LEAP: Learnable End-to-End Adaptive Pruning of Large Language Models

Mohammad Mozaffari [1 2] **Younes Hourri** [2] **Mohammad Rastegari** [1] **Mahyar Najibi** [1]

Abstract

Unstructured sparsity is now natively accelerated by recent GPU kernels and dataflow hardware, shifting the bottleneck from inference execution to the pruning algorithm. State-of-the-art methods for unstructured LLM pruning are layer-wise surrogates derived from the Optimal Brain Surgeon principle, and they sacrifice end-to-end accuracy, especially under aggressive sparsity. End-to-end alternatives such as MaskLLM and PATCH show that learnable masks can close this gap, but their categorical-over-patterns parameterization scales with the number of valid masks per row and does not port to the unstructured setting. We introduce LEAP, which replaces this intractable parameterization with a per-weight Bernoulli-via-Gumbel-Sigmoid relaxation that makes end-to-end unstructured mask learning tractable. Across five LLM families from 0.5B to 8B parameters at 50% and 60% sparsity, LEAP improves six-task average zero-shot accuracy by $+2.59$ points on average over ADMM, the best layer-wise baseline in our sweep.[1]

1. Introduction

Deployment of modern Large Language Models (LLMs) is bottlenecked by memory and compute, and weight sparsity has become a central tool for reducing both. Sparsity patterns split into three regimes: structured, semi-structured (*e.g.*, 2:4), and unstructured. Structured and semi-structured variants enjoy native GPU support but trade non-trivial accuracy for modest compression. Unstructured sparsity retains far higher accuracy, and recent kernel work (SpInfer (Fan et al., 2025), FlashLLM (Xia et al., 2023), MACKO (Macko & Boža, 2025)) together with sparsity-native dataflow hardware (Lie, 2022) now converts unstructured masks into real speedups on commodity GPUs and wafer-scale engines. The bottleneck has therefore shifted: the open problem is no longer how to execute unstructured sparsity but how to *induce* it with minimal accuracy loss.

The dominant algorithmic family for unstructured LLM pruning follows the Optimal Brain Surgeon (OBS) (Hassibi & Stork, 1993) lineage. Wanda (Sun et al., 2024), SparseGPT (Frantar & Alistarh, 2023), Thanos (Ilin & Richtárik, 2025), ADMM (Boža, 2024), and OPTIMA (Mozaffari et al., 2025a) all minimize a *layer-wise* reconstruction error as a surrogate for end-to-end model loss. This surrogate is cheap but misaligned with the quantity actually being optimized, and it accumulates local errors that compound in deep networks. Learnable-mask methods such as MaskLLM (Fang et al., 2024) and PATCH (Hourri et al., 2025) instead directly optimize masks with respect to the language modeling loss. They deliver state-of-the-art results but only for semi-structured patterns.

MaskLLM's parameterization assigns one learnable logit to each valid pattern inside a group and applies a Gumbel-softmax over this set. For 2:4 sparsity, the number of valid patterns per group of size 4 is $\binom{4}{2} = 6$, which is tractable. If one attempts to port this categorical-over-patterns scheme to unstructured 50% sparsity on a row of width d=4096, the number of valid masks is $\binom{4096}{2048} \approx 10^{1229}$, which cannot be stored, let alone indexed, as a set of logits. The parameterization that underlies MaskLLM and PATCH therefore *does not extend* to the unstructured regime, regardless of compute budget. This is not an engineering inconvenience but a combinatorial obstruction.

LEAP resolves this obstruction by replacing the categorical distribution over patterns with a product of independent Bernoullis, one per weight, each relaxed via the Gumbel-sigmoid trick. The parameter count per weight matrix scales as $O(mn)$ rather than $O(|\{\text{valid patterns}\}|)$, matching the weight count. This is the natural reformulation that preserves end-to-end differentiability for unstructured masks at LLM scale; other per-weight relaxations (*e.g.*, L_0 regularization (Louizos et al., 2018), continuous sparsifica-

[1]Elastix AI [2]Department of Computer Science, University of Toronto, Toronto, Ontario, Canada. Correspondence to: Mohammad Mozaffari <mmozaffari@elastix.ai>.

Accepted at the ICML 2026 Workshop on Resource-Adaptive Foundation Model Inference (AdaptFM), Seoul, South Korea.

[1]Code is available at https://github.com/Paramathic/patch/tree/leap.

tion (Savarese et al., 2020)) are conceptually related and similarly tractable. A small set of ingredients (Wanda-based initialization, a scale and temperature schedule, a global sparsity regularizer, and a magnitude-aware term) stabilize the resulting optimization. We keep pretrained weights *frozen* as a deliberate scope choice: decoupling mask learning from weight updates preserves calibration and simplifies deployment, while remaining compatible with any subsequent fine-tuning or distillation stage.

Our contributions are as follows:

- We identify a combinatorial obstruction that prevents MaskLLM/PATCH-style categorical-over-patterns parameterizations from being used for unstructured sparsity, and we propose a per-weight Bernoulli-via-Gumbel-sigmoid reformulation that makes end-to-end learning tractable.
- We present LEAP, a lightweight end-to-end unstructured pruning framework that operates on frozen pretrained weights and trains a per-weight mask in roughly 2,000 iterations on a small general-text calibration stream.
- On Qwen-2.5 0.5B, Gemma-3 1B, LLaMA-3.2 1B, LLaMA-3.2 3B, and LLaMA-3.1 8B at 50% and 60% sparsity, LEAP improves the six-task average zero-shot accuracy over ADMM, the best layer-wise baseline in our sweep, by $+2.59$ points on average across the ten (model, sparsity) settings and by up to $+5.40$ points on LLaMA-3.2 1B at 60% sparsity.

2. Related Work

Hardware support for unstructured sparsity. Recent kernel work (FlashLLM (Xia et al., 2023), SpInfer (Fan et al., 2025), MACKO (Macko & Boža, 2025)) achieves significant speedups for unstructured LLM sparsity at 50%–60% densities on commodity tensor cores, and wafer-scale dataflow accelerators (Lie, 2022) target unstructured patterns natively. These developments make unstructured masks a deployable compression target rather than a theoretical one.

Layer-wise OBS-derived pruning. Wanda (Sun et al., 2024) prunes based on the product of weight magnitude and input activation norms. SparseGPT (Frantar & Alistarh, 2023) solves a layer-wise Hessian-based reconstruction problem and jointly updates surviving weights. Thanos (Ilin & Richtárik, 2025) refines this reconstruction with multi-column updates, ADMM (Boža, 2024) alternates between mask and weight updates, and OPTIMA (Mozaffari et al., 2025a) casts the same reconstruction as a quadratic program. SLiM (Mozaffari et al., 2025b) extends the one-shot reconstruction setting to joint sparse-plus-low-rank-plus-quantized approximations. These methods all optimize local surrogates; their errors are provably aligned with global loss only under strong assumptions that do not hold in LLMs. Concurrently, ELSA (Lee et al., 2026) dispenses with the layer-wise surrogate altogether and uses a surrogate-free ADMM formulation to push unstructured sparsity into extreme regimes ($\sim$90%); we instead target the 50%–60% range that current accelerated kernels support, so these higher sparsity ratios are outside the scope of our work.

Per-weight learnable masks in general pruning. Per-weight learnable mask parameterizations have been explored in general neural network pruning, most notably through L_0 regularization (Louizos et al., 2018) and continuous sparsification (Savarese et al., 2020), both of which use continuous relaxations of per-weight binary gates. LEAP differs in three practical ways: (i) we target LLM-scale unstructured pruning specifically, where prior end-to-end methods (MaskLLM, PATCH) adopted categorical-over-patterns parameterizations that do not scale to the unstructured regime; (ii) we initialize from a one-shot Wanda mask rather than cold-starting, which sharply reduces the number of training steps required; (iii) we combine a global sparsity regularizer with magnitude-aware stabilization tailored to frozen pretrained weights.

End-to-end learnable masks. MaskLLM (Fang et al., 2024) and PATCH (Hourri et al., 2025) optimize masks directly against the language modeling loss by parameterizing a categorical distribution over the valid patterns inside each structured group. PATCH extends this to tile-level hybrids. A separate line of work folds sparsity into pretraining itself, e.g., SLoPe (Mozaffari et al., 2024), which combines double-pruned sparse weights with lazily attached low-rank adapters. These training-time approaches are complementary to the post-training mask-learning regime we study. Both MaskLLM and PATCH are restricted to semi-structured regimes because their logit table scales with $|\{\text{valid patterns}\}|$, which is bounded only when the group is small. For unstructured 50% sparsity over a single row of width 4096, this set has cardinality $\binom{4096}{2048}$, making the parameterization impossible. LEAP is, to our knowledge, the first practical reformulation that transfers end-to-end mask learning to the unstructured setting.

3. LEAP: Method

Why Per-Pattern Logits Do Not Scale. Fix a weight matrix $W \in \mathbb{R}^{m\times n}$ and a target density budget. A categorical-over-patterns parameterization, as used by MaskLLM and PATCH, partitions each row into groups of size g and stores a logit vector of length $|\mathcal{P}_g|$ per group, where $\mathcal{P}_g$ is the set of masks satisfying the pattern constraint. For 2:4 sparsity, $|\mathcal{P}_4| = \binom{4}{2} = 6$. For unstructured ρ-sparsity over an entire row of width n, the only natural group is the row itself and $|\mathcal{P}_n| = \binom{n}{\rho n}$. At n=4096 and ρ=0.5 this is

$\binom{4096}{2048} \approx 10^{1229}$, which cannot be represented as a logit table under any storage or indexing scheme. Smaller groups reintroduce a structural constraint that is exactly what unstructured sparsity is defined to avoid. We conclude that the per-pattern parameterization does not admit an unstructured extension.

LEAP replaces the categorical parameterization with a product of independent Bernoullis, one per weight. For each weight matrix $W \in \mathbb{R}^{m\times n}$ we introduce a parallel logit matrix $P \in \mathbb{R}^{m\times n}$ and a stochastic mask

$$M \;=\; \sigma\!\left(\frac{\alpha P + g}{\tau}\right), \tag{1}$$

where $g = -\log(-\log(u))$ is Gumbel noise with $u \sim \mathrm{Uniform}(0,1)$, σ is the sigmoid function, α is a scale factor, and τ is a temperature. Equation (1) is the Gumbel-sigmoid relaxation of a Bernoulli with logit αP_{ij} and scale τ. The parameter count per weight matrix is exactly mn, which is linear in the weight count and independent of ρ. The effective pruned weight is

$$\widetilde{W} \;=\; M \odot W. \tag{2}$$

We use soft masks throughout optimization. Hard sampling with straight-through estimators is unstable at LLM scale, and soft masks keep gradients well conditioned while the α, τ schedules below drive M toward $\{0,1\}$.

We initialize P from a one-shot Wanda (Sun et al., 2024) mask. Entries selected by Wanda are set to $+s$ and the rest to $-s$, where $s > 0$ is the initial mask strength. This gives the sigmoid relaxation a reasonable starting loss and makes the search local rather than cold.

Two lightweight schedules anneal Equation (1) from exploratory to decisive. The scale α is ramped from α_0 to α_T (e.g., $25 \to 350$), which amplifies P and pushes σ toward $\{0,1\}$. The temperature τ is decayed from τ_0 to τ_T (e.g., $4.0 \to 0.05$), sharpening the sigmoid. Early iterations explore many candidate supports; later iterations commit.

Let ρ be the target density (e.g., $\rho = 0.5$). Let $\widetilde{M}_i$ denote the soft mask for layer i and let N_i denote the number of parameters in W_i (so $N = \sum_i N_i$ is the total parameter count over prunable layers). LEAP enforces density *globally*, not per layer:

$$\mathcal{L}_{\text{sparsity}} \;=\; \lambda_1 \left| \frac{1}{N} \sum_i \|\widetilde{M}_i\|_1 - \rho \right|, \tag{3}$$

where λ_1 is a large positive coefficient. The global form lets individual layers adjust their density based on end-to-end importance.

To bias the optimization toward retaining higher-magnitude weights we add

$$\mathcal{L}_{\text{weight}} \;=\; -\lambda_2 \sum_i \|\widetilde{W}_i\|_1, \tag{4}$$

with $\lambda_2 > 0$ (typically $\sim$10). This term stabilizes mask learning and avoids degenerate minima that keep many small weights while dropping a few critical ones.

Full Objective and Scope. Combining Equations (3) and (4) with the language modeling loss on a calibration stream X yields

$$\mathcal{L} \;=\; \mathcal{L}_{\text{LM}}(\widetilde{W}; X) \;+\; \mathcal{L}_{\text{sparsity}} \;+\; \mathcal{L}_{\text{weight}}. \tag{5}$$

Only P is trained; W is held fixed. We treat this as a deliberate scope choice, not a compute concession: freezing W preserves the calibration of the pretrained weights, isolates the mask as the object being learned, and keeps the deployment pipeline simple. Joint weight-and-mask optimization is a natural extension (Section 5).

4. Experiments

Models. We evaluate LEAP across Qwen-2.5 0.5B (Yang et al., 2024), Gemma-3 1B (Team et al., 2025), LLaMA-3.2 1B, LLaMA-3.2 3B (Grattafiori et al., 2024), and LLaMA-3.1 8B (Grattafiori et al., 2024) at 50% and 60% unstructured sparsity.

Training setup. Following the dataset configuration of MaskLLM (Fang et al., 2024) and PATCH (Hourri et al., 2025), masks are trained for 2,000 steps with batch size 256 on sequences of length 4096 from SlimPajama (Soboleva et al., 2023). Weights are frozen. Hyperparameters are in Section A.

Evaluation. We report WikiText2 perplexity (Merity et al., 2017) at sequence length 4096 and zero-shot accuracy on six standard benchmarks: PIQA (Bisk et al., 2020), ARC-Easy and ARC-Challenge (Clark et al., 2018), Winogrande (Sakaguchi et al., 2020), OpenBookQA (Mihaylov et al., 2018), and MMLU (Hendrycks et al., 2021), using the `lm-evaluation-harness` (Gao et al., 2024).

Baselines. We compare against Wanda (Sun et al., 2024), SparseGPT (Frantar & Alistarh, 2023), Thanos (Ilin & Richtárik, 2025), and ADMM (Boža, 2024), each using its default configuration with 128 C4 (Raffel et al., 2020) calibration samples.

Main results. Table 1 summarizes WikiText2 perplexity and the six-task zero-shot average accuracy for all five models at 50% and 60% sparsity. Per-task breakdowns for the

0.5B–3B models are deferred to Section B. LEAP consistently outperforms all baselines, including ADMM, the best layer-wise baseline in our sweep. Averaging across the ten (model, sparsity) settings, LEAP improves the six-task average zero-shot accuracy over ADMM by $+2.59$ points, with the smallest gain being $+0.21$ points (LLaMA-3.1 8B at 50%, 57.71 vs. 57.50) and the largest being $+5.40$ points (LLaMA-3.2 1B at 60%, 50.39 vs. 44.99). On LLaMA-3.1 8B, LEAP reaches 7.66 PPL and 57.71 average accuracy at 50%, and 8.82 PPL and 54.47 average accuracy at 60%, against ADMM's 9.12/57.50 and 14.10/50.61 respectively. Gains widen at higher sparsity: ADMM is nearly competitive with LEAP on 50% LLaMA-3.1 8B but the 60% margin reopens to $+3.86$ points.

We include the learnable-mask baseline MaskLLM (Fang et al., 2024) at 50%. For Qwen-2.5 0.5B, Gemma-3 1B, and LLaMA-3.2 1B we use the numbers reported by PATCH (Hourri et al., 2025); for LLaMA-3.1 8B we instead reproduce MaskLLM ourselves using its publicly released checkpoint, obtaining 9.17 WikiText2 perplexity and 55.09 six-task average accuracy. Because MaskLLM's categorical-over-patterns parameterization is restricted to 2:4 (see Section 3), its row in Table 1 is 2:4 semi-structured at 50% density, not 50% unstructured. We compute its six-task average over the tasks shared with our evaluation (MMLU, PIQA, ARC-E, ARC-C, Winogrande, OBQA); RACE and HellaSwag are excluded. LLaMA-3.2 3B is not reported in the PATCH paper. Note that MaskLLM's small-model 2:4 accuracy in (Hourri et al., 2025) is notably below its large-model results, consistent with the known difficulty of training 2:4 masks on small backbones.

Ablations. Table 2 ablates the key components of LEAP on Qwen-2.5 0.5B at 50% unstructured sparsity. Removing the weight regularizer ($\lambda_2{=}0$) causes the largest accuracy drop ($44.93 \rightarrow 42.87$) and a 1.67-point PPL increase, confirming that magnitude-aware stabilization is the most load-bearing ingredient when weights are frozen. Disabling either the scale schedule (fixed α) or the temperature schedule (fixed τ) costs roughly 2 PPL points each while keeping average accuracy within 0.5 points of the full method, indicating that the two schedules contribute primarily to the sharpness of the final mask rather than to where it lands. Random initialization (no Wanda warm start) leaves the average zero-shot accuracy essentially unchanged (44.94 vs. 44.93) but worsens PPL by 2.54 points (14.43 vs. 11.89), suggesting that the warm start primarily improves language-modeling quality rather than zero-shot accuracy at this training budget. Per-task numbers are in Section C.

Sparsity allocation. We study how LEAP distributes sparsity across transformer blocks under the *global* budget of Equation (3). Figure 1 shows the learned per-block densities. In all four models at both sparsity levels, LEAP converges to a near-uniform per-block allocation, with only minor boundary effects at the earliest and latest blocks. This contrasts with 2:4-constrained methods such as PATCH, which exhibit non-trivial inter-block variation under the same end-to-end objective.

We read this as a regime-specific observation rather than a general claim against non-uniform allocation: under end-to-end optimization with a global budget, at the models and sparsity levels we study, we do not observe accuracy headroom from non-uniform per-block budgeting. A uniform per-block allocation is therefore a reasonable default in this regime. Whether the same holds at higher sparsities, at larger scales, or under joint weight-mask optimization is an open question.

Runtime and compute. Table 3 reports wall-clock mask-learning time for LEAP. Models up to 3B are trained with data parallelism on $4\times$H100; for LLaMA-3.1 8B we use model parallelism (no data parallelism) on the same $4\times$H100 node.

LEAP is more expensive than one-shot layer-wise methods such as Wanda, SparseGPT, Thanos, and ADMM, which finish in minutes on a single GPU. In exchange, LEAP delivers the accuracy gains reported above. Because mask learning is a one-time, offline preprocessing step and the learned mask is reused at every deployment, we view the additional compute as a favorable trade-off against the accuracy improvement.

Compared to the other learnable-mask line, LEAP is substantially cheaper. MaskLLM (Fang et al., 2024) reports 1,280 A100 GPU-hours for a 7B model and 2,304 A100 GPU-hours for a 13B model. Per NVIDIA's published specifications, the H100 delivers higher tensor-core throughput and memory bandwidth than the A100 (NVIDIA, 2023), so per-GPU-hour numbers across the two platforms are not directly comparable; even after generous allowance for the hardware gap, LEAP's 276 GPU-hours on LLaMA-3.1 8B is well below MaskLLM's reported cost at comparable scale. Two factors in the LEAP formulation contribute to this gap. First, LEAP trains one Bernoulli logit per weight, i.e., 1 mask parameter per weight entry. MaskLLM parameterizes each 2:4 group of 4 weights with 6 logits, i.e., 1.5 mask parameters per weight entry, so the trainable state is $1.5\times$ larger for an equivalently-sized model. Second, a per-weight Bernoulli is a simpler operation than a per-group softmax over 6 patterns, which reduces the per-step cost of the mask forward and backward.

Kernel compatibility. LEAP produces masks in the 50%–60% unstructured regime, which is exactly the regime that recent accelerated kernels target, so the masks it outputs are

Table 1. Summary of pruning results. We report WikiText2 perplexity (PPL ↓, sequence length 4096) and six-task zero-shot average accuracy (Avg. ↑, averaged over MMLU, PIQA, ARC-E, ARC-C, Winogrande, OBQA). Per-task breakdowns for the 0.5B–3B models are in Section B. Bold marks the best sparse method in each column of each sparsity block. MaskLLM† is 2:4 semi-structured (not unstructured); LLaMA-3.2 3B is not reported in the PATCH paper. The LLaMA-3.1 8B dense row is computed from (Hourri et al., 2025) by averaging over the six tasks we report (so it differs from PATCH's eight-task averages, which additionally include RACE and HellaSwag).

		Qwen-2.5 0.5B		Gemma-3 1B		LLaMA-3.2 1B		LLaMA-3.2 3B		LLaMA-3.1 8B	
Sparsity	Method	PPL↓	Avg.↑	PPL↓	Avg.↑	PPL↓	Avg.↑	PPL↓	Avg.↑	PPL↓	Avg.↑
0%	Dense	14.17	49.10	9.75	49.09	7.81	57.95	13.08	48.48	5.84	63.89
50%	Wanda	32.98	42.29	23.49	39.98	12.92	49.95	24.00	41.71	9.64	55.81
	SparseGPT	28.37	43.17	18.82	42.32	12.32	50.20	20.33	41.93	9.30	57.33
	Thanos	28.65	43.88	19.70	41.62	12.26	50.81	20.85	42.10	9.34	56.82
	ADMM	26.63	44.56	17.35	43.05	11.61	52.28	19.70	42.28	9.12	57.50
	MaskLLM†	15.22	40.91	12.82	43.39	12.93	42.59	—	—	9.17	55.09
	LEAP	**11.89**	**44.93**	**11.29**	**44.81**	**8.67**	**54.44**	**14.09**	**44.53**	**7.66**	**57.71**
60%	Wanda	90.50	36.64	71.53	33.83	31.13	38.77	83.42	34.66	21.66	45.19
	SparseGPT	60.95	38.65	47.98	36.78	22.00	43.78	40.56	37.29	15.30	49.63
	Thanos	62.22	38.94	46.78	36.95	22.48	42.49	44.29	37.64	16.10	48.87
	ADMM	50.55	40.41	33.87	38.73	19.14	44.99	33.41	38.33	14.10	50.61
	LEAP	**13.16**	**43.41**	**13.06**	**42.42**	**9.77**	**50.39**	**15.66**	**41.53**	**8.82**	**54.47**

Table 2. Ablations on Qwen-2.5 0.5B at 50% unstructured sparsity. Per-task breakdowns are in Section C.

Variant	PPL ↓	Avg. Acc. ↑
LEAP (full)	**11.89**	44.93
λ_2=0	13.56	42.87
Random init	14.43	**44.94**
Fixed α	14.03	44.44
Fixed τ	13.99	44.52

Table 3. LEAP mask-learning time, converted to GPU-hours. The 0.5B–3B models use data parallelism on 4×H100; the 8B model uses model parallelism on the same node.

Model	Wall-clock (4×H100)	GPU-hours
Qwen-2.5 0.5B	5h 40m	22.7
LLaMA-3.2 1B	7h 55m	31.7
Gemma-3 1B	16h	64
LLaMA-3.2 3B	19h	76
LLaMA-3.1 8B	69h	276

deployable without any change to the kernel stack. We quote the reported numbers from those kernels; we do not measure them ourselves. Flash-LLM (Xia et al., 2023) reports up to 2.9× faster SpMM than Sputnik and up to 1.5× faster than SparTA, which translates to up to 3.8× higher end-to-end throughput over DeepSpeed and 3.6× over FasterTransformer on OPT-30B/66B/175B under unstructured sparsity. SpInfer (Fan et al., 2025) improves on this line, reporting up to 2.14× faster SpMM than Flash-LLM and 2.27× faster than SparTA across 30%–70% sparsity, with end- to-end inference speedups up to 1.58× and surpassing cuBLAS from as low as 30% sparsity. MACKO (Macko & Boža, 2025), designed explicitly for the low-sparsity regime, reports 1.2–1.5× speedup and 1.5× memory reduction over dense fp16 at 50% sparsity, with a 1.5× end-to-end inference speedup on a 50%-sparse LLaMA-2 7B. Beyond commodity GPUs, wafer-scale dataflow accelerators (Lie, 2022) report near-linear speedup with unstructured sparsity on GPT-class models. LEAP's 50%–60% unstructured masks feed directly into all of these kernels; the end-to-end deployment speedup is the composition of LEAP's accuracy-preserving mask with the speedups the kernels deliver.

5. Discussion and Limitations

Mask learning memory overhead. The per-weight logit matrix P has the same shape as W and therefore roughly doubles the trainable state during mask learning, on top of the optimizer state P introduces (momentum and second-moment buffers). In practice, however, this is not the dominant component of GPU memory at our sequence length. At sequence length 4096 the activations stored for backpropagation are the majority of the memory footprint, and the P-plus-optimizer state fits alongside them for every model up to 3B on a single H100. Accordingly, we train masks for 0.5B–3B models with data parallelism across 4×H100 within one node: the entire model, its activations, P, and P's optimizer state fit on each GPU, and the four GPUs process four shards of the calibration batch in parallel. For LLaMA-3.1 8B, the model plus activations at sequence length 4096 no longer fit on a single H100, so we switch to model parallelism within the same 4×H100 node. The P-induced memory doubling can be further reduced by partitioning P and its optimizer state across data-parallel ranks using ZeRO-style optimizer sharding (Rajbhandari et al., 2020), or by adopting optimizers that compress curvature

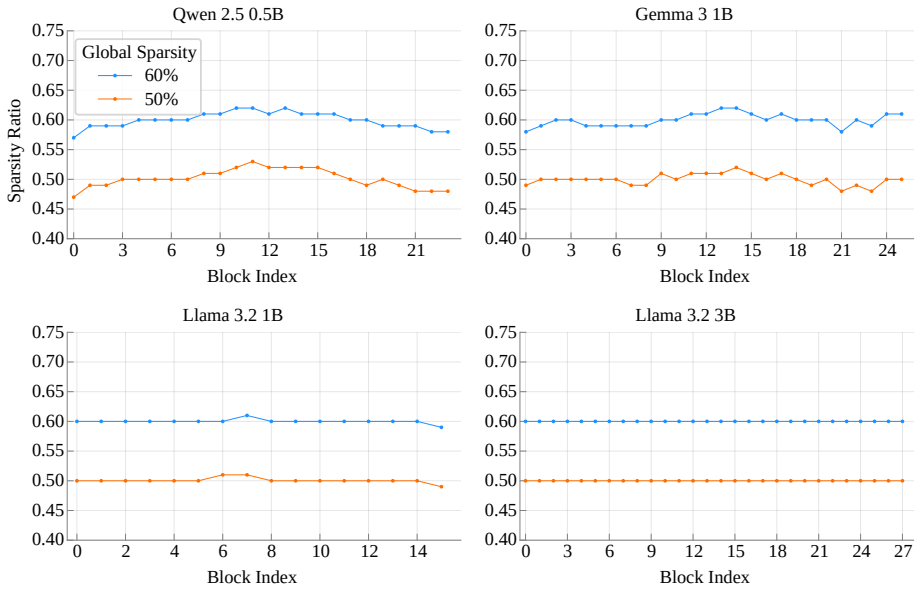

Figure 1. Learned per-block density at a global unstructured sparsity budget of 50% and 60% for Qwen-2.5 0.5B, Gemma-3 1B, LLaMA-3.2 1B, and LLaMA-3.2 3B. Masks are initialized from Wanda and trained for 2,000 steps. The grey dashed line marks the global budget.

state via low-rank updates (Mozaffari et al., 2023); we did not need either at our scales but they are drop-in extensions for larger models. Note that P is discarded after mask learning; deployment memory is unaffected and matches a standard sparse checkpoint.

Frozen weights as a scope choice. LEAP keeps W fixed during mask learning. This is not a compute concession: freezing W preserves pretrained calibration, isolates the mask as the learned object, and keeps deployment pipelines simple. Joint weight-and-mask optimization is a natural extension, and can be layered on top of LEAP as a short fine-tuning stage.

Kernel compatibility, not kernel speedup. We do not claim downstream inference speedups as contributions of LEAP. LEAP produces unstructured masks in the density regime already targeted by existing kernels; the reported end-to-end speedup is the composition of LEAP's accuracy improvement with the speedups those kernels deliver.

6. Conclusion

We identified a combinatorial obstruction that prevents the categorical-over-patterns parameterization used by MaskLLM and PATCH from extending to unstructured sparsity, and we proposed LEAP, a per-weight Bernoulli-via-Gumbel-sigmoid reformulation that makes end-to-end unstructured mask learning tractable. On five LLM families at 50% and 60% sparsity, LEAP improves the six-task average zero-shot accuracy over ADMM, the best layer-wise baseline in our sweep, by $+2.59$ points on average across the ten (model, sparsity) settings and by up to $+5.40$ points on LLaMA-3.2 1B at 60% sparsity, while keeping pretrained weights frozen. Because the resulting masks are in the 50%–60% unstructured regime, they are directly consumable by existing accelerated kernels. We believe the per-weight reformulation opens a practical path to end-to-end unstructured compression of large foundation models.

Impact Statement

LEAP reduces the memory and compute cost of deploying large language models by producing 50%–60% unstructured masks that are consumable by existing accelerated kernels. The intended impact is to lower the barrier to running capable models on commodity hardware and in resource-constrained settings, and to reduce the energy footprint of inference at scale. The same compression that enables lower-cost inference can also make it easier to deploy models outside of supervised environments, and compressed models may exhibit different failure modes on long-tail inputs than their dense counterparts; downstream users applying LEAP to safety-critical deployments should re-evaluate on the distributions they actually care about rather than relying solely on our headline benchmarks. Because LEAP keeps pretrained weights frozen and only trains a mask, it does not introduce new training-data provenance concerns beyond those already present in the base models.

References

Bisk, Y., Zellers, R., Bras, R. L., Gao, J., and Choi, Y. PIQA: Reasoning about physical commonsense in natural language. In *AAAI Conference on Artificial Intelligence*, 2020.

Boža, V. Fast and effective weight update for pruned large language models. *arXiv preprint arXiv:2401.02938*, 2024.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Fan, R., Yu, X., Dong, P., Li, Z., Gong, G., Wang, Q., Wang, W., and Chu, X. SpInfer: Leveraging low-level sparsity for efficient large language model inference on GPUs. In *Proceedings of the Twentieth European Conference on Computer Systems (EuroSys)*, 2025.

Fang, G., Yin, H., Muralidharan, S., Heinrich, G., Pool, J., Kautz, J., Molchanov, P., and Wang, X. MaskLLM: Learnable semi-structured sparsity for large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Frantar, E. and Alistarh, D. SparseGPT: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning (ICML)*, 2023.

Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Noac'h, A. L., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. A framework for few-shot language model evaluation. https://github.com/EleutherAI/lm-evaluation-harness, 2024.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Hassibi, B. and Stork, D. G. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in Neural Information Processing Systems*, 1993.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. In *International Conference on Learning Representations (ICLR)*, 2021.

Hourri, Y., Mozaffari, M., and Dehnavi, M. M. PATCH: Learnable tile-level hybrid sparsity for large language models. *arXiv preprint arXiv:2509.23410*, 2025.

Ilin, I. and Richtárik, P. Thanos: A block-wise pruning algorithm for efficient large language model compression. *arXiv preprint arXiv:2504.05346*, 2025. URL https://arxiv.org/abs/2504.05346.

Lee, K., Jang, H., Lee, D., Alistarh, D., and Lee, N. The unseen frontier: Pushing the limits of LLM sparsity with surrogate-free ADMM. In *International Conference on Learning Representations (ICLR)*, 2026.

Lie, S. Harnessing the power of sparsity for large GPT AI models. Technical report, Cerebras Systems, 2022.

Louizos, C., Welling, M., and Kingma, D. P. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations (ICLR)*, 2018.

Macko, V. and Boža, V. MACKO: Sparse matrix-vector multiplication for low sparsity. *arXiv preprint arXiv:2511.13061*, 2025.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. In *International Conference on Learning Representations (ICLR)*, 2017.

Mihaylov, T., Clark, P., Khot, T., and Sabharwal, A. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Mozaffari, M., Li, S., Zhang, Z., and Dehnavi, M. M. MKOR: Momentum-enabled kronecker-factor-based optimizer using rank-1 updates. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Mozaffari, M., Yazdanbakhsh, A., Zhang, Z., and Dehnavi, M. M. SLoPe: Double-pruned sparse plus lazy low-rank adapter pretraining of LLMs. *arXiv preprint arXiv:2405.16325*, 2024.

Mozaffari, M., Kushnir, S., Dehnavi, M. M., and Yazdanbakhsh, A. OPTIMA: Optimal one-shot pruning for LLMs via quadratic programming reconstruction. *arXiv preprint arXiv:2512.13886*, 2025a.

Mozaffari, M., Yazdanbakhsh, A., and Dehnavi, M. M. SLiM: One-shot quantized sparse plus low-rank approximation of LLMs. In *International Conference on Machine Learning (ICML)*, 2025b.

NVIDIA. NVIDIA H100 Tensor Core GPU. https://www.nvidia.com/en-us/data-center/h100/, 2023. Accessed: 2026-04-24.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text

transformer. *Journal of Machine Learning Research*, 21 (140):1–67, 2020.

Rajbhandari, S., Rasley, J., Ruwase, O., and He, Y. ZeRO: Memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2020.

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. WinoGrande: An adversarial Winograd schema challenge at scale. In *AAAI Conference on Artificial Intelligence*, 2020.

Savarese, P., Silva, H., and Maire, M. Winning the lottery with continuous sparsification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Soboleva, D., Al-Khateeb, F., Myers, R., Steeves, J. R., Hestness, J., and Dey, N. SlimPajama: A 627b token cleaned and deduplicated version of RedPajama. https://huggingface.co/datasets/cerebras/SlimPajama-627B, 2023.

Sun, M., Liu, Z., Bair, A., and Kolter, J. Z. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Team, G., Kamath, A., Ferret, J., Pathak, S., Vieillard, N., Merhej, R., Perrin, S., Matejovicova, T., Ramé, A., Rivière, M., et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Xia, H., Zheng, Z., Li, Y., Zhuang, D., Zhou, Z., Qiu, X., Li, Y., Lin, W., and Song, S. L. Flash-LLM: Enabling cost-effective and highly-efficient large generative model inference with unstructured sparsity. In *Proceedings of the VLDB Endowment (PVLDB), Vol. 17, No. 2*, pp. 211–224, 2023.

Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., and Qiu, Z. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

A. Hyperparameters

Table 4 lists the hyperparameters used in all LEAP experiments in the main paper. Hyperparameters were tuned on the smallest model (Qwen-2.5 0.5B) and reused for larger models.

Table 4. Hyperparameters used for LEAP experiments.

Hyperparameter	Value
Learning rate	$\{10^{-2}, 10^{-3}\}$
Temperature pair (τ_0, τ_T)	$(4.00, 0.05)$
Scale pair (α_0, α_T)	$(25, 350)$
Weight regularizer λ_2	10
Initial mask strength s	3
Weight decay	0
Batch size	256
Sequence length	4096
Mask training steps	2,000
Calibration corpus	SlimPajama (Soboleva et al., 2023)

B. Per-Task Zero-Shot Results

This appendix reports the per-task breakdown behind the averaged numbers in Table 1. Each table covers one model at 50% and 60% unstructured sparsity across the six zero-shot tasks (MMLU, PIQA, ARC-E, ARC-C, Winogrande, OBQA) plus WikiText2 perplexity at sequence length 4096. The MaskLLM† rows are $2{:}4$ semi-structured at 50% density and are reproduced from (Hourri et al., 2025).

Table 5. Per-task results on Qwen-2.5 0.5B. PPL is on WikiText2 (lower is better). Other columns are zero-shot accuracy in percent.

Method	Sparsity	PPL ↓	MMLU	PIQA	ARC-E	ARC-C	Wino.	OBQA	Avg.
Dense	0%	14.17	24.95	74.81	71.93	35.41	58.72	28.80	49.10
Wanda	50%	32.98	22.92	67.46	60.77	26.37	56.04	20.20	42.29
SparseGPT		28.37	24.81	69.10	61.15	27.13	55.64	21.20	43.17
Thanos		28.65	23.09	69.75	62.16	27.99	56.51	23.80	43.88
ADMM		26.63	24.05	70.08	63.80	27.73	56.51	25.20	44.56
MaskLLM†		15.22	25.11	67.03	56.57	23.98	52.57	20.20	40.91
LEAP		**11.89**	23.80	**71.27**	63.13	**27.90**	**60.03**	23.20	**44.93**
Wanda	60%	90.50	23.02	62.13	49.87	18.34	51.07	15.40	36.64
SparseGPT		60.95	24.54	65.61	52.02	21.59	51.54	16.60	38.65
Thanos		62.22	24.62	64.53	52.86	20.65	52.17	18.80	38.94
ADMM		50.55	25.16	65.29	55.89	22.69	53.82	19.60	40.41
LEAP		**13.16**	24.44	**68.66**	**60.61**	**25.09**	**58.64**	**23.00**	**43.41**

C. Per-Task Ablation Results

Table 9 reports the per-task zero-shot breakdown for the ablation study summarized in Table 2 (Qwen-2.5 0.5B at 50% unstructured sparsity).

D. Additional Notes on the Combinatorial Argument

The obstruction argument in Section 3 relies only on the statement that a categorical parameterization over $\mathcal{P}_n$ requires $|\mathcal{P}_n|$ logits. For ρ-sparsity at row width n,

$$|\mathcal{P}_n| \;=\; \binom{n}{\rho n},$$

and Stirling gives

$$\log_2 \binom{n}{\rho n} \;=\; n\,H_2(\rho) + o(n), \qquad H_2(\rho) = -\rho \log_2 \rho - (1{-}\rho)\log_2(1{-}\rho).$$

Table 6. Per-task results on Gemma-3 1B.

Method	Sparsity	PPL ↓	MMLU	PIQA	ARC-E	ARC-C	Wino.	OBQA	Avg.
Dense	0%	9.75	36.92	74.27	65.53	31.31	60.30	26.20	49.09
Wanda	50%	23.49	26.27	64.96	51.98	23.55	54.93	18.20	39.98
SparseGPT		18.82	25.64	67.85	54.55	26.53	57.14	22.20	42.32
Thanos		19.70	25.37	67.63	52.99	27.13	54.38	22.20	41.62
ADMM		17.35	**27.92**	69.75	56.06	26.37	56.04	22.20	43.05
MaskLLM†		12.82	25.03	69.91	60.27	27.65	56.27	21.20	43.39
LEAP		**11.29**	27.39	**71.98**	**60.90**	**28.24**	**57.53**	**22.80**	**44.81**
Wanda	60%	71.53	22.93	59.36	39.65	18.86	50.20	12.00	33.83
SparseGPT		47.98	23.06	62.24	43.77	21.76	52.25	17.60	36.78
Thanos		46.78	23.25	62.57	44.49	51.59	53.20	16.60	36.95
ADMM		33.87	**25.77**	64.15	47.22	22.44	54.62	18.20	38.73
LEAP		**13.06**	24.90	**69.70**	**57.53**	**25.60**	**55.80**	**21.00**	**42.42**

Table 7. Per-task results on LLaMA-3.2 1B.

Method	Sparsity	PPL ↓	MMLU	PIQA	ARC-E	ARC-C	Wino.	OBQA	Avg.
Dense	0%	7.81	54.13	76.55	74.28	42.75	69.38	30.60	57.95
Wanda	50%	12.92	40.79	72.03	65.45	32.34	63.69	25.40	49.95
SparseGPT		12.32	37.96	73.45	65.19	33.02	66.38	25.20	50.20
Thanos		12.26	40.11	72.80	64.77	32.85	67.72	26.60	50.81
ADMM		11.61	42.90	74.48	66.62	34.56	66.93	28.20	52.28
MaskLLM†		12.93	26.28	69.10	57.41	25.85	55.48	21.40	42.59
LEAP		**8.67**	**44.55**	**75.19**	**71.09**	**39.33**	**67.48**	**29.00**	**54.44**
Wanda	60%	31.13	25.53	65.23	47.90	22.70	55.25	16.00	38.77
SparseGPT		22.00	31.27	69.37	53.66	26.02	61.33	21.00	43.78
Thanos		22.48	29.23	67.63	55.01	26.02	57.85	19.20	42.49
ADMM		19.14	33.46	69.15	57.70	27.39	59.82	22.40	44.99
LEAP		**9.77**	**37.55**	**74.32**	**66.50**	**34.81**	**63.14**	**26.00**	**50.39**

At ρ=0.5, $H_2(\rho)$=1, so the logit table for a single row of width n=4096 already requires $\sim 2^{4096}$ entries. No sparsification of the logit table preserves the unstructured constraint: any coarser grouping reimposes structural constraints on $\mathcal{P}$. This is why we argue the per-weight Bernoulli parameterization is the natural reformulation, not a heuristic alternative.

Table 8. Per-task results on LLaMA-3.2 3B.

Method	Sparsity	PPL ↓	MMLU	PIQA	ARC-E	ARC-C	Wino.	OBQA	Avg.
Dense	0%	13.08	47.36	69.97	64.18	29.18	55.80	24.40	48.48
Wanda	50%	24.00	30.52	64.09	57.41	24.06	54.38	19.80	41.71
SparseGPT	50%	20.33	29.38	64.74	56.52	24.15	56.20	20.60	41.93
Thanos	50%	20.85	28.94	65.40	55.93	24.40	56.35	**21.60**	42.10
ADMM	50%	19.70	29.38	65.67	55.89	25.26	56.27	21.20	42.28
LEAP	50%	**14.09**	**34.37**	**68.44**	**61.99**	**26.11**	55.25	21.00	**44.53**
Wanda	60%	83.42	23.02	59.96	43.81	18.09	50.28	12.80	34.66
SparseGPT	60%	40.56	22.90	61.59	48.40	21.25	52.80	16.80	37.29
Thanos	60%	44.29	23.78	62.02	48.65	21.33	52.25	17.80	37.64
ADMM	60%	33.41	24.22	62.40	50.13	22.44	52.96	17.80	38.33
LEAP	60%	**15.66**	**24.66**	**67.90**	**56.69**	**25.09**	**55.49**	**19.40**	**41.53**

Table 9. Per-task ablation results on Qwen-2.5 0.5B at 50% unstructured sparsity.

Variant	PPL ↓	MMLU	PIQA	ARC-E	ARC-C	Wino.	OBQA	Avg.
LEAP (full)	**11.89**	23.80	**71.27**	**63.13**	**27.90**	**60.03**	**23.20**	44.93
λ_2=0	13.56	26.32	68.61	60.82	26.20	54.07	21.20	42.87
Random init	14.43	**34.16**	68.44	62.84	26.96	55.41	21.80	**44.94**
Fixed α	14.03	32.31	68.34	62.96	27.22	54.38	21.40	44.44
Fixed τ	13.99	33.81	68.34	62.21	26.62	54.14	22.00	44.52