 Related Work

Traditional Network Pruning. Network pruning was proposed to remove redundant parts of the DNN models, thereby reducing the computational and memory demands of neural networks without accuracy loss (Liu et al., 2018; Louizos et al., 2017; Han et al., 2016; Hassibi et al., 1993). Traditional network pruning techniques usually fall into two primary categories: unstructured pruning (Hassibi et al., 1993; Han et al., 2015, 2016) and structured pruning (Li et al., 2017; Luo et al., 2017; Liu et al., 2017; Li et al., 2020, 2022; Ding et al., 2021; Li et al., 2021; Xia et al., 2022). Unstructured pruning methods (Han et al., 2016, 2015) aim to iteratively prune unimportant connections whose absolute weights are smaller than a given threshold, which achieves good performance on parameter compression. However, such kind of methods are implementation-unfriendly. Structured pruning methods (Li et al., 2017; Luo et al., 2017; Liu et al., 2019) prune or sparse entire parts of the network (e.g., channels, blocks) instead of individual weights, thus require less specialized libraries to achieve inference speedup. A common feature of the traditional pruning techniques mentioned above is that the pruned network usually needs to be retrained to recover the accuracy loss, which hinders their application on LLMs that consume huge training resources.

N:M Sparsity. N:M sparsity (Mishra et al., 2021; Pool and Yu, 2021; Akiva-Hochman et al., 2022; Zhou et al., 2021) is a kind of special pruning technique that introduces an intermediate sparsity pattern between unstructured and structured pruning, called semi-structured sparsity. N:M sparsity aims to prune N out of every M consecutive parameters, rather than pruning individual weights or entire channels/blocks. The appeal of N:M sparsity is its ability to reason for specific hardware architectures (such as NVIDIA Ampere(Pool, 2020)), enabling efficient computation. (Akiva-Hochman et al., 2022) suggests a Neural Architecture Search (NAS) strategy to sparse both activations and weights throughout the network. (Zhou et al., 2021) defines a metric, Sparse Architecture Divergence (SAD) to learn N:M sparse neural networks. However, these are only designed for CNNs or small models, and how to design efficient N:M sparsity for LLMs has been rarely studied.

Pruning for LLMs. Due to the massive size and computational costs of large language models, training-based pruning methods (Ma et al., 2023; Xia et al., 2023; Singh and Bhatele, 2023) will bring a large overhead. So existing popular solutions aim at post-training pruning strategy(Frantar and Alistarh, 2023; Sun et al., 2023). Such methods only need a small number of calibration data to prune the pre-trained LLMs models, which is suitable for rapid deployment. SparseGPT(Frantar and Alistarh, 2023) develops a layer-wise weight update for LLMs via an approximate second-order Hessian. This schema is iteratively executed between weight pruning and weight update at each layer, which is computationally expensive. Wanda(Sun et al., 2023) presents to remove the insignificant weights based on the magnitude and norm of corresponding input activations, without updating the remaining weights. Our work further proposes a new metric based on the information richness and designs an effective search strategy for N:M sparsity.

7 Conclusion

In this paper, we propose a novel entropy-based pruning method, called E-Sparse, to carry out N:M sparsity on LLMs in a one-shot manner. The design of our pruning metric is based on the observation of the information richness of hidden state channels and relatively concentrated distributions of information-rich channels. Extensive experiments show the superior performance of our proposal against existing LLMs pruning methods.

8 Limitations

Beyond NLP tasks, the applicability of E-Sparse to other tasks (including computer vision or speech recognition), remains to be tested. For fair comparison with other methods, we only conducted experiments on public datasets with limited sentence lengths. In addition, the combined optimization of E-Sparse and other orthogonal methods (quantization or distillation) has not yet been studied.

References

2020. NVIDIA A100 Tensor Core GPU Architecture. https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf.

2023a. cuSPARSE. https://docs.nvidia.com/cuda/cusparselt/index.html.

2023b. cuSPARSELt. https://docs.nvidia.com/cuda/cusparselt/index.html.

2023. FasterTransformer. https://github.com/NVIDIA/FasterTransformer/tree/release/v5.3_tag.

2023. NVIDIA H100 Tensor Core GPU Architecture. https://resources.nvidia.com/en-us-tensor-core.

Ruth Akiva-Hochman, Shahaf E Finder, Javier S Turek, and Eran Treister. 2022. Searching for n: M fine-grained sparsity of weights and activations in neural networks. In *European Conference on Computer Vision*, pages 130–143. Springer.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Shijie Cao, Chen Zhang, Zhuliang Yao, Wencong Xiao, Lanshun Nie, Dechen Zhan, Yunxin Liu, Ming Wu, and Lintao Zhang. 2019. Efficient and effective sparse lstm on fpga with bank-balanced sparsity. In *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 63–72.

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

Xiaohan Ding, Tianxiang Hao, Jianchao Tan, Ji Liu, Jungong Han, Yuchen Guo, and Guiguang Ding. 2021. Resrep: Lossless cnn pruning via decoupling remembering and forgetting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4510–4520.

Elias Frantar and Dan Alistarh. 2023. Massive language models can be accurately pruned in one-shot. *arXiv preprint arXiv:2301.00774*.

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

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, et al. 2021. A framework for few-shot language model evaluation. *Version v0. 0.1. Sept*.

Song Han, Huizi Mao, and William J Dally. 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1135–1143.

Babak Hassibi, David G Stork, and Gregory J Wolff. 1993. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pages 293–299. IEEE.

Yu Ji, Ling Liang, Lei Deng, Youyang Zhang, Youhui Zhang, and Yuan Xie. 2018. Tetris: Tile-matching the tremendous irregular sparsity. *Advances in neural information processing systems*, 31.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. 2017. Pruning filters for efficient convnets. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Yun Li, Zechun Liu, Weiqun Wu, Haotian Yao, Xiangyu Zhang, Chi Zhang, and Baoqun Yin. 2022. Weight-dependent gates for network pruning. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(10):6941–6954.

Yun Li, Weiqun Wu, Zechun Liu, Chi Zhang, Xiangyu Zhang, Haotian Yao, and Baoqun Yin. 2020. Weight-dependent gates for differentiable neural network pruning. In *European Conference on Computer Vision Workshops*, pages 23–37. Springer.

Yun Li, Chen Zhang, Shihao Han, Li Lyna Zhang, Baoqun Yin, Yunxin Liu, and Mengwei Xu. 2021. Boosting mobile cnn inference through semantic memory. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 2362–2371.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *arXiv preprint arXiv:2007.08124*.

Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. 2019. Metapruning: Meta learning for automatic neural network channel pruning. In *Proceedings of the IEEE*

international conference on computer vision (ICCV), pages 3296–3305.

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

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. 2018. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*.

Christos Louizos, Max Welling, and Diederik P Kingma. 2017. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*.

Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. 2017. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 5058–5066.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. *arXiv preprint arXiv:2305.11627*.

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

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*.

Asit Mishra, Jorge Albericio Latorre, Jeff Pool, Darko Stosic, Dusan Stosic, Ganesh Venkatesh, Chong Yu, and Paulius Micikevicius. 2021. Accelerating sparse deep neural networks. *arXiv preprint arXiv:2104.08378*.

C Pedersen, M Otokiak, I Koonoo, J Milton, E Maktar, A Anaviapik, M Milton, G Porter, A Scott, C Newman, et al. 2020. Sciq: an invitation and recommendations to combine science and inuit qaujimajatuqangit for meaningful engagement of inuit communities in research. *Arctic Science*, 6(3):326–339.

Jeff Pool. 2020. Accelerating sparsity in the nvidia ampere architecture. *GTC 2020*.

Jeff Pool and Chong Yu. 2021. Channel permutations for n: M sparsity. *Advances in neural information processing systems*, 34:13316–13327.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Claude Elwood Shannon. 1948. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423.

Siddharth Singh and Abhinav Bhatele. 2023. Exploiting sparsity in pruned neural networks to optimize large model training. *arXiv preprint arXiv:2302.05045*.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2023. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

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

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2023. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv:2310.06694*.

Mengzhou Xia, Zexuan Zhong, and Danqi Chen. 2022. Structured pruning learns compact and accurate models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1513–1528.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR.

Zhuliang Yao, Shijie Cao, Wencong Xiao, Chen Zhang, and Lanshun Nie. 2019. Balanced sparsity for efficient dnn inference on gpu. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 5676–5683.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Aojun Zhou, Yukun Ma, Junnan Zhu, Jianbo Liu, Zhijie Zhang, Kun Yuan, Wenxiu Sun, and Hongsheng Li. 2021. Learning n: m fine-grained structured sparse neural networks from scratch. *arXiv preprint arXiv:2102.04010*.