Pruning Large Language Models to Intra-module Low-rank Structure with Transitional Activations

Anonymous ACL submission

Abstract

Structured pruning offers a viable approach to the local deployment of large language models (LLMs) by reducing computational and memory overheads. Compared to unstructured pruning and quantization, structured pruning has the advantage of being recoverable, since the pruned model remains dense and high-precision rather than sparse or low-precision. However, achieving a high compression ratio for scaled-up LLMs remains a challenge, as the coarse-grained structured pruning poses large damage to the highly interconnected model. In this paper, we introduce TransAct, a task-agnostic structured pruning approach coupled with a compact architecture design. TransAct reduces transitional activations inside multi-head attention (MHA) and multi-layer perceptron (MLP) modules, while preserving the inter-module activations that are sensitive to perturbations. Hence, the LLM is compressed into an intra-module low-rank architecture, significantly reducing weights and KV Cache. TransAct is implemented on the Llama2 model and evaluated on downstream benchmarks. Results verify the optimality of our approach at high compression with respect to both speed and performance. Furthermore, ablation studies revealed the strength of iterative pruning and provides insights on the redundancy of MHA and MLP modules.

1 Introduction

Deploying large language models (LLMs) locally on edge devices instead of relying on remote APIs has been a pressing initiative. Local deployment of LLMs ensures independence from network conditions and enhances privacy at an advanced level (Ma et al., 2023a). Nevertheless, deploying a scaled-up LLM onto a resource-constrained end device poses multifaceted challenges, encompassing inference speed, memory footprint, and power consumption. Therefore, comprehensive optimizations on the efficiency of LLMs are imperative, including architecture design (Gu and Dao, 2023), model compression (Zhu et al., 2023), inference schemes (Leviathan et al., 2023; Cai et al., 2024), compilation and runtime (Lai et al., 2023).

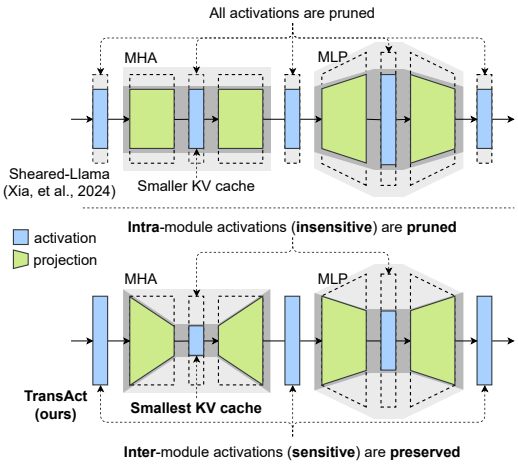

Figure 1: An illustration of TransAct model structure. The model weights and activations are colored green and blue, respectively. Dashed hollow blocks represent the weights and activations that are pruned out.

Model compression emerges as the silver-bullet solution for reducing deployment costs given an accessible LLM. To essentially reduce model computation and memory overhead, pruning aims to discard weights with low salience to the LLM. Jaiswal et al. (2023) suggest that state-of-the-art (SOTA) unstructured pruning approaches i.e., SparseGPT (Frantar and Alistarh, 2023) and Wanda (Sun et al., 2023), along with their semi-structured variations, often underperform in downstream benchmarks. Zimmer et al. (2023) emphasize the significance of post-training after pruning to restore the capabilities of the LLM. However, the post-training and inference of a sparse model are notably inefficient. Also, an unstructured pruning with arbitrary sparsity pattern has no speedup or memory saving on the LLM, whereas a semi-

structured sparse model heavily relies on specific hardware (Frantar and Alistarh, 2023).

An alternative pruning category, i.e., structured pruning, has shown feasibility for LLMs. LLM-Pruner (Ma et al., 2023b), the pioneering structured pruning of LLM, incorporates the approximated Taylor series as the pruning metric. However, this approximation loses accuracy when pruning a large ratio of the model (LeCun et al., 1989). While Taylor expansion assumes small perturbations, it is not applicable when a large number of parameters are pruned (i.e., set to zero). The SOTA approach Sheared-Llama (Xia et al., 2023), on the other hand, completely transfers the evaluation of the pruning metric to supervised training with masks. However, training with masks poses much more computation and memory footprint at training time, as well as the training unstableness. Also, the pruned architecture of Sheared-Llama, as illustrated in the upper part of Figure 1, involves the unified pruning of layer normalization (LN) weights, disregarding the varying sensitivity of LN parameters to perturbation across layers (Zhao et al., 2023).

To address the challenges of efficient and effective LLM pruning, we propose TransAct, a transitional activation-based structured pruning approach. From the perspective of pruning architectural design, TransAct reduces intra-module activations, which prunes the MHA and MLP in LLM into low intrinsic dimension as depicted in Figure 1. TransAct pruning metric is inspired by the observation of Dettmers et al. (2022) that a small proportion of activations within the LLM exhibit outlier magnitudes, rendering them particularly sensitive to perturbations and need to be preserved. This approach effectively reduces the memory footprint of both model weights and KV cache, alleviating the memory constraints inherent in autoregressive generation on edge devices (Kwon et al., 2023). Specifically, the contributions of this paper are outlined as follows.

- We propose a co-design of pruning architecture and pruning metric named TransAct, which substantially compresses the KV cache as well as the model weights.
- TransAct pruning architecture achieves the fastest inference speed among SOTA pruned models, while the pruning is efficient without gradients or masked training.
- Experiment results on downstream benchmarks verified the stableness of TransAct at a high compression ratio. Ablation studies on module redundancy provide insights for compact model design.

2 Related Work

Extensive works have been proposed to optimize the efficiency of Transformer-based LMs, covering pruning, quantization, dynamic acceleration, etc. However, to generalize these approaches to the continually scaling-up LLMs remains challenging.

Quantization, which reduces the bit representation of values, stands out due to its ease of implementation. Post-training quantization (PTQ) approaches, e.g., GPTQ (Frantar et al., 2022) and AWQ (Lin et al., 2023), are without any further tuning after the quantization. On the contrary, quantization-aware training (QAT) approaches train the model along with the quantization parameters and is still challenging when the LLM is scaled up (Liu et al., 2023). Quantizing an LLM from `float16` to `int3` with weight-only PTQ approaches like GPTQ (Frantar et al., 2022) can reach roughly 80% compression of model weights. However, the KV cache which contributes to a large amount of memory overheads is still in `float16` and uncompressed. Also, obtaining an acceptable quantization precision with `int3` weights remains a challenge. Xiao et al. (2023) proposed a W8A8 PTQ approach where both weights and activations are quantized to `int8`, saving 50% memory footprint. The lack of flexibility poses a significant limitation to quantization. Most general computing platforms and libraries primarily support low-bit representations such as `int8` and `int4` (Nagel et al., 2021). However, opting for representations lower than 4 bits necessitates dequantization back to the supported higher-bit representations, thereby introducing additional computation and memory overheads.

Apart from quantization, unstructured pruning is also an efficient approach to obtain a sparse LLM. Frantar and Alistarh (2023) and Sun et al. (2023) enabled fully unstructured and semi-structured N:M sparsity (i.e., N zeros in M consecutive weights) of LLM across different sizes. However, there are two major obstacles hindering the adoption of unstructured sparsity. (1) The pruned sparse LLM cannot be efficiently further trained. Although Sun et al. (2023) claimed to use

LoRA (Hu et al., 2022) to train the compressed model, the LoRA modules cannot be merged into the sparse backbone LLM, which incurs additional overhead at inference time. (2) The sparsity is fixed at 50% with current hardware and platform affinity. While only NVIDIA Ampere and Hopper GPUs support the 2:4 sparsity pattern, achieving customized sparsity requires hardware co-design (Fang et al., 2022). This limitation restricts the broader application of unstructured pruning.

The approaches discussed above are static compression of LLM, where the computation at inference is fixed. On the contrary, dynamic acceleration at inference time speeds up LLM generation by selective computation. Early exiting approaches (Schuster et al., 2022; Corro et al., 2023) allow the LLM to finish the decoding of a token without passing all the layers. Mixture-of-Expert (MoE) architecture (Jiang et al., 2024; Lepikhin et al., 2021) incorporates multiple parameter shards in MLP as experts and selects experts to compute when facing different inputs. The dynamic approaches usually do not reduce parameters. Thus, the storage of the model is not reduced, while the runtime memory can be saved by fine-grained neuron-aware offloading (Song et al., 2023).

3 Methodology

In this section, we first recap the preliminaries of Transformer-based LLM architecture and introduce the transitional activations. Then, we propose our approach TransAct with the pruning metric and architecture design of the pruned model.

3.1 Transitional Activations in LLM

Transformer-based LLMs generally consist of embedding, MHA (multi-head attention), MLP (multi-layer perceptron), and LM head.

The majority of model weights lie in MHA and MLP, which exist in every Transformer layer of the LLM. Specifically, MHA has three matrices $\boldsymbol{W_Q}$, $\boldsymbol{W_K}$, $\boldsymbol{W_V}$ with the shape of $H \times A$, and one matrix $\boldsymbol{W_O}$ of the inverted. The MHA mechanism splits the output dimension A into $A_n \times A_d$ (i.e., head number by head dimension), which forms A_n logical attention heads. The input activation $\boldsymbol{h}^l$ of the l-th layer is projected by $\{\boldsymbol{W_Q}_k^l, \boldsymbol{W_K}_k^l, \boldsymbol{W_V}_k^l\}_{k=1}^{A_n}$ and split into A_n groups of query, key, and value $\{\boldsymbol{q}_k^l, \boldsymbol{k}_k^l, \boldsymbol{v}_k^l\}_{k=1}^{A_n}$. Then the multi-head self-attention computation is as

$$\boldsymbol{act_A}_k^l = \mathrm{Softmax}(\boldsymbol{q}_k^l \boldsymbol{k}_k^{l\intercal} / \sqrt{A_d}) \boldsymbol{v}_k^l, \quad (1)$$

where k is the attention head index counted from 1 to A_n, and l indicates the l-th layer. $H \times A_d$ at the superscript is the shape annotation of the weight matrix. Then, the results are concatenated to shape A and projected back to shape H by $\boldsymbol{W_O}$.

$$\boldsymbol{h_A}^l = \mathrm{Concat}[\boldsymbol{act_A}_k^l]_{k=1}^{A_n} {\boldsymbol{W_O}^l}^{A \times H}. \quad (2)$$

As a bound between the group of $\boldsymbol{W_Q}$, $\boldsymbol{W_K}$, $\boldsymbol{W_V}$ and $\boldsymbol{W_O}$, we define $\boldsymbol{act_A}^l$ as the transitional activation of MHA module. By default, the transitional size A of MHA is the same as hidden dimension H, but A can be smaller than H by reducing A_n or A_d in the case of pruning.

The other module, MLP, has a pair of upcast and downcast phases. In the first phase, the input hidden state $\boldsymbol{h}$ is projected to a transitional state with larger dimension P through $\boldsymbol{W_U}$ and an optional gate $\boldsymbol{W_G}$, the later phase consists of a downcast $\boldsymbol{W_D}$ that projects the transitional state back to the original shape H. We consider $\boldsymbol{W_G}$ exists and formulate MLP as follows.

$$\boldsymbol{act_P}^l = \sigma(\boldsymbol{h_A}^l {\boldsymbol{W_G}^l}^{H \times P}) \odot (\boldsymbol{h_A}^l {\boldsymbol{W_U}^l}^{H \times P}), \quad (3)$$

$$\boldsymbol{h_P}^l = \boldsymbol{act_P}^l {\boldsymbol{W_D}^l}^{P \times H}. \quad (4)$$

In case there is no optional gating in the model, the σ function can be viewed as $\sigma(\cdot) \equiv 1$. Similar to the MHA module, we define $\boldsymbol{act_P^l}$ as the transitional activation of the MLP module at the l-th layer.

3.2 Pruning with Transitional Activations

Based on the model structure, we identify the pruning target as the following. (1) A_n, i.e., the number of attention heads in MHA. On the other hand, A_d is kept intact, as reducing it incurs the adaption of RoPE (rotary positional embedding) (Su et al., 2024) used by a high quantity of LLMs and increases the training unstableness. (2) P, i.e., the transitional dimension of MLP. Studies (Mirzadeh et al., 2023) indicate that, with an activation function suppressing negative values, the transitional state of MLP is with high redundancy. It is worth mentioning that H, i.e., the hidden dimension throughout the model is not compressed. We justify the reason as compressing H incurs the unified pruning of layer normalization (LN)

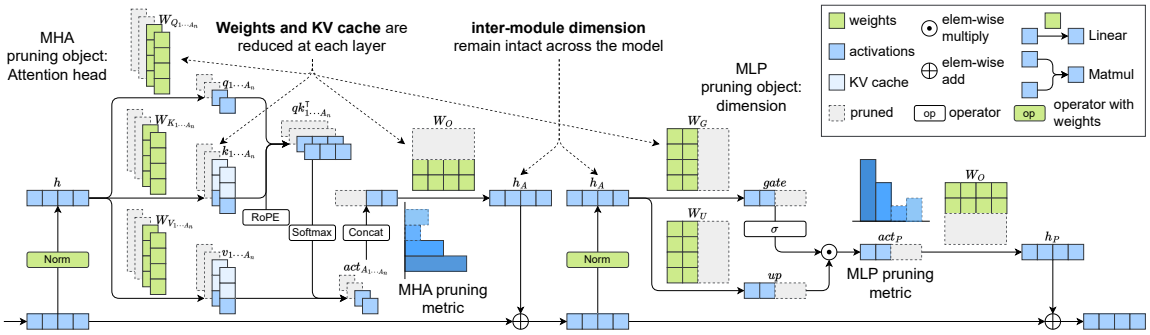

Figure 2: Detailed TransAct workflow on a Transformer layer in Llama. Bar charts indicate the activation-based pruning metric.

weights across layers, whereas the sensitivity of LN parameters to perturbation is not unified across layers (Zhao et al., 2023). Although further training can reconstruct the LN module from the damage of compressing, the significant training cost is contrary to efficiency.

With the definition of transitional activations and the pruning objects, we propose the transitional activation-based pruning approach to compress MHA and MLP modules into an intra-module low-rank architecture as depicted in Figure 2. For the MHA module, we define the pruning granularity (i.e., the least separable structure) to be the attention head, in turn reducing A_n while keeping A_d intact. Such an attention head pruning is unified on $\boldsymbol{W_Q}$, $\boldsymbol{W_K}$ and $\boldsymbol{W_V}$ because the self-attention calculation, as formulated in Equation 1, requires the aligned head index among the three matrices. Then, we can define the salience of all heads in MHA as

$$\mathcal{S}^l_{\mathcal{A}k} = \frac{1}{A_d}\sum_{i=0}^{A_d}\left\|\boldsymbol{act_A}^l_{ki}\right\|_2 + \alpha \max_i \left\|\boldsymbol{act_A}^l_{ki}\right\|_2, \tag{5}$$

where α is a weight factor amplifying the maximum activation value in the k-th head. By Equation 5, we want to evaluate both the general and outlier values in the activations, so that we can precisely prune out the most insignificant head. For MLP, we can simply use the corresponding value of $\boldsymbol{act_P}$ to represent the salience of MLP transitional dimension as $\mathcal{S}^l_{\mathcal{P}i} = \left\|\boldsymbol{act_P}^l_i\right\|_2$.

With the salience $\mathcal{S}_\mathcal{A}$ and $\mathcal{S}_\mathcal{P}$ formulated, we can model the activation-based structured pruning of a weight matrix W as

$$\text{prune}(\boldsymbol{W}, K, \mathcal{S}) = \text{Concat}[\boldsymbol{W}_i]_{i\in \arg\text{topK}(\mathcal{S})}. \tag{6}$$

Specifically, the pruning dimension of $\boldsymbol{W_Q}$, $\boldsymbol{W_K}$, $\boldsymbol{W_V}$, $\boldsymbol{W_G}$ (optional) and $\boldsymbol{W_U}$ is the output, while the pruning dimension of $\boldsymbol{W_O}$ and $\boldsymbol{W_D}$ is the input as depicted in Figure 2.

Obtaining the salience of the source LLM requires only forward passes with a small amount of calibration samples. Hence, the pruning procedure is efficient in both memory and computation. To avoid a single shot pruning to compression ratio R posing unrecoverable damage to the model, we provide an enhanced implementation where the model is iteratively pruned to the target size. A set of pruning ratios is defined as $\mathcal{R} = \{r_1, r_2, \cdots, r_n\}$, where the i-th shot prunes the model to the size of (A'_i, P'_i) subject to $\sum_{r_i\in\mathcal{R}} = R$, and $A'_i \mod A_d = 0$. During the interval of two pruning steps, full fine-tuning (FT) is performed on the model to recover the pruning damage. The pipeline of TransAct is introduced in Algorithm 1.

Algorithm 1 TransAct pruning and post-training

procedure EVAL_PRUNE($\mathcal{M}, X, r$)
  $\boldsymbol{act_A}, \boldsymbol{act_P} \leftarrow \mathcal{M}(X)$
  for $l \in \mathcal{M}.layers$ **do**
    $\mathcal{S}^l_\mathcal{A}, \mathcal{S}^l_\mathcal{P} \leftarrow \text{salience}(\boldsymbol{act_A}^l, \boldsymbol{act_P}^l)$
    for $\boldsymbol{W}_i \in \mathcal{A}^l \cup \mathcal{P}^l$ **do**
      $K \leftarrow \lfloor r \times \text{len}(\mathcal{S}^l_i) \rfloor$
      $\boldsymbol{W}' \leftarrow \text{prune}(\boldsymbol{W}, K, \mathcal{S}^l_i)$
    end for
  end for
  return $\mathcal{M}$
end procedure
for $r_i \in \{r_1, \cdots r_n\}$ **do**
  $\mathcal{M} \leftarrow$ Eval_Prune($\mathcal{M}, X, r_i$)
  $\mathcal{M} \leftarrow$ Full-FT($\mathcal{M}$)
end forreturn $\mathcal{M}$

4 Experimental Evaluation

4.1 Experiment Setup

Model and Datasets Settings In this paper, we select the representative Llama2-7B-base (Touvron et al., 2023) as the source model to prune, as its size is suitable for experiments and has shown important features of LLMs. We also use the pre-trained OPT-1.3B and OPT-2.7B (Zhang et al., 2022) as the baseline of the pruned models.

We use subsets of RedPajama-V1 (Together Computer, 2023), a 1 trillion-token corpus, as the training dataset. Specifically, a subset of 800 million tokens are randomly sampled in the iterative pruning process, while 50 billion tokens are randomly sampled in post-training. For evaluation, we select held-out downstream tasks from Huggingface open LLM leaderboard[1], Llama2 paper (Touvron et al., 2023), and Sheared-Llama paper (Xia et al., 2023). The tasks include zero-shot ARC-E (Clark et al., 2018), BoolQ (Clark et al., 2019), LogiQA (Liu et al., 2020), OpenbookQA (OBQA) (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), SciQ (Welbl et al., 2017) and few-shot ARC-C (Clark et al., 2018), HellaSwag (Zellers et al., 2019), TriviaQA (Joshi et al., 2017), TruthfulQA (Lin et al., 2022) and WinoGrande (Sakaguchi et al., 2020). Details of the evaluation tasks can be found in Appendix A.

Comparative Methods and Pruning Settings We compare the following baselines and methods. (1) Taylor expansion-based pruning. We reproduced LLM-Pruner (Ma et al., 2023b) with the same size as our model. (2) Masked training-based pruning. We use the open-sourced Sheared-Llama-pruned (Xia et al., 2023) and post-trained by us. The size of each module is listed in Table 1. The reproduced LLM-Pruner is not listed in Table 1 as it has the same shape with TransAct.

LLM-Pruner and TransAct are implemented in iterative pruning mode. Sheared-Llama is reproduced without dynamic batch loading to expose the real performance of pruning without adding influential factors of training. Our implementation is with DeepSpeed on 8 NVIDIA A100 80G GPU, while the sequence length is 4096. Please refer to Appendix B for more implementation details.

[1] https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard

4.2 Experiment Results

4.2.1 Speed and Memory Consumption

We deploy TransAct-2.5B and Sheared-Llama-2.7B on a single NVIDIA A100 GPU using MLC-LLM (team, 2023). The models are tested in the original `float16` precision and with `int8` and `int4` weight-only quantization. We report the generation speed with a context of 1300 words and a maximum sequence length of 4096.

As shown in Table 2, our architecture consumes 11% less memory than Sheared-Llama with a similar number of parameters, attributed to the reduced size of KV cache. Notably, an NVIDIA A100 GPU has a significantly large memory bandwidth, which makes the acceleration of TransAct architecture not obvious. However, reducing the KV cache size is crucial on resource-constrained edge devices. This importance stems from the convention that when deploying an LLM on edge devices, weight-only quantization (W4A16) is preferred over activation quantization (W8A8). As discussed in Section 2, model weights can be quantized to 4 bits while activations are generally preserved in 16 bits. Hence, despite the small number of the KV cache compared to model weights, the memory footprint of the 16-bit KV cache is comparable to that of the 4-bit model weights with 4 times amplified.

4.2.2 Pruned Model Performance

The evaluation results of pruned models on held-out benchmarks are listed in Table 3. On few-shot tasks, TransAct-2.5B achieves the best performance performance compared to SOTA approaches. TransAct exhibits a significant leap over LLM-Pruner and Sheared-Llama on TriviaQA and TruthfulQA, which evaluate the truthfulness and world knowledge of the LLM. Whereas, the pre-trained OPT models achieve the highest metric on the two tasks although other abilities are inferior to the pruned models. We interpret that TransAct better preserved the world knowledge of the original LLM, which is much harder than preserving language modeling and commonsense reasoning capabilities. At 80% compression, TransAct-1.3B achieves 78.0% performance of Llama-7B on average, addressing the effectiveness of TransAct at highly compressed settings. Whereas LLM-Pruner fails at most few-shot tasks. Thereby, we address the inapplicability of structured pruning with the Taylor expansion-based metric. LLMs

Model	Size					
	L	H	A	P	$\sum$model	$\sum$KV cache
Llama2-7B	32	4096	32×128	11008	6.74B	1073M
Sheared-Llama-2.7B	32	2560	20×128	6912	2.70B (-60%)	671M (-38%)
TransAct-2.5B (ours)	32	4096	16×128	3072	2.54B (-62%)	536M (**-50%**)
Sheared-Llama-1.3B	24	2048	16×128	5504	1.35B (-80%)	403M (-63%)
TransAct-1.3B (ours)	32	4096	6×128	1536	1.27B (-81%)	201M (**-81%**)

Table 1: Compressed models with different architectures. L is the number of layers and H is the dimension of hidden states. A denotes the MHA size as $A_n \times A_d$, and the transitional size of MLP is P. KV cache is computed with a sequence length of 4096 tokens. B and M stand for billion (10^9) and million (10^6), respectively.

Model	Metrics	fp16	int8-fp16	int4-fp16
Sheared-Llama-2.7B	Speed (token/s) ↑	78.5	96.4	101.0
	Memory (MiB) ↓	7258	5000	3704
TransAct-2.5B (ours)	Speed (token/s) ↑	**83.4**	**101.9**	**105.3**
	Memory (MiB) ↓	**6544**	**4294**	**3082**

Table 2: Generation speed tested on NVIDIA A100 with models compiled by MLC-LLM.

are fundamentally pre-trained on a large corpus to obtain world knowledge. However, the Taylor expansion-based metric, which guides the pruning by minimizing the approximated language modeling loss on a small calibration set, fails to preserve knowledge and degrade the pruned LLM. Amplifying the calibration set by a significant order of magnitude is an intuitive solution. However, the computation of Jacobian and Hessian matrices of LLM weights on a large calibration set is enormous.

Notably, the reproduced LLM-Pruner-2.5B with iterative pruning reaches 83.6% performance of the uncompressed Llama2-7B. Whereas in its original paper, the performance at 50% compression ratio can barely reach 78% of the original model (Ma et al., 2023b). The results strengthen the necessity of iterative pruning at LLM structured pruning. Specifically, iterative pruning is gradual and conservative at each step, lessening the approximation error of pruning metrics.

Figure 3 illustrate the zero-shot LAMBADA language modeling performance at each checkpoint of the pruned model post-training. Although TransAct-2.5B has a clear advantage between 10b to 30b tokens trained, the gap between different pruning approaches diminishes as the pruned model is gradually recovered by post-training. Notably, the result of LLM-Pruner-2.5B exhibits the lowest perplexity in Figure 3. However, it does not necessarily indicate the highest accuracy on LAMBADA, nor the performance on other tasks.

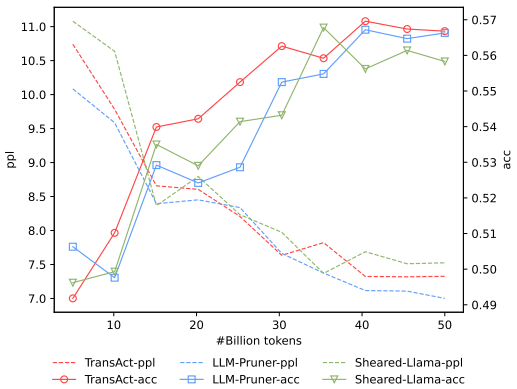

Figure 3: LAMBADA perplexity and accuracy on every checkpoint of TransAct-2.5B, LLM-Pruner-2.5B and Sheared-Llama-2.7B post-training.

4.2.3 Ablation Studies

We conduct a comprehensive evaluation of the pruned LLM, considering factors of pruning shots, calibration samples, and the pruning ratio of each module. The findings provide insights for the further development of compact LMs.

Impact of iterative pruning While LLM-Pruner has demonstrated a close performance gap to the original model at a moderate ratio of 20%, the significant performance degradation observed at over 50% pruned is far from acceptable in the original implementation (Ma et al., 2023b). However, the results in Table 3 indicate that LLM-Pruner achieves comparable performance to the SOTA approach Sheared-Llama even at a compres-

Model	ARC-E	BoolQ	LogiQA	OBQA	PIQA	SciQ
Llama2-7B	74.4	80.7	30.4	43.8	76.7	94.7
OPT-2.7B	60.8	60.4	25.7	35.2	74.5	85.9
Sheared-Llama-2.7B†	66.8	66.0	**28.1**	38.6	76.9	89.9
LLM-Pruner-2.5B*	**67.0**	65.9	27.7	**38.8**	**77.1**	90.1
TransAct-2.5B (ours)	65.5	**66.3**	27.9	38.2	76.9	**91.0**
OPT-1.3B	57.1	57.7	27.0	33.4	72.4	84.4
Sheared-Llama-1.3B†	59.3	61.6	27.5	33.0	74.2	85.8
LLM-Pruner-1.3B*	**60.0**	59.5	**28.7**	**35.2**	73.6	86.1
TransAct-1.3B (ours)	57.4	**63.4**	27.5	33.8	**74.4**	**86.7**

Model	ARC-C (25)	HellaSwag (10)	TriviaQA (5)	TruthfulQA **	WinoGrande (5)	Average
Llama2-7B	53.4	78.6	55.1	44.6	72.3	64.1
OPT-2.7B	34.0	61.4	23.7	**37.6**	61.7	51.0
Sheared-Llama-2.7B†	**40.0**	71.0	21.2	32.0	65.0	54.1
LLM-Pruner-2.5B*	38.6	70.8	17.3	32.9	63.6	53.6
TransAct-2.5B (ours)	38.9	**71.2**	**33.9**	33.6	**65.5**	**55.3**
OPT-1.3B	29.7	54.6	16.7	38.7	**60.0**	48.3
Sheared-Llama-1.3B†	30.3	**62.6**	14.0	34.1	59.3	49.2
LLM-Pruner-1.3B*	30.3	59.0	7.9	35.9	56.4	48.4
TransAct-1.3B (ours)	**32.2**	59.9	**18.4**	**39.6**	56.5	**50.0**

Table 3: Results on standard evaluation benchmarks with 50 billion tokens for post-training. Llama2-7B is the source model used as the baseline. Results of LLM-Pruner* are reproduced by us with the same training setting, while Sheared-Llama† models are post-trained without data selection. (n) below task name indicates n-shots evaluation. Note that TruthfulQA** prepends 6 examples even in the zero-shot setting. The best results are in **bold**.

sion ratio of 85%. This achievement can be attributed to our iterative implementation of pruning.

To further verify the effectiveness of iterative pruning, we conduct experiments on LLM-Pruner-2.5B and TransAct-2.5B with different numbers of pruning shots. Specifically, we explore pruning shots ranging from $\{1, 2, 4, 8, 16\}$. Except for single-shot pruning, all others have a total of 800 million tokens throughout the iterative pruning stage. After the final pruning, all models undergo full fine-tuning with 200 million tokens. Sheared-Llama is considered an ∞-shot pruning approach with all the parameters trained and is not compared.

Results in Figure 4 indicate the relationship between pruning shots and performance on LAMBADA language modeling. Although iterative pruning is beneficial, the pruning shots need to be controlled with a total number of tokens is fixed. The performance of 2.5B models degrades when the pruning shot is increased from 4 to 8. The rationale of this phenomenon is that when training is insufficient between two pruning shots, the pruning would be misguided and the pruned model would exhibit a degradation. Whereas, for 1.3B models, the performance exhibits a slight degradation at 16 shots, indicating the benefit of increased shots has not yet been overwhelmed by the insufficiency of training data. LLM-Pruner has a slight advantage over TransAct at 16 shots pruning, as fewer parameters pruned at each shot reduce the approximation error of loss with Taylor expansion.

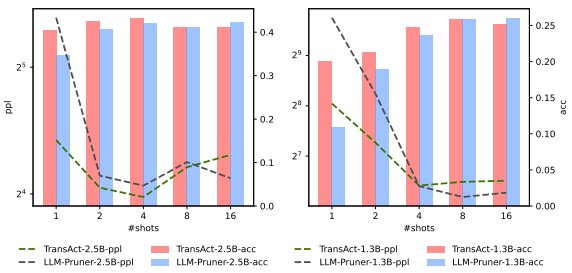

Figure 4: LAMBADA perplexity and accuracy on models with different numbers of pruning shots.

Impact of calibration samples To evaluate the sensitivity of pruning methods to calibration samples used in the pruning process, we conduct single-shot pruning experiments on different numbers of calibration samples. 200 million tokens are

used for the restoration after pruning.

The results in Figure 5 indicate that increasing the sample size can bring gains, but the marginal benefits decrease after increasing to 128 samples. When leveraging 256 samples, the performance of both TransAct and LLM-Pruner degrade. Also, the degradation trend is more obvious on LLM-Pruner than on TransAct. We attribute this to early overfitting of calibration samples, where the pruning guided by Taylor expansion of loss quickly overfits on the calibration set, and the calibration samples are not large enough to exhibit diversity. As pruning is efficient in our implementation, we prefer using 128 samples for the pruning metric, which can be computed in less than 1 minute on a single A100 GPU to prune Llama2-7B.

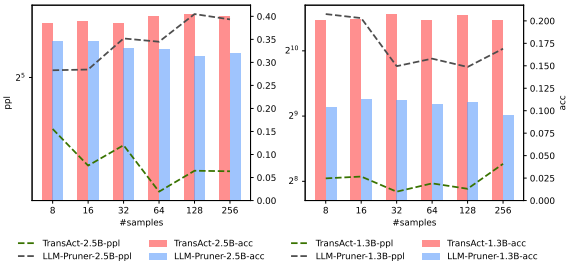

Figure 5: LAMBADA perplexity and accuracy on models with different numbers of calibration samples.

Analysis on module redundency To help future compact model design, we conduct experiments on different compression ratios. After single-shot pruning with TransAct and post-training on 200 million tokens, the accuracy of LAMBADA language modeling is evaluated. Specifically, using the shape of our TransAct-2.5B as the center point, we vary the MHA size A ranged from $\{512, 1280, 2048, 2816, 3584\}$ and the MLP size P ranged from $\{1024, 2048, 3072, 4096, 5120\}$. These configurations resulted in 25 distinct models obtained by pairwise combinations. Notably, the 25 models are organized into 9 groups, each containing an equal number of parameters. These groups are visually distinguished by color in Figure 6.

The results presented in Figure 6 reveal a clear trend that, the models at the center exhibit the best performance within each group, and in some cases, even surpass models of larger sizes. For instance, the combination of 2048A-3072P (i.e., TransAct-2.5B) model surpasses both 3584A-2048P and 1280A-5120P (2.9B) models. Also, when pruning the MHA intermediate size to 512, the performance drops to the worst within each group. We interpret that MHA functions as the crucial module of Transformer-based LLMs while MLP has a larger redundancy that can be compressed. Further, the findings indicate that models with a uniform MHA and MLP size generally outperform the others. For 2048A-3072P, an MHA module has 33.5 million parameters and an MLP module has 37.7 million parameters. On the contrary, extreme pruning of either MHA or MLP alone leads to severe performance degradation. Hence, the collaborative compression of both MHA and MLP is encouraged.

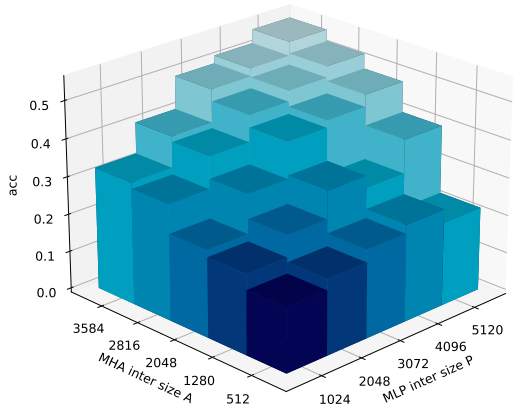

Figure 6: LAMBADA accuracy on 25 pruned models with different architectures. Bars with the same color indicate models with the same number of parameters.

5 Conclusion

In this paper, we introduce TransAct, an effective and efficient pruning approach coupled with an architecture designed for pruned LLMs. TransAct compresses the original LLM into a compact dense model with an intra-module low-rank structure, achieving the fastest inference speed compared to models of similar sizes. To identify the dimensions to preserve, we investigate the transitional activations at the low-rank bottleneck and use their magnitudes as the pruning metric. Experiments on downstream benchmarks demonstrate the strength of our approach, particularly at high compression rates. Also, we thoroughly evaluated the pruned LLM with respect to calibration samples, pruning ratio, and pruning shots. The results provide valuable insights for the further development of compact LMs.

Limitations

Although InterAct is found effective in the experiments, some points are not fully covered in this paper. We list the limitations and future directions as follows. (1) InterAct is a static pruning approach where the computation of the pruned LLM is irrelvant to input instances. However, recent research progress in MoE (Jiang et al., 2024) indicates that dynamically compressed models are model powerful than statically compressed ones. Hence, a pruning approach integrating static and dynamic compression with approporate ratio can be further studied. (2) InterAct is targeted to Transformer-based LLMs. Different architectures including RWKV (Peng et al., 2023), Mamba (Gu and Dao, 2023) are not yet investigated. (3) The pruning of InterAct is conducted on base models, and further alignment is needed before the pruned model can be used in human-computer interaction. Structurally pruning a human-aligned LLM still remains challenging.

References

Yonatan Bisk, Rowan Zellers, Ronan LeBras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7432–7439. AAAI Press.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *ArXiv preprint*, abs/2401.10774.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Luciano Del Corro, Allie Del Giorno, Sahaj Agarwal, Bin Yu, Ahmed Awadallah, and Subhabrata Mukherjee. 2023. Skipdecode: Autoregressive skip decoding with batching and caching for efficient llm inference.

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

Chao Fang, Aojun Zhou, and Zhongfeng Wang. 2022. An algorithmhardware co-optimized framework for accelerating n:m sparse transformers. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 30(11).

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pages 10323–10337. PMLR.

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

Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *ArXiv preprint*, abs/2312.00752.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Ajay Jaiswal, Zhe Gan, Xianzhi Du, Bowen Zhang, Zhangyang Wang, and Yinfei Yang. 2023. Compressing llms: The truth is rarely pure and never simple. *ArXiv preprint*, abs/2310.01382.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *ArXiv preprint*, abs/2401.04088.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention.

Ruihang Lai, Junru Shao, Siyuan Feng, Steven S Lyubomirsky, Bohan Hou, Wuwei Lin, Zihao Ye, Hongyi Jin, Yuchen Jin, Jiawei Liu, et al. 2023. Relax: Composable abstractions for end-to-end dynamic machine learning. *ArXiv preprint*, abs/2311.02103.

Yann LeCun, John Denker, and Sara Solla. 1989. Optimal brain damage. In *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2021. Gshard: Scaling giant models with conditional computation and automatic sharding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. 2023. Awq: Activation-aware weight quantization for llm compression and acceleration. *ArXiv preprint*, abs/2306.00978.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland. Association for Computational Linguistics.

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. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3622–3628. ijcai.org.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. 2023. Llm-qat: Data-free quantization aware training for large language models. *ArXiv preprint*, abs/2305.17888.

Ruilong Ma, Jingyu Wang, Qi Qi, Xiang Yang, Haifeng Sun, Zirui Zhuang, and Jianxin Liao. 2023a. Pipellm: Pipeline llm inference on heterogeneous devices with sequence slicing. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pages 1126–1128.

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

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. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391.

Iman Mirzadeh, Keivan Alizadeh, Sachin Mehta, Carlo C Del Mundo, Oncel Tuzel, Golnoosh Samei, Mohammad Rastegari, and Mehrdad Farajtabar. 2023. Relu strikes back: Exploiting activation sparsity in large language models.

Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart Van Baalen, and Tijmen Blankevoort. 2021. A white paper on neural network quantization. *ArXiv preprint*, abs/2106.08295.

Denis Paperno, German David Kruszewski Martel, Angeliki Lazaridou, Ngoc Pham Quan, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda Torrent, Fernández Raquel, et al. 2016. The lambada dataset: Word prediction requiring a broad discourse context. In *The 54th Annual Meeting of the Association for Computational Linguistics Proceedings of the Conference: Vol. 1 Long Papers*, volume 3, pages 1525–1534. ACL.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, Kranthi Kiran GV, et al. 2023. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8732–8740. AAAI Press.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. 2022. Confident adaptive language modeling.

Yixin Song, Zeyu Mi, Haotong Xie, and Haibo Chen. 2023. Powerinfer: Fast large language model serving with a consumer-grade gpu.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

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

MLC team. 2023. MLC-LLM.

Together Computer. 2023. Redpajama: an open dataset for training large language models.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava,

Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *ArXiv preprint*, abs/2307.09288.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text, NUT@EMNLP 2017, Copenhagen, Denmark, September 7, 2017*, pages 94–106. Association for Computational Linguistics.

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

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.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

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. 2022. Opt: Open pre-trained transformer language models.

Jun Zhao, Zhihao Zhang, Yide Ma, Qi Zhang, Tao Gui, Luhui Gao, and Xuanjing Huang. 2023. Unveiling a core linguistic region in large language models. *ArXiv preprint*, abs/2310.14928.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. 2023. A survey on model compression for large language models. *ArXiv preprint*, abs/2308.07633.

Max Zimmer, Megi Andoni, Christoph Spiegel, and Sebastian Pokutta. 2023. Perp: Rethinking the prune-retrain paradigm in the era of llms. *ArXiv preprint*, abs/2312.15230.

A Details of Evaluation Tasks

The downstream tasks used for evaluation are listed in Table 4. The evaluations are conducted based on lm-evaluation-harness [2] repository with MIT license. In Table 4, "acc_norm" stands for accuracy after normalization by byte-length, "em" stands for exact match, and "mc2" stands for the normalized probability assigned to all true answers in multiple choices (Lin et al., 2022).

B Details of Training Arguments

The training arguments are listed in Table 5. The experiments are conducted on Huggingface Transformers with DeepSpeed and FlashAttention2 integration. We set the training arguments based on accessible computational resources and setting of Xia et al. (2023). There is no hyperparameters searching or tuning in this work, and we believe it is potentially beneficial to tune the hyperparameters with sufficient resources.

[2] https://github.com/EleutherAI/lm-evaluation-harness

Task	Used by (1)	(2)	(3)	#samples	#shots	Metric
ARC-C (Clark et al., 2018)	✓	✓	✓	1172	25	acc_norm
ARC-E (Clark et al., 2018)		✓	✓	2376	-	acc
BoolQ (Clark et al., 2019)		✓	✓	3270	-	acc
HellaSwag (Zellers et al., 2019)	✓	✓	✓	10042	10	acc_norm
LAMBADA (Paperno et al., 2016)			✓	5153	-	ppl & acc
LogiQA (Liu et al., 2020)		✓	✓	651	-	acc_norm
OBQA (Mihaylov et al., 2018)		✓		500	-	acc_norm
PIQA (Bisk et al., 2020)		✓	✓	1838	-	acc_norm
SciQ (Welbl et al., 2017)			✓	1000	-	acc
TriviaQA (Joshi et al., 2017)		✓		11313	5	em
TruthfulQA (Lin et al., 2022)	✓	✓		817	*	mc2
WinoGrande (Sakaguchi et al., 2020)	✓	✓	✓	1267	5	acc

Table 4: Details of evaluation tasks. (1), (2) and (3) refer to Open LLM Leaderboard, Llama2 paper and Sheared-Llama paper, respectively. TruthfulQA prepends 6 examples even in zero-shot setting.

Argument	Value
Length	4096
N GPUs	8
Global batch size	64
Optimizer	AdamW
β_1, β_2	0.9, 0.95
Learning rate	5e-5
Learning rate schelduler	Cosine
Warmup	0.03
Data type	`bfloat16`
DeepSpeed	Zero-2
Attention implementation	FlashAttention2

Table 5: Details of training arguments.