GROUPMASK: LAYER-ADAPTIVE GROUP-WISE SPARSITY FOR SEMI-STRUCTURED LLM PRUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Semi-structured pruning compresses large language models (LLMs) while keeping a regular sparse structure, but the prevailing N:M pattern fixes the same local sparsity ratio in every layer. Layer-adaptive sparsity allocation improves unstructured pruning, yet it has been reported to be less effective under N:M sparsity, leaving open whether adaptive allocation is of limited value for semi-structured pruning in general or only under the fine-grained N:M pattern. We examine this question with group-level sparsity, which partitions each weight matrix into regular groups, retains or prunes each group as a whole, and allows each layer's sparsity ratio to vary under a global budget. We propose **GroupMask**, which generates the group selectors of all layers with a lightweight hypernetwork, relaxes them with a Gumbel-Sigmoid parameterization and a straight-through estimator, and learns them through sparsity-budget regularization and self-distillation while keeping the pretrained weights frozen. On LLaMA-2-7B at 50% sparsity with the same 1×256 group size, learned layer-adaptive allocation reduces WikiText-2 perplexity from 10.02 to 8.30 and raises the average zero-shot accuracy from 0.455 to 0.496 relative to a uniform per-layer ratio. GroupMask obtains the lowest WikiText-2 perplexity on LLaMA-2-7B and the highest average zero-shot accuracy with Alpaca calibration among the evaluated baselines on five LLaMA and Qwen models. Our code is available at https://anonymous.4open.science/r/GroupMask-D037.

1 INTRODUCTION

Large Language Models (LLMs) have improved performance across a wide range of natural language processing tasks by scaling parameters, training data, and computation Zhang et al. (2022); Touvron et al. (2023); Zhao et al. (2023). This scale also incurs high computational and memory costs, which complicates deployment on resource-constrained hardware and in latency-sensitive applications. Model compression aims to reduce these costs while preserving the general capabilities of LLMs.

Existing compression techniques for LLMs include pruning Han et al. (2015b;a); Frantar and Alistarh (2023); Sun et al. (2023), knowledge distillation Hinton et al. (2015), quantization Frantar et al. (2022), and low-rank or modular compression Wang et al. (2025); Lin et al. (2025). We focus on pruning, which removes redundant parameters directly from pretrained models. According to the granularity of sparsity, pruning methods can be divided into unstructured, structured, and semi-structured pruning. Unstructured pruning removes individual weights and offers the most flexibility, which helps preserve model quality LeCun et al. (1989); Hassibi and Stork (1992); Frantar and Alistarh (2023); Sun et al. (2023); however, its irregular sparsity leads to inefficient memory access and limited acceleration on GPUs without specialized sparse kernels Fan et al. (2025). Structured pruning removes larger units such as channels, hidden dimensions, attention heads, or layers, producing models that run efficiently on standard hardware Liu et al. (2017); Ma et al. (2023); Men et al. (2025); Ashkboos et al. (2024); Gao et al. (2024b), but its coarse granularity typically causes a larger loss in quality, especially when a large fraction of parameters is removed.

Semi-structured pruning lies between these two extremes Mishra et al. (2021); Pool and Yu (2021); Zhou et al. (2021). Existing semi-structured methods mainly adopt N:M sparsity, which retains N non-zero weights in every M consecutive weights; the 2:4 case is natively accelerated on NVIDIA Ampere

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

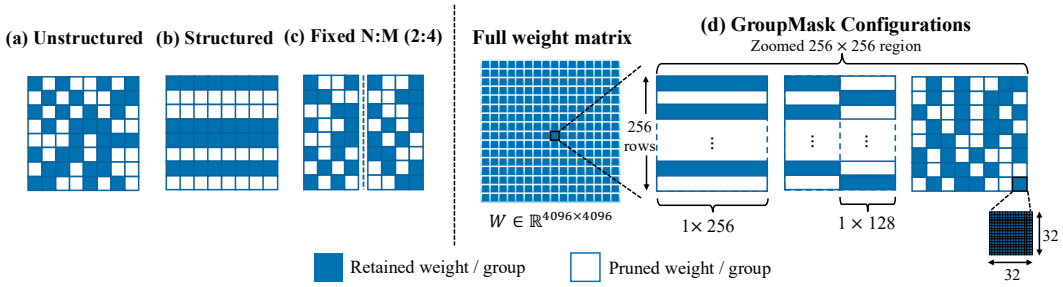


Figure 1: Comparison of pruning granularities: (a) unstructured pruning, (b) structured pruning, (c) fixed N:M sparsity, and (d) group-level sparsity with different group shapes.

and later GPUs Mishra et al. (2021); Zhou et al. (2021). Learnable methods such as MaskLLM and ProxSparse further optimize N:M masks end-to-end instead of relying on hand-crafted importance scores Fang et al. (2024); Liu et al. (2025). Regardless of how the mask is obtained, N:M sparsity fixes the local sparsity ratio to $1 - N/M$ in every layer and module.

For unstructured pruning, allocating sparsity non-uniformly across layers has been shown to improve over a uniform ratio, with the per-layer ratios either derived from outlier statistics Yin et al. (2024) or learned Xu et al. (2024). Under N:M sparsity, however, the same benefit is not observed: Liu et al. (2025) report that non-uniform layer-wise ratios become less effective under fine-grained N:M patterns, since critical weights may still be removed within each block. Two explanations are consistent with this observation: layer-adaptive allocation may be of limited value for semi-structured LLM pruning in general, or the fine-grained local constraint of N:M may prevent it from taking effect.

Distinguishing these explanations requires a sparse pattern that is regular but imposes no fixed local ratio. We therefore study group-level sparsity, illustrated in Figure 1(d). Whereas N:M sparsity keeps exactly N weights in every M-wide window, group-level sparsity partitions each weight matrix into regular groups, such as 1×256, 1×128, or 32×32, and retains or prunes each group as a whole. The sparsity ratio of a layer is then determined by how many of its groups are retained, and can fall above or below the global budget.

Selecting groups under a global budget is a combinatorial problem, since the decision for each group depends on its own importance, on the layer it belongs to, and on the budget consumed by the other layers. We address this problem with **GroupMask**. A lightweight hypernetwork produces selector logits for the groups of all layers, which are converted into binary selectors through a Gumbel-Sigmoid parameterization with a straight-through estimator (STE). A regularizer on the global kept-parameter ratio enforces the budget, and self-distillation from the dense model provides the training signal Hinton et al. (2015); Xia et al. (2022). The pretrained weights remain frozen, so the compressed model is a sparse sub-network of the original LLM.

Under the same 1×256 group structure on LLaMA-2-7B at 50% sparsity, learned layer-adaptive allocation reduces WikiText-2 perplexity from 10.02 to 8.30 and raises the average zero-shot accuracy from 0.455 to 0.496 relative to a uniform per-layer ratio (Section 3.4). Adaptive allocation is therefore beneficial once the fixed local ratio is removed, which suggests that the limitation observed under N:M sparsity is tied to the pattern rather than to adaptive allocation itself. In perplexity, GroupMask outperforms SparseGPT Frantar and Alistarh (2023), Wanda Sun et al. (2023), LLM-Pruner Ma et al. (2023), SliceGPT Ashkboos et al. (2024), MoDeGPT Lin et al. (2025), and DISP-LLM Gao et al. (2024b) on LLaMA-2-7B; in zero-shot accuracy, with Alpaca calibration, it obtains the highest average among the evaluated baselines on all five LLaMA and Qwen models.

Our contributions are summarized as follows:

- **Layer-adaptive allocation beyond N:M.** To the best of our knowledge, we are the first to learn group-level semi-structured masks for billion-parameter LLMs with frozen pretrained weights. With this pattern, we show that learned layer-adaptive allocation outperforms a uniform per-layer ratio at the same group granularity, in contrast to prior observations under N:M sparsity.
- **GroupMask.** We propose a hypernetwork-based method that generates the group selectors of all layers jointly and learns them with sparsity-budget regularization and self-distillation, without updating the pretrained weights.

- **Empirical evaluation and analysis.** We evaluate GroupMask on five LLaMA and Qwen models with perplexity and six zero-shot benchmarks, and analyze the learned allocation, which retains 36.6% of attention parameters and 56.6% of MLP parameters in LLaMA-2-7B and keeps only about 8% of the query and key groups in layers 22–31.

2 METHODOLOGY

2.1 NOTATION AND GROUP SPARSITY PATTERN

Consider a transformer-based language model, and let $W^l \in \mathbb{R}^{d^l_{\text{out}} \times d^l_{\text{in}}}$, $l = 1, \ldots, L$, denote its L target linear layers, where d^l_{out} and d^l_{in} are the output and input dimensions. To impose a regular sparse structure, we partition each weight matrix into non-overlapping groups of size $g^l_{\text{out}} \times g^l_{\text{in}}$, giving $G^l_{\text{out}} = d^l_{\text{out}}/g^l_{\text{out}}$ and $G^l_{\text{in}} = d^l_{\text{in}}/g^l_{\text{in}}$ groups along the output and input dimensions. Each group is retained or pruned as a whole according to a group-level selector $B^l \in \{0,1\}^{G^l_{\text{out}} \times G^l_{\text{in}}}$, where $B^l_{uv} = 1$ retains group (u, v) and $B^l_{uv} = 0$ prunes it. We refer to each W^l as a linear layer, so layer-adaptive allocation assigns a separate sparsity ratio to each projection (e.g., the query or down projection of a Transformer block) and can vary across both depth and projection type.

The full-resolution binary mask and the corresponding pruned weight matrix are given by

$$M^l = B^l \otimes \mathbf{1}_{g^l_{\text{out}} \times g^l_{\text{in}}}, \qquad \hat{W}^l = W^l \odot M^l, \tag{1}$$

where $\otimes$ denotes the Kronecker product. The layer-wise and overall sparsity ratios are defined as

$$s^l = 1 - \frac{\sum_{u=1}^{G^l_{\text{out}}} \sum_{v=1}^{G^l_{\text{in}}} B^l_{uv}}{G^l_{\text{out}} G^l_{\text{in}}}, \qquad s = \frac{\sum_l s^l d^l_{\text{out}} d^l_{\text{in}}}{\sum_l d^l_{\text{out}} d^l_{\text{in}}}. \tag{2}$$

Compared with element-wise masks, group-level selectors reduce the number of binary decision variables by a factor of $g^l_{\text{out}} g^l_{\text{in}}$, e.g., 256 for the default 1×256 groups.

2.2 HYPERNETWORK-BASED MASK GENERATION

Rather than scoring groups with a fixed heuristic such as weight magnitude, GroupMask generates the selectors with a hypernetwork h_θ. Its input $z = (z_1, \ldots, z_L)$ is a fixed random sequence with one vector per target layer. A bidirectional GRU processes this sequence, and a layer-specific output head maps the GRU output of layer l to $G^l_{\text{out}} G^l_{\text{in}}$ selector logits, which are binarized into

$$B^l = h^l_\theta(z), \qquad B^l \in \{0,1\}^{G^l_{\text{out}} \times G^l_{\text{in}}}. \tag{3}$$

Because B^l is discrete, it cannot be optimized directly by gradient descent. We use a Gumbel-Sigmoid relaxation Jang et al. (2016) with a straight-through estimator (STE) Bengio et al. (2013): the forward pass uses binary selectors, and the backward pass propagates gradients through the continuous relaxation to θ. At evaluation time, the Gumbel noise is removed and the selectors are obtained by thresholding. The selectors are expanded and applied to the pretrained weights according to Eq. (1). Since the selectors of all layers are produced by one network and trained under a single global budget, the per-layer sparsity ratios are determined jointly rather than fixed in advance.

2.3 SPARSITY BUDGET REGULARIZATION

To enforce a target sparsity s_t, we regularize the global sparsity ratio s defined in Eq. (2) with a log-ratio objective $\mathcal{L}_{\text{reg}} = \lambda \, |\log(s/s_t)|$, where λ controls the regularization strength. Unlike distance-based penalties such as the squared error, this objective depends only on the ratio s/s_t and assigns the same penalty to $s = c\, s_t$ and $s = s_t/c$. It is computed on the discrete selectors B^l, so it reflects the exact sparsity of the applied masks. Because $\mathcal{L}_{\text{reg}}$ constrains only the global ratio s, the layer-wise ratios s^l may deviate from s_t, which is what allows layer-adaptive allocation. The uniform variant in Section 3.4 instead applies the same regularizer to each s^l separately.

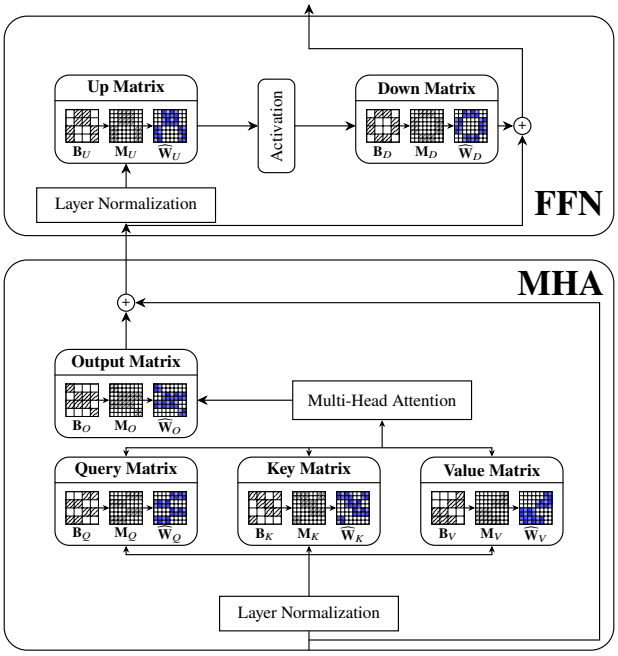


Figure 2: Structural configuration of our group-sparse pruning framework within a single Transformer block. For each linear projection, the mask generation module yields a structural group selector matrix B, which is expanded into a full-resolution binary mask M to determine the pruned weight $\hat{W}^l$ via element-wise multiplication.

2.4 Self-Distillation Optimization Framework

Fig. 2 illustrates the overall optimization framework. The hypernetwork first generates group-level selector matrices B^l for the target projections, which are expanded into full-resolution masks and applied to the pretrained weights according to Eq. (1). The resulting sparse model is optimized through self-distillation while the pretrained weights remain frozen.

Unlike conventional knowledge distillation with separate teacher and student models, our framework uses a single pretrained model with dynamically switched sparsity masks. The teacher forward pass uses the dense model by setting all selector matrices to $B^l = \mathbf{1}$, whereas the student forward pass applies the generated selectors B^l to obtain the sparse model. Since the pretrained weights remain frozen throughout training, gradients are propagated only to the hypernetwork parameters θ.

To preserve the predictive behavior of the dense model, we minimize the cross-entropy between the teacher and student output distributions:

$$\mathcal{L}_{\mathrm{KD}} = -\frac{1}{N}\sum_{t=1}^{N}\sum_{v\in\mathcal{V}} p_T^{(t,v)} \log p_S^{(t,v)}, \tag{4}$$

where N denotes the number of valid tokens and $\mathcal{V}$ denotes the vocabulary. This objective is equivalent to minimizing the forward KL divergence from the teacher distribution to the student distribution up to an additive constant:

$$\mathrm{KL}(p_T \,\|\, p_S) = \underbrace{-\sum_v p_T(v)\log p_S(v)}_{\text{cross-entropy}} - \underbrace{\left(-\sum_v p_T(v)\log p_T(v)\right)}_{H(p_T)}.$$

The entropy term $H(p_T)$ depends only on the teacher distribution and is constant with respect to the student. Therefore, minimizing $\mathcal{L}_{\mathrm{KD}}$ is equivalent to minimizing the forward KL divergence without explicitly computing this constant term.

The final training objective combines self-distillation and sparsity regularization:

$$\mathcal{L}_{\mathrm{total}} = \mathcal{L}_{\mathrm{KD}} + \mathcal{L}_{\mathrm{reg}}. \tag{5}$$

Together, these objectives guide GroupMask to preserve the predictive behavior of the dense model while satisfying the target sparsity constraint.

3 EXPERIMENTS

We conduct experiments to answer five research questions for GroupMask: (1) Can GroupMask preserve language modeling quality after pruning? (2) Does GroupMask generalize across different LLM families and downstream tasks? (3) Does layer-adaptive allocation outperform a uniform per-layer ratio under the same group pattern? (4) How does group granularity affect pruning quality? (5) What is the contribution of the hypernetwork-based mask generator?

3.1 SETTINGS

Models. We evaluate LLaMA-2-7B, LLaMA-2-13B, LLaMA-3-8B, Qwen3-8B, and Qwen3-14B, which cover two model families and 7B to 14B parameters.

Implementation. We implement our method with PyTorch Paszke et al. (2019) and Hugging Face Transformers Wolf et al. (2020). The pretrained weights are frozen, and no post-pruning fine-tuning is applied. Unless stated otherwise, we use 1×256 groups, 50% sparsity, and WikiText-2 calibration. For zero-shot evaluation, we additionally learn masks with Alpaca calibration; the two settings are denoted GroupMask-Wiki and GroupMask-Alpaca. Further details are given in Appendix A.

Datasets. We use WikiText Merity et al. (2016) and Alpaca Taori et al. (2023) as calibration data. We measure language modeling quality by perplexity on WikiText-2, and zero-shot performance on six benchmarks Gao et al. (2024a): ARC-Challenge Clark et al. (2018), ARC-Easy Clark et al. (2018), BoolQ Clark et al. (2019), HellaSwag Zellers et al. (2019), PIQA Bisk et al. (2019), and WinoGrande Sakaguchi et al. (2019). The average zero-shot score is the mean over the six tasks.

Baselines. We compare with the one-shot 2:4 methods SparseGPT Frantar and Alistarh (2023) and Wanda Sun et al. (2023), and with the structured methods LLM-Pruner Ma et al. (2023), SliceGPT Ashkboos et al. (2024), MoDeGPT Lin et al. (2025), and DISP-LLM Gao et al. (2024b), all at 50% sparsity. We do not include MaskLLM Fang et al. (2024): its reported LLaMA-2-7B result uses 512k training samples and 1,280 A100 GPU-hours, whereas GroupMask uses about 3.7 GPU-hours (Table 5), so the two methods operate at compute budgets that are not comparable.

3.2 LANGUAGE MODELING PERFORMANCE

Table 1 reports WikiText-2 perplexity at 50% sparsity. On LLaMA-2-7B, GroupMask with 1×256 groups obtains the lowest perplexity among the compared methods, 8.30 compared with 9.84 for DISP-LLM, a 15.7% relative reduction. On LLaMA-2-13B, it obtains 7.19, second to DISP-LLM (7.11). Structured methods yield smaller dense matrices that run without sparse kernels, so this comparison is made at equal parameter sparsity rather than at equal inference cost. The 1×256 configuration outperforms 32×32 on both models (8.30 vs. 9.37 and 7.19 vs. 7.79). A 1×256 group contains 256 weights, whereas a 32×32 group contains 1,024, so the former makes four times as many selection decisions per matrix; Section 3.5 examines group granularity further.

Table 1: WikiText-2 perplexity (↓) of LLaMA-2 models at 50% sparsity.

Method	Sparsity type	W Update	7B	13B
Dense	–	–	5.12	4.57
LLM-Pruner	Structured	✗	109.43	49.71
SliceGPT	Structured	✓	24.82	20.57
MoDeGPT	Structured	✓	11.88	8.95
DISP-LLM	Structured	✗	9.84	**7.11**
SparseGPT	2:4	✓	10.17	8.32
Wanda	2:4	✗	11.02	8.27
GroupMask (1×256)	Group	✗	**8.30**	<u>7.19</u>
GroupMask (32×32)	Group	✗	<u>9.37</u>	7.79

3.3 ZERO-SHOT DOWNSTREAM PERFORMANCE

Table 2 reports zero-shot accuracy at 50% sparsity; all GroupMask results use 1×256 groups. With Alpaca calibration, GroupMask obtains the highest average accuracy among the evaluated baselines on all five models. On LLaMA-2-7B, LLaMA-2-13B, and LLaMA-3-8B, its averages of 0.523, 0.577, and 0.517 exceed the strongest baseline, MoDeGPT, by 3.0, 2.6, and 2.0 percentage

Table 2: Zero-shot downstream performance under 50% sparsity (↑).

Model	Method	ARC-C acc-norm	ARC-E acc-norm	BoolQ acc	Hella acc-norm	PIQA acc-norm	Wino acc	**Avg**
LLaMA-2-7B	Dense	0.462	0.745	0.778	0.760	0.788	0.692	0.704
	LLM-Pruner* Ma et al. (2023)	0.239	0.304	0.454	0.276	0.525	0.499	0.383
	MoDeGPT Lin et al. (2025)	0.293	0.375	0.621	0.458	0.610	0.604	0.493
	DISP-LLM Gao et al. (2024b)	0.260	0.393	0.629	0.433	0.607	0.529	0.475
	SliceGPT Ashkboos et al. (2024)	0.237	0.348	0.378	0.325	0.536	0.531	0.393
	GroupMask-Wiki	0.261	0.462	0.588	0.472	0.644	0.550	0.496
	GroupMask-Alpaca	0.306	0.542	0.547	0.496	0.670	0.577	**0.523**
LLaMA-2-13B	Dense	0.492	0.775	0.805	0.794	0.805	0.721	0.732
	LLM-Pruner* Ma et al. (2023)	0.240	0.343	0.481	0.309	0.577	0.493	0.407
	MoDeGPT Lin et al. (2025)	0.325	0.487	0.649	0.537	0.630	0.680	0.551
	DISP-LLM Gao et al. (2024b)	0.323	0.502	0.639	0.530	0.641	0.572	0.534
	SliceGPT Ashkboos et al. (2024)	0.249	0.370	0.378	0.344	0.555	0.555	0.409
	GroupMask-Wiki	0.312	0.523	0.623	0.558	0.676	0.587	0.546
	GroupMask-Alpaca	0.355	0.620	0.631	0.571	0.707	0.578	**0.577**
LLaMA-3-8B	Dense	0.533	0.777	0.814	0.792	0.808	0.729	0.742
	MoDeGPT Lin et al. (2025)	0.315	0.453	0.669	0.410	0.628	0.509	0.497
	DISP-LLM Gao et al. (2024b)	0.279	0.432	0.556	0.468	0.639	0.553	0.488
	SliceGPT Ashkboos et al. (2024)	0.211	0.303	0.378	0.297	0.517	0.503	0.368
	GroupMask-Wiki	0.261	0.439	0.622	0.430	0.635	0.559	0.491
	GroupMask-Alpaca	0.297	0.524	0.618	0.453	0.663	0.545	**0.517**
Qwen3-8B	Dense	0.564	0.809	0.866	0.749	0.777	0.676	0.740
	MoDeGPT Lin et al. (2025)	0.263	0.428	0.594	0.350	0.607	0.504	0.457
	DISP-LLM Gao et al. (2024b)	0.316	0.500	0.624	0.459	0.631	0.559	0.515
	SliceGPT Ashkboos et al. (2024)	0.229	0.327	0.378	0.331	0.545	0.534	0.391
	GroupMask-Wiki	0.290	0.498	0.628	0.442	0.630	0.571	0.510
	GroupMask-Alpaca	0.359	0.612	0.657	0.469	0.671	0.563	**0.555**
Qwen3-14B	Dense	0.602	0.828	0.894	0.788	0.798	0.729	0.773
	MoDeGPT Lin et al. (2025)	0.275	0.447	0.636	0.368	0.620	0.532	0.480
	DISP-LLM Gao et al. (2024b)	0.365	0.570	0.453	0.551	0.669	0.616	0.537
	SliceGPT Ashkboos et al. (2024)	0.235	0.349	0.378	0.355	0.553	0.576	0.408
	GroupMask-Wiki	0.312	0.547	0.640	0.500	0.647	0.587	0.539
	GroupMask-Alpaca	0.386	0.628	0.708	0.521	0.698	0.592	**0.589**

*LLM-Pruner is reported only on LLaMA-2 models because its available implementation does not directly support the LLaMA-3 and Qwen3 architectures evaluated here.

Table 3: Layer-adaptive versus uniform sparsity allocation on LLaMA-2-7B at 50% sparsity. Both settings use the same 1×256 group structure, WikiText-2 calibration, and training schedule; the uniform setting constrains every linear layer to $s^l = 50\%$.

Allocation	PPL ↓	ARC-C	ARC-E	BoolQ	Hella	PIQA	Wino	Avg ↑
Uniform	10.02	0.238	0.376	**0.622**	0.378	0.596	0.522	0.455
Adaptive	**8.30**	**0.261**	**0.462**	0.588	**0.472**	**0.644**	**0.550**	**0.496**

points. On Qwen3-8B and Qwen3-14B, its averages of 0.555 and 0.589 exceed the strongest baseline, DISP-LLM, by 4.0 and 5.2 points. With WikiText-2 calibration, GroupMask obtains the highest average on LLaMA-2-7B and Qwen3-14B and is within 0.6 points of the strongest baseline on the other three models. Alpaca calibration yields a higher average than WikiText-2 calibration on all five models, with the largest difference on Qwen3-14B (0.539 to 0.589).

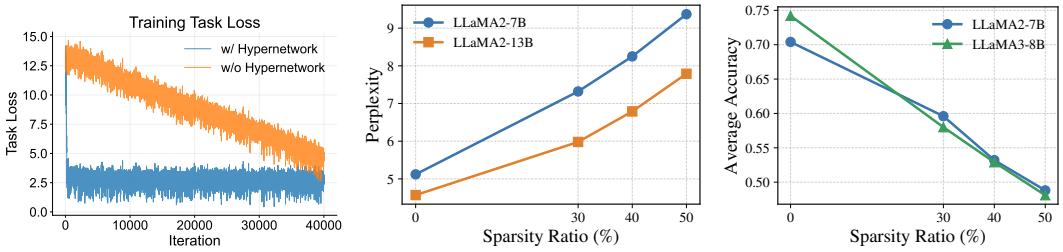
3.4 Layer-Adaptive versus Uniform Allocation

To isolate the effect of layer-adaptive allocation, we compare GroupMask with a uniform variant that constrains every target linear layer to $s^l = 50\%$, while keeping the group structure, calibration data, and training schedule unchanged. As shown in Table 3, adaptive allocation reduces WikiText-2 perplexity from 10.02 to 8.30 and raises the average zero-shot accuracy from 0.455 to 0.496, improving five of the six tasks; BoolQ decreases from 0.622 to 0.588. Since the two settings differ only in whether the per-layer ratio is fixed or learned, the gap can be attributed to layer-adaptive allocation under the group-level pattern. This result contrasts with the observation of Liu et al. (2025) that non-uniform layer-wise ratios become less effective under N:M sparsity. The non-uniform ratios evaluated there differ from ours in two respects: they are derived from heuristics such as OWL Yin

Table 4: Zero-shot ablation on LLaMA-2-7B at 50% sparsity (↑).

Setting	Cal.	ARC-C	ARC-E	BoolQ	Hella.	PIQA	Wino.	Avg.
Dense	–	0.462	0.745	0.778	0.760	0.788	0.692	0.704
Hypernetwork ablation								
w/o HN	Wiki	0.240	0.322	0.478	0.314	0.552	0.511	0.403
	Alp.	0.242	0.283	0.528	0.269	0.522	0.479	0.387
Group granularity and calibration								
1×128	Wiki	0.269	0.441	0.631	0.469	0.642	0.560	0.502
	Alp.	0.292	0.551	0.626	0.486	0.680	0.537	**0.528**
1×256	Wiki	0.261	0.462	0.588	0.472	0.644	0.550	0.496
	Alp.	0.306	0.542	0.547	0.496	0.670	0.577	<u>0.523</u>
16×16	Wiki	0.269	0.446	0.604	0.471	0.656	0.549	0.499
	Alp.	0.288	0.524	0.628	0.459	0.669	0.542	0.518
32×32	Wiki	0.270	0.440	0.573	0.464	0.653	0.526	0.488
	Alp.	0.279	0.481	0.624	0.465	0.656	0.541	0.508
64×64	Wiki	0.278	0.460	0.610	0.437	0.630	0.535	0.492
	Alp.	0.271	0.476	0.602	0.453	0.655	0.541	0.499
128×128	Wiki	0.232	0.405	0.554	0.437	0.599	0.493	0.453
	Alp.	0.247	0.435	0.624	0.433	0.620	0.541	0.483

et al. (2024), and OWL assigns one ratio per Transformer block, whereas GroupMask learns a ratio for each projection. The comparison therefore suggests, but does not by itself establish, that the fine-grained N:M constraint is what limits the benefit of non-uniform ratios. Section 3.5 shows how the learned allocation departs from a uniform ratio: about half of its variation is across projection types, which block-level ratios cannot express, and deep attention projections receive much smaller budgets than MLP projections.



(a) Hypernetwork optimization (b) Perplexity (c) Zero-shot accuracy

Figure 3: Ablation analysis of GroupMask. (a) Effect of the hypernetwork on mask optimization. (b–c) Perplexity and zero-shot performance under increasing sparsity.

3.5 Ablation Study

We examine four design choices: (1) the hypernetwork, (2) group granularity, (3) calibration data, and (4) the sparsity ratio. We then analyze the allocation learned by GroupMask.

Effect of the hypernetwork. We compare mask learning with and without the hypernetwork on LLaMA-2-7B. In the setting without a hypernetwork, the selector logits are learnable parameters initialized from a standard normal distribution and passed through the same Gumbel-Sigmoid relaxation and STE, with the same regularizer and distillation objective. With the hypernetwork, the task loss is lower throughout training (Figure 3a), and removing it reduces the average zero-shot accuracy from 0.496 to 0.403 with WikiText-2 calibration and from 0.523 to 0.387 with Alpaca calibration (Table 4). The corresponding sparsity regularization loss is shown in Appendix D.

Group granularity and calibration data. As shown in Table 4, average accuracy tends to decrease as each group contains more weights. With WikiText-2 calibration, it decreases from 0.502 for 1×128 to 0.453 for 128×128, although the trend is not strictly monotonic (32×32: 0.488; 64×64: 0.492). With Alpaca calibration, 1×128 obtains the highest average (0.528), followed by the default

1×256 (0.523). For every group shape in Table 4, Alpaca calibration yields a higher average than WikiText-2 calibration, e.g., 0.496 to 0.523 for 1×256.

Effect of sparsity ratio. Figures 3b and 3c report perplexity and zero-shot accuracy at 30%, 40%, and 50% sparsity with 32×32 groups and WikiText-2 calibration. From 30% to 50% sparsity, the perplexity of LLaMA-2-7B increases from 7.32 to 9.37 and its average accuracy decreases from 0.596 to 0.488; the average accuracy of LLaMA-3-8B decreases from 0.580 to 0.481 (Appendix B).

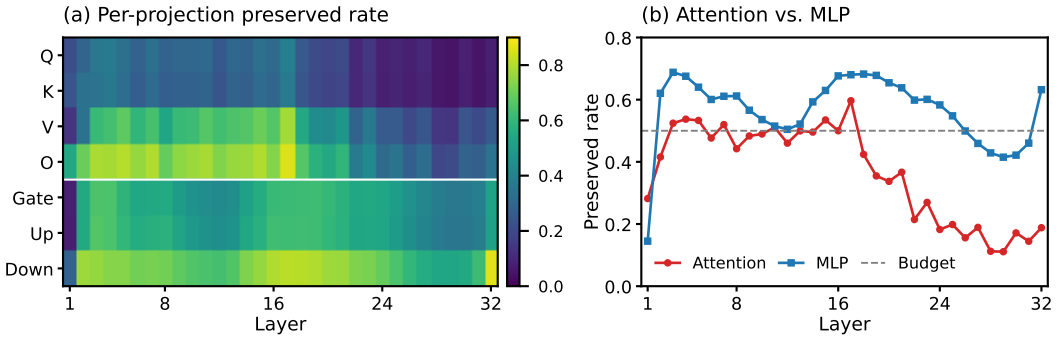


Figure 4: Learned preserved rates of LLaMA-2-7B at 50% sparsity. (a) Preserved rate of each projection across layers. (b) Average preserved rates of attention and MLP projections; the dashed line marks the global 50% budget.

Kernel-level efficiency. Group-level sparsity has no native hardware support, so we implement a preliminary group-sparse SpMM kernel with CUDA/CUTLASS on an NVIDIA H100 using Hopper's Tensor Memory Accelerator (TMA), and compare it with dense cuBLAS in FP32 on a LLaMA-2 gate projection ($M = 11008$, $K = 4096$, $N = 65536$). With group size 64, the kernel reaches $1.09\times$ speedup at 50% sparsity and $1.88\times$ at 90% (Figure 5c); at 50% sparsity, changing the group size has little effect on speedup (Figure 5b). These are kernel-level measurements; end-to-end acceleration of group-sparse LLMs requires further system support, which we leave to future work.

Analysis of learned pruning structures. Figure 4 shows the preserved rate of each projection in the exported LLaMA-2-7B model. The global budget is met (49.98% of parameters preserved), but the allocation is non-uniform: attention projections retain 36.6% of their parameters and MLP projections 56.6%, and the down projection accounts for 30.5% of all preserved parameters. Three regularities appear. First, functionally coupled projections receive nearly identical budgets: across layers, the preserved rates of Q and K, and those of Gate and Up, have a Pearson correlation of 0.994 and differ by less than 0.05. Second, within attention, V retains about 2.2 times as many parameters as Q or K, and O retains the most, so the learned masks prune the query–key pathway more heavily than the value–output pathway. Third, the allocation varies with depth: in layers 3–17, attention and MLP receive comparable budgets, whereas in layers 22–31, Q and K keep about 8% of their groups while the down projection keeps more than 53%. This depth-dependent reduction of attention is consistent with reports that many deep attention layers in LLMs can be removed with limited quality loss (He et al., 2024; Men et al., 2025). In GroupMask, the pattern is learned from the distillation objective rather than specified by a pruning rule, and it is expressed per projection rather than per Transformer block. Across the 224 projections in Figure 4, projection type accounts for about half of the variance in preserved rates and depth for about one third, and roughly 40% of all target parameters lie in projections pruned beyond 50% sparsity, which neither 2:4 sparsity nor PATCH Hourri et al. (2025) can represent.

4 Related Work

Pruning for LLMs. Unstructured pruning removes individual weights and preserves model quality well LeCun et al. (1989); Hassibi and Stork (1992); Frantar and Alistarh (2023); Sun et al. (2023), but its irregular sparsity limits practical acceleration Fan et al. (2025). Structured pruning removes channels, hidden dimensions, attention heads, or layers Ma et al. (2023); Ashkboos et al. (2024); Men et al. (2025); Gao et al. (2024b), yielding dense models that run efficiently on standard hardware at the cost of coarser pruning decisions. Post-training methods such as SparseGPT and Wanda show

that LLMs can be pruned without full retraining Frantar and Alistarh (2023); Sun et al. (2023), while distillation-guided pruning replaces recovery fine-tuning with supervision from the dense model Hinton et al. (2015); Xia et al. (2022).

Semi-structured sparsity. N:M sparsity retains N non-zero weights in every M consecutive weights, and the 2:4 case is natively accelerated on NVIDIA Ampere and later GPUs Mishra et al. (2021); Pool and Yu (2021); Zhou et al. (2021). Learnable N:M methods optimize the mask end-to-end, either through Gumbel-softmax sampling over candidate patterns Fang et al. (2024) or through proximal regularization Liu et al. (2025). PATCH Hourri et al. (2025) partitions weights into tiles and makes each tile either dense or 2:4 sparse, which lets the sparsity ratio vary across layers; since each tile is at most 50% sparse, however, no layer can exceed 50% sparsity, and at a 50% global budget the pattern reduces to uniform 2:4 sparsity. Block pruning has been studied for BERT-scale encoders, where block masks are learned by movement pruning together with weight updates during task-specific fine-tuning Lagunas et al. (2021), and GPU kernels for block-sparse weights were developed earlier Gray et al. (2017). GroupMask instead learns group-level masks for billion-parameter decoder-only LLMs in a task-agnostic, post-training setting with frozen weights, and generates the masks of all layers jointly, so that each layer's sparsity ratio can fall above or below the global budget.

Layer-wise sparsity allocation. For unstructured pruning, non-uniform layer-wise sparsity improves over a uniform ratio, whether the ratios are derived from outlier statistics Yin et al. (2024) or learned Xu et al. (2024).OWL assigns one ratio per Transformer block and reports that assigning ratios to individual layers degrades performance. Under N:M sparsity, Liu et al. (2025) observe that non-uniform layer-wise ratios become less effective, because critical weights may still be removed within each block. We examine this discrepancy by comparing adaptive and uniform allocation under the same group-level pattern.

Learnable masks. Learnable pruning methods parameterize pruning decisions and optimize them with gradient-based relaxations such as movement scores Sanh et al. (2020), Gumbel-based sampling Chang et al. (2019), or decaying masks Kao et al. (2022). DISP-LLM Gao et al. (2024b) uses a hypernetwork to generate structural pruning decisions for LLMs. GroupMask follows this line by generating the group-level selectors of all layers with a shared hypernetwork, trained through self-distillation while the pretrained weights stay frozen.

5 Conclusion

We revisited layer-adaptive sparsity allocation for semi-structured LLM pruning, which prior work found to be less effective under the N:M pattern. With group-level sparsity, which keeps a regular structure but leaves each layer's sparsity ratio free, learned adaptive allocation reduces WikiText-2 perplexity from 10.02 to 8.30 relative to a uniform ratio under the same group structure on LLaMA-2-7B, suggesting that the earlier observation is tied to the N:M pattern. GroupMask learns this allocation with a hypernetwork-generated group mask, sparsity-budget regularization, and self-distillation, without updating the pretrained weights, and with Alpaca calibration it obtains the highest average zero-shot accuracy among the evaluated baselines on five LLaMA and Qwen models. The learned masks remove most query and key groups in deep layers while retaining more of the MLP projections. **Limitations.** First, the comparison with N:M sparsity is indirect: the non-uniform ratios evaluated under N:M are heuristic and assigned per Transformer block, whereas ours are learned per projection; a heuristic allocation under group-level sparsity would further separate the two factors. Second, the adaptive-versus-uniform comparison is conducted on LLaMA-2-7B only. Third, the highest zero-shot averages are obtained with Alpaca calibration; with WikiText-2 calibration, GroupMask is best on two of the five models. Finally, group-level sparsity currently lacks native hardware support, our kernel measurements are preliminary, and our experiments cover sparsity of at most 50% and models of up to 14B parameters; end-to-end acceleration and co-design with sparse kernels remain future work.

AI USE STATEMENT

Generative AI tools were used during this work to assist with research methodology and experimental design discussions, software development and debugging, analysis and interpretation of experimental results, and preparation of the manuscript. They were also used for drafting and editing text, improving clarity and organization, formatting LaTeX tables and equations, and providing feedback on figures and presentation. All AI-assisted research decisions, implementations, experimental results, and manuscript content were reviewed and verified by the authors. Experimental results reported in this paper were obtained from our implementations and experiments rather than generated by AI tools. The authors take full responsibility for the methodology, results, claims, and final content of the paper.

REPRODUCIBILITY STATEMENT

We provide implementation and experimental details to support reproducibility. The methodology section specifies the group sparsity formulation, mask generation, sparsity regularization, and self-distillation objective, while the appendix reports training hyperparameters, calibration settings, additional results, mask export procedures, and sparse-kernel evaluation details. Our implementation is available in the anonymous repository linked in the abstract.

ETHICS STATEMENT

This work studies model compression for large language models and does not involve human subjects or the collection of personal data. Our experiments use publicly available pretrained models and benchmark datasets.

REFERENCES

Saleh Ashkboos, Maximilian L Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. Slicegpt: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*, 2024.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language, 2019. URL https://arxiv.org/abs/1911.11641.

Jianlong Chang, Xinbang Zhang, Yiwen Guo, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Differentiable architecture search with ensemble gumbel-softmax. *arXiv preprint arXiv:1905.01786*, 2019.

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

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. *arXiv:1803.05457v1*, 2018.

Ruibo Fan, Xiangrui Yu, Peijie Dong, Zeyu Li, Gu Gong, Qiang Wang, Wei Wang, and Xiaowen Chu. Spinfer: Leveraging low-level sparsity for efficient large language model inference on gpus. In *Proceedings of the Twentieth European Conference on Computer Systems*, pages 243–260, 2025.

Gongfan Fang, Hongxu Yin, Saurav Muralidharan, Greg Heinrich, Jeff Pool, Jan Kautz, Pavlo Molchanov, and Xinchao Wang. Maskllm: Learnable semi-structured sparsity for large language models. *Advances in Neural Information Processing Systems*, 37:7736–7758, 2024.

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

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

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024a. URL https://zenodo.org/records/12608602.

Shangqian Gao, Chi-Heng Lin, Ting Hua, Tang Zheng, Yilin Shen, Hongxia Jin, and Yen-Chang Hsu. Disp-llm: Dimension-independent structural pruning for large language models. *Advances in Neural Information Processing Systems*, 37:72219–72244, 2024b.

Scott Gray, Alec Radford, and Diederik P. Kingma. GPU kernels for block-sparse weights. Technical report, OpenAI, 2017.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015a.

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, 2015b.

Babak Hassibi and David Stork. Second order derivatives for network pruning: Optimal brain surgeon. In S. Hanson, J. Cowan, and C. Giles, editors, *Advances in Neural Information Processing Systems*, volume 5. Morgan-Kaufmann, 1992. URL https://proceedings.neurips.cc/paper_files/paper/1992/file/303ed4c69846ab36c2904d3ba8573050-Paper.pdf.

Shwai He, Guoheng Sun, Zheyu Shen, and Ang Li. What matters in transformers? not all attention is needed, 2024. URL https://arxiv.org/abs/2406.15786.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Younes Hourri, Mohammad Mozaffari, and Maryam Mehri Dehnavi. Patch: Learnable tile-level hybrid sparsity for llms. *arXiv preprint arXiv:2509.23410*, 2025.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Sheng-Chun Kao, Amir Yazdanbakhsh, Suvinay Subramanian, Shivani Agrawal, Utku Evci, and Tushar Krishna. Training recipe for n: M structured sparsity with decaying pruning mask. *arXiv preprint arXiv:2209.07617*, 2022.

François Lagunas, Ella Charlaix, Victor Sanh, and Alexander M. Rush. Block pruning for faster transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021.

Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989. URL https://proceedings.neurips.cc/paper_files/paper/1989/file/6c9882bbac1c7093bd25041881277658-Paper.pdf.

Chi-Heng Lin, Shangqian Gao, James Seale Smith, Abhishek Patel, Shikhar Tuli, Yilin Shen, Hongxia Jin, and Yen-Chang Hsu. ModeGPT: Modular decomposition for large language model compression. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=8EfxjTCg2k.

Hongyi Liu, Rajarshi Saha, Zhen Jia, Youngsuk Park, Jiaji Huang, Shoham Sabach, Yu-Xiang Wang, and George Karypis. Proxsparse: Regularized learning of semi-structured sparsity masks for pretrained llms. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025. URL https://arxiv.org/abs/2502.00258.

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

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720, 2023.

Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20192–20204, 2025.

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

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

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. URL https://arxiv.org/abs/1912.01703.

Jeff Pool and Chong Yu. Channel permutations for n:m sparsity. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 13316–13327. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/6e8404c3b93a9527c8db241a1846599a-Paper.pdf.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019. URL https://arxiv.org/abs/1907.10641.

Victor Sanh, Thomas Wolf, and Alexander Rush. Movement pruning: Adaptive sparsity by fine-tuning. *Advances in neural information processing systems*, 33:20378–20389, 2020.

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

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. Svd-llm: Truncation-aware singular value decomposition for large language model compression. In *International Conference on Learning Representations*, volume 2025, pages 19299–19319, 2025.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. URL https://aclanthology.org/2020.emnlp-demos.6/.

Mengzhou Xia, Zexuan Zhong, and Danqi Chen. 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, 2022.

Peng Xu, Wenqi Shao, Mengzhao Chen, Shitao Tang, Kaipeng Zhang, Peng Gao, Fengwei An, Yu Qiao, and Ping Luo. Besa: Pruning large language models with blockwise parameter-efficient sparsity allocation. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2402.16880.

Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Gen Li, Ajay Jaiswal, Mykola Pechenizkiy, Yi Liang, Michael Bendersky, Zhangyang Wang, and Shiwei Liu. Outlier weighed layerwise sparsity (owl): A missing secret sauce for pruning llms to high sparsity. In *International Conference on Machine Learning*, 2024. URL https://arxiv.org/abs/2310.05175.

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

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2):1–124, 2023.

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

A Implementation Details

This section provides additional implementation details for reproducing our experiments. All experiments are implemented with PyTorch and Hugging Face Transformers. For each pruned model, we keep the original pretrained LLM weights frozen and optimize only the hypernetwork-based mask generator. No LoRA adaptation, recovery fine-tuning, or post-pruning weight update is applied.

Training objective. We use the same objective as described in Sec. 2, and provide additional implementation details here. During mask learning, the dense pretrained model is used as the teacher, and the masked model is used as the student. The teacher logits are computed with pruning masks disabled and are detached from the computation graph. The student logits are computed with group-sparse masks enabled. The pretrained weights remain frozen throughout this process, and only the hypernetwork parameters used to generate group-level masks are updated.

Calibration data. We use WikiText and Alpaca as calibration datasets for learning pruning structures. The two calibration settings are denoted as GroupMask-Wiki and GroupMask-Alpaca, respectively. Unless otherwise specified, the calibration dataset seed is fixed to 42. Input sequences are tokenized by the corresponding model tokenizer, and the maximum calibration sequence length is set to 2048.

Optimization hyperparameters. Table 5 summarizes the major hyperparameters used in our experiments. For experiments with different sparsity ratios, we only change the target sparsity parameter while keeping the other hyperparameters unchanged. For group-shape ablations, we only change the group dimensions and keep the optimization setting fixed.

Table 5: Main hyperparameters used for hypernetwork-based mask learning.

Hyperparameter	Value
Optimizer	AdamW / AdamW-8bit
Learning rate	1×10^{-3}
Weight decay	0.05
Adam β	$(0.9, 0.999)$
Batch size	1
Sequence length	2048
Training steps	40,000
Training time (LLaMA-2-7B)	$\sim$3.7 GPU-hours
Training hardware	1 $\times$ NVIDIA DGX B200
Checkpoint interval	10,000
Calibration seed	42
Temperature T	0.4
Regularization coefficient λ	16
Default sparsity ratio	50%
Default group shape	1×256
Precision	bfloat16
FSDP	Enabled when needed
Post-pruning finetuning	No
LoRA adaptation	No

Group shapes and sparsity ratios. The main experiments use the default group configuration unless otherwise specified. For group-granularity analysis, we additionally evaluate 16×16, 1×128, 1×256, 32×32, 64×64, and 128×128 group structures. For sparsity sensitivity analysis, we evaluate 30%, 40%, and 50% sparsity. The target sparsity is controlled by the mask regularization objective, while the original model weights remain unchanged throughout the pruning process.

Evaluation protocol. We evaluate language modeling quality using perplexity on WikiText-2. For zero-shot downstream evaluation, we report accuracy on ARC-Challenge, ARC-Easy, BoolQ, HellaSwag, PIQA, and WinoGrande, and compute the average score across the six tasks. All reported zero-shot results are obtained without task-specific finetuning.

B Additional Sparsity Sensitivity Results

Table 6 reports zero-shot results at 30% and 40% sparsity with 32×32 groups and WikiText-2 calibration. Together with the 50% results under the same setting (Tables 4 and 8), the average

Table 6: Zero-shot downstream performance under different sparsity ratios. We report results on six benchmarks and the average score.

Model	Sparsity	ARC-C	ARC-E	BoolQ	Hella.	PIQA	Wino.	Avg ↑
LLaMA-2-7B	30%	0.3481	0.6124	0.6606	0.6203	0.7258	0.6101	0.596
	40%	0.302	0.5109	0.5829	0.5394	0.6817	0.5738	0.532
LLaMA-3-8B	30%	0.3396	0.5934	0.6602	0.5913	0.7171	0.5809	0.580
	40%	0.2969	0.5004	0.6434	0.5092	0.6741	0.5509	0.529

accuracy of both LLaMA-2-7B and LLaMA-3-8B decreases monotonically as sparsity increases from 30% to 50%.

Table 7: WikiText-2 perplexity under different sparsity ratios using the 32×32 group setting. Lower values indicate better language modeling performance.

Sparsity	LLaMA-2-7B ↓	LLaMA-2-13B ↓
30%	7.32	5.98
40%	8.25	6.79
50%	9.37	7.79

Table 7 reports WikiText-2 perplexity under the same setting. For both LLaMA-2-7B and LLaMA-2-13B, perplexity increases monotonically as sparsity increases from 30% to 50%, consistent with the zero-shot results.

Table 8: Additional zero-shot downstream results under square group configurations.

Group	Cal.	ARC-C	ARC-E	BoolQ	Hella.	PIQA	Wino.	Avg ↑
LLaMA-2-13B								
32×32	Wiki	0.3131	0.5640	0.6336	0.5621	0.6986	0.5691	0.557
32×32	Alpaca	0.3311	0.5800	0.6309	0.5486	0.7029	0.5801	0.562
64×64	Wiki	0.3131	0.5505	0.5728	0.5600	0.6986	0.5762	0.545
64×64	Alpaca	0.3217	0.5703	0.6330	0.5505	0.6855	0.5833	0.557
128×128	Wiki	0.3080	0.5122	0.5982	0.5654	0.6806	0.5706	0.539
128×128	Alpaca	0.3379	0.5319	0.6232	0.5592	0.6997	0.5643	0.553
LLaMA-3-8B								
32×32	Wiki	0.2560	0.4415	0.6135	0.4143	0.6257	0.5351	0.481
32×32	Alpaca	0.2824	0.4954	0.5251	0.4283	0.6632	0.5335	0.488
64×64	Wiki	0.2432	0.4108	0.6217	0.3933	0.6104	0.5588	0.473
64×64	Alpaca	0.2688	0.4461	0.5309	0.4027	0.6295	0.5454	0.471
128×128	Wiki	0.2457	0.3864	0.6217	0.3775	0.6012	0.5422	0.462
128×128	Alpaca	0.2509	0.3956	0.6217	0.3713	0.6181	0.5272	0.464
Qwen3-8B								
32×32	Wiki	0.3148	0.5227	0.6220	0.4214	0.6534	0.5288	0.511
32×32	Alpaca	0.2918	0.5105	0.6009	0.5480	0.6458	0.5114	0.518
64×64	Wiki	0.2961	0.4726	0.6220	0.4167	0.6415	0.5296	0.496
64×64	Alpaca	0.3183	0.5303	0.5110	0.4141	0.6545	0.5391	0.495
128×128	Wiki	0.2509	0.4251	0.6232	0.3870	0.6186	0.5328	0.473
128×128	Alpaca	0.2944	0.4874	0.6162	0.3983	0.6447	0.5280	0.495
Qwen3-14B								
32×32	Wiki	0.3270	0.5644	0.5214	0.5038	0.6719	0.5706	0.527
32×32	Alpaca	0.3532	0.6090	0.6682	0.4947	0.6850	0.5588	0.561
64×64	Wiki	0.3305	0.5535	0.4960	0.4896	0.6529	0.5549	0.513
64×64	Alpaca	0.3575	0.6157	0.5135	0.4809	0.6746	0.5659	0.535
128×128	Wiki	0.3012	0.5021	0.6211	0.4542	0.6540	0.5406	0.512
128×128	Alpaca	0.3140	0.5400	0.6349	0.4647	0.6638	0.5359	0.526

C SQUARE GROUP GRANULARITY

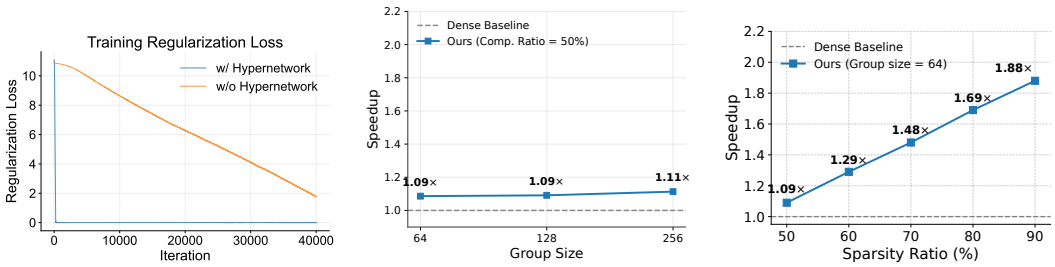
We provide additional zero-shot results for square group configurations in Table 8. These results complement the main ablation study by evaluating 32×32, 64×64, and 128×128 group shapes across multiple model families under both WikiText-2 and Alpaca calibration.

Overall, the results show that smaller square groups generally provide better pruning quality than coarser square groups. For LLaMA-2-13B, the best square-group result is obtained by the 32×32 Alpaca setting, with an average score of 0.562. For LLaMA-3-8B, 32×32 also gives the best square-group performance, with average scores of 0.481 using WikiText-2 calibration and 0.488

using Alpaca calibration. A similar trend appears on Qwen3-8B, where 32×32 Alpaca achieves the highest average score among the square-group settings. For Qwen3-14B, 32×32 Alpaca again performs best, reaching an average score of 0.561.

Larger square groups reduce the number of selection decisions: each 128×128 group covers 16 times as many weights as a 32×32 group. Consistent with this, 128×128 underperforms 32×32 in every setting of Table 8, and in Table 4 the 1×128 and 1×256 configurations obtain higher averages than 32×32 and larger square groups.

D Additional Ablation and Efficiency Analysis



(a) Regularization loss. (b) Speedup vs. group size. (c) Speedup vs. Sparsity

Figure 5: Additional analysis of GroupMask. (a) Training dynamics of the sparsity regularization loss with and without the hypernetwork. (b) Kernel speedup under different group sizes at 50% sparsity. (c) Kernel speedup under group size 64 at different sparsity ratios.

Figure 5a shows that the sparsity regularization loss decreases faster with the hypernetwork than without it, complementing the task-loss comparison in Figure 3a. Figure 5b shows that, at 50% sparsity, kernel speedup varies only slightly across the evaluated group sizes, and Figure 5c shows that speedup increases with sparsity.

E Baseline and Evaluation Details

For baselines whose official results are available under the same setting, we report the published numbers. For models or settings that are not reported by prior work and are not directly applicable, we leave the corresponding entries blank. The symbol * indicates that the result is taken from the corresponding paper or follows its reported setting. All zero-shot evaluations are conducted with lm-evaluation-harness using the official task metrics.

F Mask Export and Preserved Rate Computation

After mask learning, the continuous mask scores are converted into binary group selectors using the same thresholding rule as in evaluation. The exported binary masks are then applied to the original pretrained weights to obtain the final sparse model. For each projection matrix, the preserved rate is computed as the ratio between the number of active weights and the total number of weights after applying the binary masks. For QKV projections, we report the average preserved rate over the query, key, and value projection matrices.