SUSI: Scalable Semi-Structured Pruning via Differentiable Subset Sampling

Anonymous ACL submission

Abstract

Semi-structured $N{:}M$ sparsity has emerged as a practical direction for accelerating large language models (LLMs). However, existing learnable-mask approaches incur substantial parameter and memory overhead, limiting their scalability to large models and aggressive sparsity regimes. In this work, we revisit $N{:}M$ pruning from a perspective that reconciles efficiency with scalability. We propose SUSI[1], **S**emi-structured pr**U**ning via **S**ubset sampl**I**ng, a lightweight semi-structured pruning framework that learns sparsity masks through differentiable subset sampling via weighted reservoir sampling. Unlike prior methods that model full categorical distributions over all feasible $N{:}M$ patterns, SUSI reformulates sparsity mask learning as a sampling without replacement from a compact set of logits, reducing trainable parameters from combinatorial complexity to $\mathcal{O}(M)$. As a result, SUSI requires 1.5–8.75$\times$ fewer learnable parameters and significantly lower memory cost, while remaining fully aligned with hardware-friendly sparsity patterns. Extensive evaluations across multiple scales of the Qwen2.5 LLM family (0.5-7B parameters) demonstrate that SUSI achieves competitive performance with strong memory efficiency, stability across random seeds, and scalability to more aggressive $N{:}M$ sparsity patterns, offering a practical path toward efficient LLM deployment.

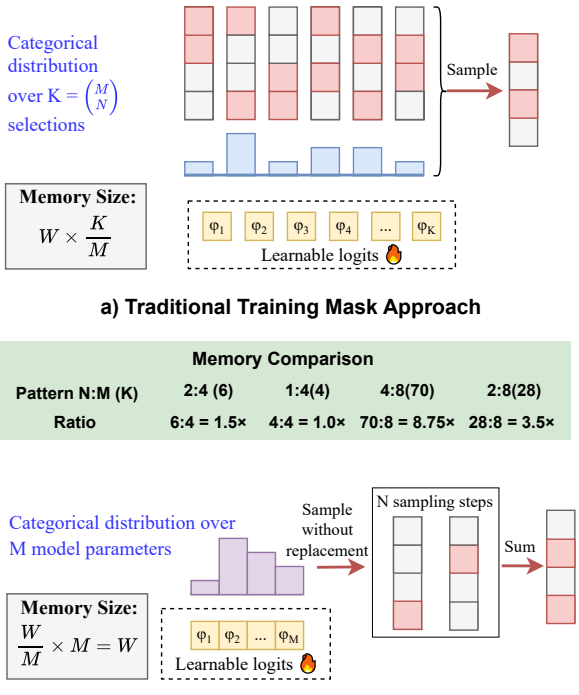

Figure 1: Learnable semi-structured $N{:}M$ sparsity methods: a) modeling the mask selection process using a categorical distribution over feasible masks, and b) our proposed method by learning to sample subsets without replacement of model parameters. The proposed method is more memory efficient than previous works for most practical $N{:}M$ sparsity patterns. The memory advantage becomes more pronounced as M increases or when N is around $M/2$.

1 Introduction

Pruning techniques play a central role in developing sparse LLMs that reduce memory footprint and improve inference efficiency. Post-training pruning methods generally fall into three categories: (i) unstructured pruning, which removes individual weight parameters without considering architectural constraints (Sun et al., 2024); (ii) structured pruning, which removes entire components such as neurons, attention heads, or layers (Xia et al., 2024; Le et al., 2025); and (iii) semi-structured pruning, which balances flexibility and regularity by enforcing hardware-friendly sparsity patterns such as $N{:}M$ sparsity (Fang et al., 2024; Huang et al., 2025). In this work, we focus on semi-structured pruning because it removes redundant weights while maintaining regular sparsity patterns that are compatible with modern accelerators. In particular, enforcing $N{:}M$ sparsity patterns (Hubara et al.,

[1] https://anonymous.4open.science/r/susi-2E2C

2021) provides principled trade-offs between sparsity flexibility and hardware efficiency, enabling practical inference acceleration.

Recent progress in semi-structured pruning has increasingly focused on learnable-mask approaches that directly optimize $N{:}M$ sparsity patterns through gradient-based training. Representative methods (e.g., MaskLLM (Fang et al., 2024) and AST (Huang et al., 2025)) formulate each pruning group as a categorical distribution over all feasible $N{:}M$ configurations, which demonstrate strong pruning performance and generalization. However, the major challenge of this approach is the combinatorial parameterization required to model a multinomial distribution of size $\binom{M}{N}$ for each weight group. Specifically, as M increases, this design imposes substantial memory and parameter overhead, as demonstrated in the Figure 1 (a). Motivated by this limitation, we propose a lightweight alternative that replaces full categorical modeling with differentiable subset sampling, termed SUSI (**S**emi-structured pr**U**ning via **S**ubset sampl**I**ng). Specifically, instead of parameterizing the probability of every possible $N{:}M$ pattern, SUSI samples N entries without replacement from a compact logit vector of size M using Weighted Reservoir Sampling and Gumbel-Top-K relaxation. This reduces the learnable mask parameters from $\mathcal{O}\left(\binom{M}{N}\right)$ a simple $\mathcal{O}(M)$ per group while retaining differentiability and hardware alignment, as shown in Figure 1 (b). By avoiding combinatorial distributions and relying on efficient subset-sampling dynamics, the proposed method enables scalable, stable, and memory-efficient mask learning for semi-structured pruning of modern LLMs.

2 Related Work

Semi-structured pruning has emerged as an effective compromise between structured pruning (An et al., 2024; Liu et al., 2025b) and unstructured pruning (Dong et al., 2024; Sun et al., 2024). By enforcing regular sparsity patterns (i.e., $N{:}M$ sparsity), semi-structured pruning enables efficient hardware acceleration while preserving model accuracy (Hubara et al., 2021).

2.1 Heuristic-Based Methods

Heuristic-based approaches, including SparseGPT (Frantar and Alistarh, 2023) and Wanda (Sun et al., 2024), rely on a small calibration dataset, typically a subset of the pretraining data, to approximate the knowledge encoded in a pretrained language model. These methods assign importance scores to individual weights or weight groups based on heuristics such as weight magnitude, gradient information, or second-order statistics (e.g., Hessian approximations), and prune parameters accordingly. While computationally efficient, the choice of importance criteria is largely heuristic and may not be optimal for downstream performance. Moreover, the limited calibration data may fail to adequately capture the rich and diverse knowledge embedded in large language models, potentially leading to suboptimal pruning decisions.

2.2 Learning-Based Methods

Learning-based semi-structured pruning has recently gained attention due to its ability to optimize pruning decisions directly through gradient-based training, rather than relying on handcrafted importance metrics. Under the $N{:}M$ sparsity constraint, the objective is to retain exactly N nonzero weights within each group of M parameters, thereby producing hardware-friendly sparsity patterns while minimizing performance degradation (Zhou et al., 2021). A prominent line of work formulates mask learning as an optimization problem over categorical distributions that enumerate all feasible $N{:}M$ configurations. For example, MaskLLM (Fang et al., 2024) models each group's pruning mask as a multinomial distribution and employs Gumbel-Softmax sampling to enable differentiable training. This approach achieves strong pruning performance and good generalization across tasks; however, it incurs substantial computational and memory overhead due to the $\mathcal{O}\left(\binom{M}{N}\right)$ parameterization per group. On the other hand, ProxSparse (Liu et al., 2025a) reduces training complexity via regularized optimization and smaller learning samples. Despite these improvements, challenges related to large-scale training efficiency and performance generalization remain open.

In contrast to prior work, our study follows a large-scale learning-based paradigm while substantially reducing the parameterization cost of mask learning. Specifically, we improve the categorical formulation with an $\mathcal{O}(M)$ parameterization per group by leveraging weighted reservoir sampling and differentiable subset sampling. This design achieves significant memory savings while retain-

ing high pruning accuracy, making it well-suited for large-scale semi-structured pruning, as detailed in the following section.

3 Preliminaries

3.1 Weighted Reservoir Sampling

Weighted Reservoir Sampling (WRS) (Efraimidis and Spirakis, 2006) is an extension of the Reservoir Sampling class of algorithms (Vitter, 1985), which aims to sample K elements from a set of N. In WRS, each item is associated with a non-negative weight, and items with larger relative weights are preferentially sampled. Formally, consider a finite population $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$ equipped with weights $\boldsymbol{w} = [w_1, w_2, \ldots, w_N]$, WRS produces an ordered subset $\mathcal{Y} = \{y_1, y_2, \ldots, y_K\}$ given by:

$$\mathbb{P}_{\text{WRS}}(\mathcal{Y}|\boldsymbol{w}) = \frac{w_{y_1}}{W} \times \frac{w_{y_2}}{W - w_{y_1}} \times \ldots \times \frac{w_{y_K}}{W - \sum_{j=1}^{K-1} w_{y_j}}, \quad (1)$$

where $W = \sum_{i=1}^{N} w_i$ denotes the total weight and w_{y_i} is the weight of element y_i. This distribution corresponds to a without-replacement sampling process, where the probability of selecting a subset is proportional to its item weights.

3.2 Gumbel-Top-K Trick

Gumbel-Max (Gumbel, 1954) can be viewed as a monotonic transformation of the WRS technique, providing a reparameterization trick to sample from a categorical distribution by perturbing the log-probabilities with Gumbel noise. Consider a categorical distribution over $\{x_1, \ldots, x_N\}$, parameterized by logits $\boldsymbol{\phi} = [\phi_1, \ldots, \phi_N]$, with probabilities $\pi_i = \exp(\phi_i)/\sum_{j=1}^{N} \exp(\phi_j)$. The Gumbel-Max trick samples from this distribution by first associating each item with a random key obtained via Gumbel perturbation:

$$\kappa_i = \phi_i + g_i, \quad g_i \overset{\text{i.i.d}}{\sim} \text{Gumbel}(0, 1), \quad (2)$$

where each g_i is independently drawn from the $\text{Gumbel}(0, 1)$ distribution. The sampler outputs the item x_j having the largest key κ_j. The index j is the output of taking argmax over every perturbed key values ($j = \text{argmax}_i\ \kappa_i$).

Gumbel-Top-K (Xie and Ermon, 2019) is a generalization of the Gumbel-Max trick, selecting the top-K items with the highest keys, rather than just the maximum. This corresponds to drawing K items without replacement from a categorical distribution over N candidates. By replacing the argtop_K operator with a sequence of softmax relaxations (Plötz and Roth, 2018), the sampling process becomes differentiable, thereby enabling end-to-end learning via backpropagation.

To sample a size-K subset using the Gumbel-Top-K trick, logits are first independently perturbed with Gumbel noise to form random keys κ_i, analogously to the Gumbel-Max trick. Subsequently, a sequence of softmax operations is applied to generate differentiable approximations of the selected one-hot indicators. Let $\boldsymbol{\alpha}^{(k)} = [\alpha_1, \ldots, \alpha_N]$ denote adjusted keys at sampling step k, which are defined recursively as follows:

$$\begin{aligned} \boldsymbol{\alpha}^{(1)} &= [\kappa_1, \ldots, \kappa_N], \\ \boldsymbol{\alpha}^{(k)} &= \boldsymbol{\alpha}^{(k-1)} + \log(1 - \boldsymbol{\mu}^{(k-1)}), \end{aligned} \quad (3)$$

where $\boldsymbol{\mu}^{(k-1)} = [\mu_1^{(k-1)}, \ldots, \mu_N^{(k-1)}]$ is the relaxed one-hot indicator of the item selected at step $k-1$. This representation is achieved by applying a softmax with temperature τ to the adjusted keys:

$$\mu_i^{(k-1)} = \frac{\exp(\alpha_i^{(k-1)}/\tau)}{\sum_{j=1}^{N} \exp(\alpha_j^{(k-1)}/\tau)}. \quad (4)$$

After K iterations, the procedure produces an ordered collection of relaxed one-hot vectors $\mathcal{S} = \{\boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(K)}\}$. Summing these vectors yields a soft K-hot representation, and the mapping from logits ϕ_i to this representation is differentiable, enabling gradient-based training.

4 Methodology

4.1 Problem Statement

The task of determining the optimal $N{:}M$ sparsity pattern can be formulated as selecting, for each contiguous group of M parameters, a length-M binary mask containing exactly N non-zero entries that minimizes the calibration loss. Let G be the number of such groups, $\mathbf{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_G\}$ the corresponding parameter groups, and $\mathbf{M} = \{\mathbf{m}_1, \ldots, \mathbf{m}_G\}$ their associated masks. The resulting optimization problem is:

$$\mathbf{M}^* = \underset{\mathbf{M}}{\text{argmin}}\ \mathcal{L}_{\text{CE}}(\mathcal{D}; \mathbf{W} \odot \mathbf{M}). \quad (5)$$

Here, $\mathcal{L}_{\text{CE}}$ denotes the cross-entropy loss for language modeling, $\mathcal{D}$ is the calibration dataset, and $\odot$ represents the element-wise multiplication between each weight group and its corresponding

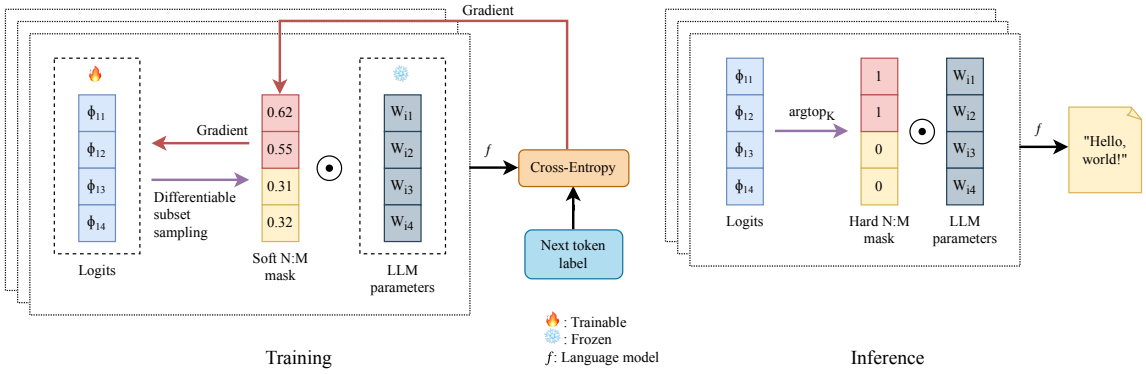

Figure 2: Overview of the SUSI Framework for Semi-Structured Pruning via Differentiable Subset Sampling, illustrating the training and inference phases.

mask. This optimization problem is NP-hard due to the combinatorial search space, containing $\binom{M}{N}^G$ feasible configurations. For LLMs, the number of weight groups G is gargantuan, rendering exhaustive search intractable. Therefore, in the following section, we reformulate the objective as a stochastic variational optimization problem to obtain a tractable and efficient approximation.

4.2 Proposed Method

An overview of the proposed method, SUSI, is presented in Figure 2. Stochastic variational optimization (Bird et al., 2018) relies on the observation that for any arbitrary distribution $q(x)$, the expected value of a function $f(x)$ provides an upper bound for its minimum:

$$\min_x f(x) \leq \mathbb{E}_{q(x)}[f(x)]. \tag{6}$$

By modeling pruning masks as random variables, the optimization problem in Equation 5 can be reframed as minimizing a variational upper bound of the objective with respect to the variational distribution parameters. Specifically, we solve:

$$\mathbf{\Phi}^* = \operatorname*{argmin}_{\mathbf{\Phi}} \mathbb{E}_{\mathbb{P}(\mathbf{M}|\mathbf{\Phi})}[\mathcal{L}_{\text{CE}}(\mathcal{D}; \mathbf{W} \odot \mathbf{M})], \tag{7}$$

where $\mathbf{\Phi} = \{\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_G\}$ parameterizes independent variational factors $\mathbb{P}(\mathbf{m}_1|\boldsymbol{\phi}_1)$, …, $\mathbb{P}(\mathbf{m}_G|\boldsymbol{\phi}_G)$, with joint distribution $\mathbb{P}(\mathbf{M}|\mathbf{\Phi}) = \prod_{i=1}^{G} \mathbb{P}(\mathbf{m}_i|\boldsymbol{\phi}_i)$. Under this stochastic variational formulation problem, mask sampling can be reparameterized and continuously relaxed into a differentiable mapping with respect to $\mathbf{\Phi}$, making it possible to learn through gradient-based methods.

4.2.1 Variational Distribution Selection

Since pruning masks are N-hot vectors of length M, each mask can take one of $\binom{M}{N}$ possible values. Modeling an unconstrained distribution over all possible values therefore requires $\binom{M}{N} - 1$ free parameters, which grows combinatorially with M. To efficiently learn masks with a manageable number of parameters, we instead model mask distributions $\mathbb{P}(\mathbf{m}_i|\boldsymbol{\phi}_i)$ using the WRS distribution (Equation 1) over ordered subsets. Let $\mathcal{S}_i = \{\boldsymbol{\mu}_i^{(1)}, \ldots, \boldsymbol{\mu}_i^{(N)}\}$ be a set of N one-hot vectors corresponding to the selected weights within the i-th group, sampled from the WRS distribution using the Gumbel-Top-K trick. The probability of a mask $\mathbf{m}_i$ is then:

$$\mathbb{P}(\mathbf{m}_i|\boldsymbol{\phi}_i) = \sum_{\mathcal{S}_{\mathbf{m}_i}} \mathbb{P}_{\text{WRS}}(\mathcal{S}_{\mathbf{m}_i}|\exp(\boldsymbol{\phi}_i)), \tag{8}$$

where $\mathcal{S}_{\mathbf{m}_i}$ denotes the collection of ordered subsets whose elements sum to $\mathbf{m}_i$, i.e., $\mathbf{m}_i = \sum_{j=1}^{N} \boldsymbol{\mu}_i^{(j)}$. Exactly computing this probability is expensive and unnecessary, as the construction of $\mathbf{m}_i$ discards the ordering of items in the sampled subset. Instead, the expected loss can be computed via Monte Carlo sampling. Under this formulation, the original problem reduces to learning importance scores $\exp(\phi_{ij})$ to select salient weights so as to minimize the objective. Denoting the WRS distribution of an arbitrary subset $\mathcal{S}_i$ by $\mathbb{P}_{\text{WRS}}(\mathcal{S}_i|\boldsymbol{\phi}_i)$ for brevity, the optimization problem in Equation 7 can be reformulated as follows:

$$\mathbf{\Phi}^* = \operatorname*{argmin}_{\mathbf{\Phi}} \mathbb{E}_{\mathbb{P}_{\text{WRS}}(\mathcal{S}|\mathbf{\Phi})}[\mathcal{L}_{\text{CE}}(\mathcal{D}; \mathbf{W} \odot \mathbf{M})], \tag{9}$$

where $\mathcal{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_G\}$ is a collection of G subsets sampled from the joint distribution $\mathbb{P}_{\text{WRS}}(\mathcal{S}|\mathbf{\Phi}) = \prod_{i=1}^{G} \mathbb{P}_{\text{WRS}}(\mathcal{S}_i|\boldsymbol{\phi}_i)$. Each N-hot

pruning mask $\mathbf{m}_i$ in the mask collection $\mathbf{M}$ is constructed as $\mathbf{m}_i = \sum_{\boldsymbol{\mu}_i^{(j)} \in \mathcal{S}_i} \boldsymbol{\mu}_i^{(j)}$. Parameterizing masks as sums of subsets sampled from WRS-restricted distributions yields the same expected loss as sampling masks from the target distributions, as formalized in Theorem 1. Our approach reduces parameter complexity by viewing N-hot sampling as a sequential sampling without replacement process. Rather than maintaining a full categorical distribution over all $\binom{M}{N}$ possible configurations, we instead model only a single categorical distribution over the M model parameters, requiring exactly M parameters independent of N. The proposed method achieves a reduction in parameter complexity from $\mathcal{O}\left(\binom{M}{N}\right)$ to $\mathcal{O}(M)$, yielding an exponential improvement in memory efficiency.

4.2.2 Mask Selection Relaxation

To enable differentiation of the objective with respect to the variational distributions' parameters, we relax the discrete sampling process using the Gumbel-Top-K trick. Given logits $\boldsymbol{\phi}_i = [\phi_{i1}, \ldots, \phi_{iM}]$ forming a categorical distribution over the M consecutive model weights within the i-th group, the probability of selecting the j-th weight is achieved through the $\mathrm{softmax}$ function, $\pi_{ij} = \exp(\phi_{ij}) / \sum_{k=1}^{M} \exp(\phi_{ik})$. To sample a subset $\mathcal{S}_i$ without replacement from this distribution, we first perturb the logits with independent Gumbel noise to obtain random keys:

$$\kappa_{ij} = \phi_{ij} + g_{ij}, \quad g_{ij} \overset{\text{i.i.d}}{\sim} \mathrm{Gumbel}(0, 1). \tag{10}$$

We denote the adjusted keys for the i-th group at sampling step k by $\boldsymbol{\alpha}_i^{(k)} = [\alpha_{i1}^{(k)}, \ldots, \alpha_{iM}^{(k)}]$. The update rule is defined recursively. At each subsequent step k, the adjusted keys are updated by accumulating a correction term based on the previously selected probabilities as follows:

$$\begin{aligned} \boldsymbol{\alpha}_i^{(1)} &= [\kappa_{i1}, \ldots, \kappa_{iM}], \\ \boldsymbol{\alpha}_i^{(k)} &= \boldsymbol{\alpha}_i^{(k-1)} + \log(1 - \boldsymbol{\mu}_i^{(k-1)}). \end{aligned} \tag{11}$$

Finally, a differentiable relaxed one-hot vector $\boldsymbol{\mu}_i^{(k)} = [\mu_{i1}^{(k)}, \ldots, \mu_{iM}^{(k)}]$, representing the selected item at the k-th sampling step, is achieved by applying a $\mathrm{softmax}$ operation to the adjusted keys with temperature $\tau > 0$:

$$\mu_{ij}^{(k)} = \frac{\exp(\alpha_{ij}^{(k)} / \tau)}{\sum_{k=1}^{M} \exp(\alpha_{ik}^{(k)} / \tau)}. \tag{12}$$

After N sampling steps, a set of soft one-hot vectors representing selected weights is obtained. By summing up these vectors, a continuous relaxation of the N-hot pruning mask can be constructed, enabling gradient-based optimization of the objective.

4.2.3 Temperature Annealing

Having previously defined the temperature hyperparameter τ to control the sharpness of one-hot approximations, we now introduce a separate hyperparameter λ to explicitly govern the level of stochasticity in the sampling process. This is realized by applying the Gumbel-Top-K trick to the scaled logits, $\overline{\phi}_i = \phi_i / \lambda$, such that larger values of λ induce higher sampling randomness. In our experiments, we implement annealing schedules for both τ and λ to progressively guide the mask learning process, starting with high randomness to encourage broad exploration and gradually converging to a small set of high-confidence solutions toward the end of training. Specifically, we adopt exponential annealing schedules: at the t-th training step, the temperatures are:

$$\begin{aligned} \tau_t &= \max\left\{\tau_{\text{end}}, \tau_{\text{init}} \times \left(\frac{\tau_{\text{end}}}{\tau_{\text{init}}}\right)^{\frac{t}{T_{\text{anneal}}}}\right\}, \\ \lambda_t &= \max\left\{\lambda_{\text{end}}, \lambda_{\text{init}} \times \left(\frac{\lambda_{\text{end}}}{\lambda_{\text{init}}}\right)^{\frac{t}{T_{\text{anneal}}}}\right\}, \end{aligned} \tag{13}$$

with T_{anneal} being the number of annealing steps.

5 Experiment

5.1 Experimental Setting

LLM backbones: The proposed method is evaluated on the Qwen2.5 model family (Yang et al., 2024) at multiple scales, from 0.5B to 7B, to assess its stability and scalability under semi-structured pruning. All main experiments adopt a 2:4 sparsity pattern, compatible with NVIDIA hardware acceleration. Training is performed for 2,000 steps with a batch size of 256 and a sequence length of 4096, processing approximately 2B tokens in total. The training data is drawn from the Nemotron-CC-v2 corpus (NVIDIA et al., 2025), a clean English dataset suitable for pretraining modern LLMs. We detail hyperparameters in Appendix A.4.

Baselines: To assess the effectiveness of the proposed method, we select five representative semi-structured pruning baselines spanning both classical and modern approaches: Magnitude (Fang et al.,

Method	W/U	ARC-C	ARC-E	BoolQ	HellaS.	PIQA	RACE	SciQ	Average ↑
Base Model: Qwen2.5-0.5B	-	29.01	64.48	61.80	40.54	70.51	34.74	92.90	56.28
Magnitude	✗	19.11	31.19	38.38	26.65	54.13	23.06	43.40	33.70
Wanda	✗	19.80	42.85	56.82	28.93	59.03	26.70	83.20	45.33
SparseGPT	✓	19.97	47.22	59.85	30.80	60.12	27.85	84.30	47.16
ProxSparse	✗	21.42	46.63	61.63	32.00	61.63	29.09	79.80	47.46
MaskLLM	✗	22.16	54.02	**62.10**	32.28	63.41	30.19	85.90	50.01
SUSI (Ours)	✗	**23.63**	**54.97**	60.95	**32.34**	**63.82**	**31.00**	**87.20**	**50.56**
Base Model: Qwen2.5-1.5B	-	41.55	75.21	72.48	50.18	75.52	36.65	94.20	63.68
Magnitude	✗	20.48	32.74	39.42	27.20	57.07	24.59	60.10	37.37
Wanda	✗	24.06	51.26	**63.64**	32.36	62.46	29.28	88.30	50.19
SparseGPT	✓	28.33	56.69	62.66	34.54	65.07	**33.49**	89.00	52.83
ProxSparse	✗	29.25	59.55	42.87	**39.61**	67.25	32.73	89.80	51.58
MaskLLM	✗	**29.45**	**63.51**	61.33	39.10	69.07	32.51	**89.90**	54.98
SUSI (Ours)	✗	29.37	63.01	61.93	39.29	**69.31**	32.44	**89.90**	**55.04**
Base Model: Qwen2.5-3B	-	44.62	77.57	77.22	55.04	78.29	38.47	95.90	66.73
Magnitude	✗	23.81	24.66	37.83	25.98	51.96	20.77	18.50	29.07
Wanda	✗	27.13	60.02	62.26	35.88	67.14	34.35	91.00	53.97
SparseGPT	✓	31.40	66.12	62.66	41.05	69.75	35.12	92.50	56.94
ProxSparse	✗	-	-	-	-	-	-	-	-
MaskLLM	✗	**32.95**	66.42	65.11	43.18	70.44	**35.35**	**93.00**	**58.06**
SUSI (Ours)	✗	32.08	**66.71**	**65.26**	**43.30**	**70.57**	35.22	**93.00**	58.02
Base Model: Qwen2.5-7B	-	48.21	80.43	84.65	59.98	78.78	41.91	96.60	70.08
Magnitude	✗	24.40	38.34	64.16	30.05	55.60	23.25	74.30	44.30
Wanda	✗	38.99	72.18	73.21	44.53	71.87	35.79	94.50	61.58
SparseGPT	✓	**39.65**	73.54	75.83	45.89	73.38	36.67	94.80	62.82
ProxSparse	✗	38.69	74.46	74.77	48.63	73.22	36.08	**95.60**	63.06
MaskLLM	✗	39.27	73.68	81.55	**49.12**	74.88	37.46	95.10	64.44
SUSI (Ours)	✗	38.14	**74.20**	**82.14**	49.04	**75.35**	**37.80**	95.10	**64.54**

Table 1: Comparative evaluation of zero-shot accuracy across multiple benchmark datasets for various pruning methods on the Qwen2.5 model family at varying scales with 2:4 sparsity pattern. **Bold** and underlined values indicate the highest and second-highest performance, respectively. The 'W/U' column denotes whether weight updates are applied during pruning. ProxSparse fails to converge for the 3B model.

2024), Wanda (Sun et al., 2024), SparseGPT (Frantar and Alistarh, 2023), ProxSparse (Liu et al., 2025a), and MaskLLM (Fang et al., 2024). For ProxSparse, we use the same amount of training data as SUSI and MaskLLM to ensure fair comparisons, even though ProxSparse typically requires less data. Performance across different data scales is reported in Section 5.3.2, and while detailed methodological descriptions of baselines are provided in Appendix A.2.

5.2 Main Results

Following prior work, we begin by evaluating zero-shot accuracies on a broad suite of standard benchmark datasets (Table 1). Across all model scales and evaluation tasks, sparsity-mask learning approaches (ProxSparse, MaskLLM, and SUSI) consistently outperform more naive pruning baselines, with only minor exceptions. Among these methods, approaches that explicitly learn the pruning mask (MaskLLM and SUSI) via variational optimization substantially exceed deterministic regularization-based learning (ProxSparse), improving accuracy by 2.56[2] percentage points on average across model scales and benchmarks. Despite SUSI and MaskLLM both achieving comparable results (e.g. 64.54% vs. 64.44% average accuracy for the 7B model), SUSI incurs substantially lower pruning overhead. Specifically, for 2:4 sparsity, SUSI reduces the number of trainable parameters by approximately 33%, using only 1× the original model's parameter count, compared to 1.5× for MaskLLM (illustrated in Table 3).

Additionally, Table 2 reports perplexity (PPL) values on the WikiText-2 dataset, further highlight-

[2]This average is computed from the differences between the 'Average' accuracies of MaskLLM/SUSI and ProxSparse across all model scales reported in Table 1 (excluding Qwen2.5-3B where ProxSparse fails to converge).

Method	Model Sizes			
	0.5B	1.5B	3B	7B
w/o Pruning	18.45	12.59	12.32	9.03
Magnitude	3.0E+4	5.2E+3	1.6E+6	6.5E+3
Wanda	159.05	67.97	37.31	20.87
SparseGPT	73.78	38.95	22.82	16.44
ProxSparse	146.42	28.68	-	<u>14.91</u>
MaskLLM	<u>35.16</u>	<u>21.90</u>	<u>19.85</u>	<u>14.91</u>
SUSI (Ours)	**32.63**	**21.27**	**19.37**	**14.89**

Table 2: Perplexity (PPL) scores on WikiText-2 benchmark dataset and training flops (FLOPs) across backbone models with 2:4 sparsity pattern setting. **Bold** and <u>underlined</u> values indicate the best and second-best performance, respectively.

Method	Model Sizes			
	0.5B	1.5B	3B	7B
2:4 Sparsity				
ProxSparse	0.4B (1×)	1.3B (1×)	2.7B (1×)	6.5B (1×)
MaskLLM	0.6B (1.5×)	2.0B (1.5×)	4.0B (1.5×)	9.8B (1.5×)
SUSI (Ours)	0.4B (1×)	1.3B (1×)	2.7B (1×)	6.5B (1×)
2:8 Sparsity				
MaskLLM	1.4B (3.5×)	4.6B (3.5×)	9.5B (3.5×)	22.8B (3.5×)
SUSI (Ours)	0.4B (1×)	1.3B (1×)	2.7B (1×)	6.5B (1×)

Table 3: Number of trainable parameters (in billions) under different sparsity patterns and model sizes. Values in parentheses indicate the ratio relative to the original dense model parameters (excluding embeddings). For mask-learning methods, only mask parameters are trainable while the underlying model weights remain frozen.

ing the advantages of our proposed method: **(i) Effectiveness of Differentiable Subset Sampling:** SUSI consistently outperforms all competing baselines at every model scale, indicating that its differentiable subset sampling mechanism reliably discovers high-quality sparsity patterns that better preserve model performance after pruning. **(ii) Scalability across Model Sizes:** SUSI inherits the strong generalization properties of MaskLLM and delivers consistent improvements as the model scale increases, enabled by a closely related formulation and optimization strategy. SUSI maintains its advantage over simpler methods from small to large models: achieving 14.89 perplexity at 7B parameters, compared to 20.87 for Wanda and 16.44 for SparseGPT. This stable scaling behavior contrasts sharply with Magnitude pruning - which attains its best perplexity at 1.5B parameters rather than at 7B, further emphasizing SUSI's ability to maintain reliable performance as model capacity grows. In contrast, ProxSparse exhibits unstable optimization behavior at larger scales. In particular, when applied to the Qwen2.5-3B model, ProxSparse fails to reliably converge, preventing it from achieving competitive performance. **(iii) Robustness of Learnable Sparsity-Mask Approaches:** Magnitude pruning performs poorly (consistently yielding perplexity values exceeding 5,000), and other naive baselines fall far behind methods that explicitly learn the pruning mask, reaffirming that simple pruning strategies substantially impair language modeling performance. The strong results of SUSI and MaskLLM underscore the importance of principled, learnable pruning mechanisms.

SUSI and MaskLLM adopt similar problem formulations and largely comparable optimization procedures. Although SUSI overall outperforms MaskLLM in both zero-shot accuracy and perplexity, the raw performance gap remains relatively modest. The primary advantage of SUSI instead lies in its significantly lower training cost, as illustrated in Table 3, which compares the number of trainable parameters required by each method. In particular, the training cost of MaskLLM grows rapidly with model size. Even under 2:4 sparsity, MaskLLM requires 1.5× more trainable parameters than SUSI, a difference that may appear moderate in ratio but translates into a large absolute increase at scale. For the Qwen2.5-7B model, achieving 2:4 sparsity with MaskLLM requires ≈9.75B additional trainable parameters.

5.3 Detailed Analysis

5.3.1 2:8 Sparsity as a Failure Regime

To further evaluate the generality of SUSI, we extend our experiments beyond the commonly studied 2:4 sparsity to the 2:8 configuration. This stricter constraint exposes the challenges faced by semi-structured pruning methods, particularly those that do not learn pruning masks. Under this regime, we directly compare SUSI with MaskLLM to highlight pruning efficiency and effectiveness.

2:8 sparsity is exceptionally aggressive. As shown in Table 4, only methods that learn the pruning mask achieve reasonable perplexity values, while heuristic baselines collapse, producing perplexities exceeding 100. This behavior is congruent with the 2:4 setting and highlight the critical necessity of learnable mask parameterization. Accuracy results under 2:8 sparsity also exhibit the same trend and are deferred to Appendix A.9.

In conjunction with Table 3, we again observe

Method	Model Sizes			
	0.5B	1.5B	3B	7B
w/o Pruning	18.45	12.59	12.32	9.03
Magnitude	1.5E+7	1.0E+7	6.6E+7	infinity[3]
Wanda	8.1E+5	1.0E+6	3.2E+6	4.7E+4
SparseGPT	1.1E+4	4.0E+4	257.22	158.70
MaskLLM	63.11	38.14	38.89	34.72
SUSI (Ours)	**61.89**	**36.87**	**37.72**	**32.33**

Table 4: WikiText-2 Perplexity (PPL) scores across backbone models with 2:8 sparsity pattern setting. **Bold** and underlined values indicate the best and second-best performance, respectively. ProxSparse is excluded due to its inability to learn 2:8 sparsity patterns.

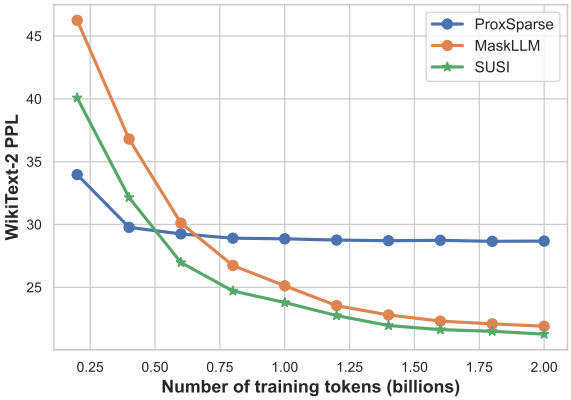

Figure 3: SUSI demonstrates better scalability than MaskLLM and ProxSparse, with WikiText-2 perplexity improving more consistently as the number of training tokens increases.

the rapid growth in training cost of MaskLLM as model size increases. Moreover, the parameter-requirement gap between SUSI and MaskLLM widens substantially under more aggressive sparsity. For the 7B model at 2:8 sparsity, MaskLLM requires more than twice the auxiliary parameters compared to the 2:4 setting. This corresponds to ≈22.75B additional parameters relative to SUSI, resulting in a 3.5× increase in parameters participating in memory allocation, backpropagation, optimizer and scheduler states, while only achieving perplexity levels comparable to SUSI.

These results demonstrate that differentiable subset sampling is sufficient to sustain language modeling capability under extreme sparsity, while requiring only 28.57% of the trainable parameters of the categorical parameterization. This establishes SUSI as the minimal learnable-mask formulation capable of sustaining non-degenerate performance.

5.3.2 Perplexity over Training Tokens

Among mask-learning pruning approaches, deterministic regularization-based learning fundamentally limits how effectively a model can exploit training data due to the bias-variance tradeoff. While regularization can accelerate early-stage optimization, it introduces a persistent bias that constrains convergence as training data grows. In contrast, directly learning sparsity masks in a variational manner avoids this limitation and is therefore intrinsically better suited to large-scale training regimes. This difference is directly reflected in Figure 3. During the early stage of training (until approximately 0.5B tokens), the regularization-based pruning baseline ProxSparse attains lower perplexity, indicating faster initial convergence. However, this advantage is temporary: as additional training data is incorporated, the performance of ProxSparse rapidly plateaus. Meanwhile, SUSI and MaskLLM continue to improve steadily and monotonically, consistently translating additional data into lower perplexity. These results provide clear empirical evidence that learning masks directly is critical for unlocking sustained improvements under large-scale training settings and achieving superior asymptotic performance.

[3]Undefined perplexity due to zero-probability predictions.

6 Conclusion

This work introduced SUSI, a novel semi-structured pruning technique that employs differentiable subset sampling to efficiently derive N:M sparsity masks. By substantially reducing the number of trainable parameters and associated memory overhead, SUSI enables effective compression of LLMs while maintaining strong downstream performance. Across Qwen2.5 models ranging 0.5 to 7B parameters, SUSI consistently achieves lower perplexity on WikiText-2 compared to existing pruning approaches, and maintains competitive zero-shot accuracy on a diverse set of benchmarks. Moreover, SUSI demonstrates enhanced data efficiency and favorable scalability as the amount of calibration data increases. Overall, these results position SUSI as a practical and scalable solution for semi-structured pruning of LLMs, offering a strong balancing between compression efficiency and performance preservation for resource-constrained deployment environments.

Limitations

Despite the promising performance and efficiency demonstrated by SUSI, several limitations remain: (i) First, the deployment of semi-structured sparsity is inherently hardware-dependent. At present, substantial throughput gains are realized only on select platforms (e.g., AMD ROCm and certain NVIDIA Ampere and Hopper GPUs) where 2:4 structured sparsity is natively supported and accelerated at the kernel level. Although SUSI can, in principle, be extended to arbitrary $N:M$ sparsity patterns, its practical utility is constrained by the absence of hardware kernels and vendor-optimized libraries for ratios other than 2:4. On accelerators or CPUs lacking such specialized support, pruning yields only marginal reductions in memory footprint and fails to deliver meaningful inference speed up. This hardware dependency poses a significant challenge for widespread adoption in heterogeneous production environments, where deployment targets may vary; (ii) Second, the current evaluation focuses exclusively on English-centric, dense Qwen2.5 models and a limited set of standard NLP benchmarks. Future research should investigate the applicability of SUSI to multilingual LLMs, different architectures like Mixture-of-Experts, larger-scale models, and domain-specific tasks (e.g., code generation, reasoning-intensive applications) to assess its generalization and scalability comprehensively.

References

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. 2024. Fluctuation-based adaptive structured pruning for large language models. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 10865–10873. AAAI Press.

Thomas Bird, Julius Kunze, and David Barber. 2018. Stochastic variational optimization. *CoRR*, abs/1809.04855.

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

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

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457.

Peijie Dong, Lujun Li, Zhenheng Tang, Xiang Liu, Xinglin Pan, Qiang Wang, and Xiaowen Chu. 2024. Pruner-zero: Evolving symbolic pruning metric from scratch for large language models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Pavlos S. Efraimidis and Paul G. Spirakis. 2006. Weighted random sampling with a reservoir. *Inf. Process. Lett.*, 97(5):181–185.

Gongfan Fang, Hongxu Yin, Saurav Muralidharan, Greg Heinrich, Jeff Pool, Jan Kautz, Pavlo Molchanov, and Xinchao Wang. 2024. Maskllm: Learnable semi-structured sparsity for large language models. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 10323–10337. PMLR.

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, and 5 others. 2024. The language model evaluation harness.

Emil Julius Gumbel. 1954. *Statistical theory of extreme values and some practical applications: a series of lectures*, volume 33. US Government Printing Office.

Weiyu Huang, Yuezhou Hu, Guohao Jian, Jun Zhu, and Jianfei Chen. 2025. Pruning large language models with semi-structural adaptive sparse training. In *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pages 24167–24175. AAAI Press.

Itay Hubara, Brian Chmiel, Moshe Island, Ron Banner, Joseph Naor, and Daniel Soudry. 2021. Accelerated sparse neural training: A provable and efficient method to find N: M transposable masks. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 21099–21111.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard H. Hovy. 2017. RACE: large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 785–794. Association for Computational Linguistics.

Qi Le, Enmao Diao, Ziyan Wang, Xinran Wang, Jie Ding, Li Yang, and Ali Anwar. 2025. Probe pruning: Accelerating llms through dynamic pruning via model-probing. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Hongyi Liu, Rajarshi Saha, Zhen Jia, Youngsuk Park, Jiaji Huang, Shoham Sabach, Yu-Xiang Wang, and George Karypis. 2025a. Proxsparse: Regularized learning of semi-structured sparsity masks for pretrained LLMS. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*. OpenReview.net.

Yijiang Liu, Huanrui Yang, Youxin Chen, Rongyu Zhang, Miao Wang, Yuan Du, and Li Du. 2025b. PAT: pruning-aware tuning for large language models. In *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pages 24686–24695. AAAI Press.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

NVIDIA, :, Aarti Basant, Abhijit Khairnar, Abhijit Paithankar, Abhinav Khattar, Adithya Renduchintala, Aditya Malte, Akhiad Bercovich, Akshay Hazare, Alejandra Rico, Aleksander Ficek, Alex Kondratenko, Alex Shaposhnikov, Alexander Bukharin, Ali Taghibakhshi, Amelia Barton, Ameya Sunil Mahabaleshwarkar, Amy Shen, and 198 others. 2025. Nvidia nemotron nano 2: An accurate and efficient hybrid mamba-transformer reasoning model. *Preprint*, arXiv:2508.14444.

Tobias Plötz and Stefan Roth. 2018. Neural nearest neighbors networks. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 1095–1106.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. 2024. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Jeffrey Scott Vitter. 1985. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57.

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

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2024. Sheared llama: Accelerating language model pre-training via structured pruning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Sang Michael Xie and Stefano Ermon. 2019. Reparameterizable subset sampling via continuous relaxations. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 3919–3925. ijcai.org.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 2024. Qwen2.5 technical report. *CoRR*, abs/2412.15115.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4791–4800. Association for Computational Linguistics.

Aojun Zhou, Yukun Ma, Junnan Zhu, Jianbo Liu, Zhijie Zhang, Kun Yuan, Wenxiu Sun, and Hongsheng Li. 2021. Learning N: M fine-grained structured sparse neural networks from scratch. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

A Appendix

A.1 WRS Yields Equivalent Variational Objective

Theorem 1. *Let $\mathbb{P}(\mathbf{m}_i|\phi_i)$ be the target distribution of each mask $\mathbf{m}_i$ as defined in Equation 8. The expected loss obtained by sampling each mask from its target distribution is equal to the expected*

loss obtained when each mask is parameterized as a sum of elements of an ordered subset $\mathcal{S}_i$ sampled from the corresponding restricted distribution $\mathbb{P}_{\text{WRS}}(\mathcal{S}_i|\boldsymbol{\phi}_i)$.

We show that:

$$\mathbb{E}_{\mathbb{P}(\mathbf{m}|\boldsymbol{\phi})}[f(\mathbf{m})] = \mathbb{E}_{\mathbb{P}_{\text{WRS}}(\mathcal{S}|\boldsymbol{\phi})}\left[f\left(\sum_{\boldsymbol{\mu}\in\mathcal{S}}\boldsymbol{\mu}\right)\right], \tag{14}$$

where $f(\mathbf{m})$ is an objective function of $\mathbf{m}$, and $\mathbb{P}(\mathbf{m}|\boldsymbol{\phi}) = \sum_{\mathcal{S}_\mathbf{m}} \mathbb{P}_{\text{WRS}}(\mathcal{S}_\mathbf{m}|\boldsymbol{\phi})$ with $\mathcal{S}_\mathbf{m}$ ranging over all sets whose elements sum to $\mathbf{m}$.

Proof. Given any set of binary masks $\mathcal{M}$ satisfying $N{:}M$ sparsity, by definition of $\mathbb{P}(\mathbf{m} \mid \boldsymbol{\phi})$:

$$\mathbb{E}_{\mathbb{P}(\mathbf{m}|\boldsymbol{\phi})}[f(\mathbf{m})] = \sum_{\mathbf{m}\in\mathcal{M}} \mathbb{P}(\mathbf{m} \mid \boldsymbol{\phi})\, f(\mathbf{m})$$

$$= \sum_{\mathbf{m}\in\mathcal{M}} \left(\sum_{\mathcal{S}_\mathbf{m}} \mathbb{P}_{\text{WRS}}(\mathcal{S}_\mathbf{m} \mid \boldsymbol{\phi})\right) f\left(\sum_{\boldsymbol{\mu}\in\mathcal{S}_\mathbf{m}}\boldsymbol{\mu}\right).$$

Reordering the sums yields:

$$\sum_{\mathbf{m}\in\mathcal{M}}\sum_{\mathcal{S}_\mathbf{m}} \mathbb{P}_{\text{WRS}}(\mathcal{S}_\mathbf{m} \mid \boldsymbol{\phi})\, f\left(\sum_{\boldsymbol{\mu}\in\mathcal{S}_\mathbf{m}}\boldsymbol{\mu}\right)$$

$$= \sum_{\mathcal{S}} \mathbb{P}_{\text{WRS}}(\mathcal{S} \mid \boldsymbol{\phi})\, f\left(\sum_{\boldsymbol{\mu}\in\mathcal{S}}\boldsymbol{\mu}\right),$$

which equals:

$$\mathbb{E}_{\mathbb{P}_{\text{WRS}}(\mathcal{S}|\boldsymbol{\phi})}\left[f\left(\sum_{\boldsymbol{\mu}\in\mathcal{S}}\boldsymbol{\mu}\right)\right].$$

□

A.2 Baseline Methods

i) **Magnitude** is a simple, data-free method that removes parameters with the smallest absolute values. While this method is easy to implement, it often yields subpar results due to the limitations of parameter sensitivity and model dynamics; ii) **Wanda** (Sun et al., 2024) combines parameter magnitudes with activation statistics at each layer, achieving better performance than pure magnitude pruning, especially under higher sparsity constraints, while maintaining favorable computational efficiency; iii) **SparseGPT** (Frantar and Alistarh, 2023) incorporates activation outputs and Hessian information to estimate parameter importance, followed by parameter updates to further reduce output error. This method can yield higher accuracy but is more computationally demanding; iv) **ProxSparse** (Liu et al., 2025a) leverages regularization to quickly find sparsity masks under very constrained data and computational settings. Owing to its deterministic optimization procedure, ProxSparse typically converges rapidly in the early stages of training. However, this early convergence can limit its ability to further explore the solution space, thereby restricting its capacity to fully exploit additional training data or extended optimization; and v) **MaskLLM** (Fang et al., 2024) is quite similar to the proposed method in this study, by learning pruning masks with minimizing calibration loss under an $N{:}M$ sparsity constraint, modeled via a multinomial distribution. It delivers strong performance across benchmarks but suffers from high computational cost.

A.3 Evaluation Metrics and Benchmark Datasets

Following previous works in this research field, three automated metrics are considered for the evaluation, including both quantitative and qualitative metrics to capture the full impact of pruning: i) *Task Accuracy (ACC)*: on common NLP tasks such as question answering in reading comprehension, mathematics, and science. These tasks are typically assessed in zero-shot or few-shot settings using benchmark datasets; *Perplexity (PPL)*: is a standard metric for assessing language model quality. It measures how well the model predicts the next word in a sequence, with lower values indicating better predictive performance.

Dataset	Questions	Task Type
ARC-Easy	2,376	Multiple-choice science
ARC-Challenge	1,172	Multiple-choice science
BoolQ	3,270	Yes/no questions
HellaSwag	10,042	Sentence completion
PIQA	1,838	Physical interaction QA
RACE	1,045	Multiple-choice comprehension
SciQ	1,000	Multiple-choice science

Table 5: Dataset statistics for zero-shot evaluation. We report the number of questions and the corresponding task type for each benchmark.

The benchmark datasets used to assess the effectiveness of pruning methods include WikiText-2 (Merity et al., 2017) for perplexity evaluation and a range of NLP benchmark datasets for zero-shot evaluation, which cover diverse task types and reasoning requirements, including ARC (Clark

et al., 2018), BoolQ (Clark et al., 2019), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), SciQ (Welbl et al., 2017), and RACE (Lai et al., 2017). All evaluations are conducted using the LM-Evaluation-Harness toolkit (Gao et al., 2024). Table 5 provides a comprehensive summary of the datasets used for zero-shot evaluation across multiple tasks. These datasets span a range of domains, including commonsense reasoning, science question answering, and reading comprehension, thereby ensuring a rigorous and diverse assessment of pruning performance.

A.4 Hyperparameter Setting

The hyperparameters used for training SUSI are listed in Table 6. These settings were carefully chosen to balance convergence stability and computational efficiency across all evaluated models.

Parameter	Values
Initialization distribution	$\mathcal{N}(0, 0.01)$
Gumbel-Softmax temperature	$\tau = 1.0 \rightarrow 0.05$
Sampling temperature	$\lambda = 1.0 \rightarrow 0.002$
Weight decay	0.05
Learning rate	$10^{-3} \rightarrow 10^{-4}$
AdamW parameters	$\beta_1 = 0.9, \beta_2 = 0.95$
Batch size	256
Sequence length	4096
Training steps	2000
Annealing steps	$T_{\text{anneal}} = 1500$

Table 6: Hyperparameter configuration used in training.

Specifically, model weights remain frozen during training. The variational distribution is initialized from a standard normal ($\mu = 0.0$, $\sigma = 0.01$), and a simulated annealing process gradually reduces randomness. For 1500 first training steps, temperatures τ and λ exponentially decay from 1.0 to 0.05 and from 1.0 to 0.002, respectively. Optimization uses AdamW with a learning rate decaying from 1×10^{-3} to 1×10^{-4}, weight decay value of 0.05, and β_1 = 0.9, β_2 = 0.95.

A.5 Gumbel-Top-K Algorithm

The algorithm for Gumbel-Top-K is illustrated in Algorithm 1. Specifically, we provide a clear description of the Gumbel-Top-K sampling procedure employed to enable differentiable mask learning. This formulation allows for efficient sampling of K items without replacement while preserving differentiability for gradient-based optimization.

Algorithm 1 Gumbel-Top-K Sampling Algorithm (Differentiable)

Input: Set of candidates $\mathcal{X} = \{x_1, \ldots, x_N\}$ with corresponding logits $\boldsymbol{\phi} = [\phi_1, \ldots, \phi_N]$, number of samples K, temperature $\tau > 0$

Output: Soft K-hot selection vector $\mathbf{S} \in \mathbb{R}^N$

for $i \leftarrow 1$ to N **do**
  $u_i \sim \text{Uniform}(0, 1)$
  $g_i \leftarrow -\log(-\log(u_i))$ {Gumbel noise}
  $\kappa_i \leftarrow \phi_i + g_i$ {Compute perturbed key}
end for
$\boldsymbol{\alpha}^{(1)} \leftarrow [\kappa_1, \ldots, \kappa_N]$
for $k \leftarrow 1$ to K **do**
  $\boldsymbol{\mu}^{(k)} \leftarrow \text{softmax}\big(\boldsymbol{\alpha}^{(k)}/\tau\big)$
  $\boldsymbol{\alpha}^{(k+1)} \leftarrow \boldsymbol{\alpha}^{(k)} + \log\big(1 - \boldsymbol{\mu}^{(k)}\big)$
end for
$\mathbf{S} \leftarrow \sum_{k=1}^{K} \boldsymbol{\mu}^{(k)}$ {Soft K-hot vector}
return S

A.6 Comparison with Straight-Through Gumbel-Top-K

We further examined whether adopting a straight-through (ST) Gumbel-Top-K estimator benefits pruning performance. In this variant, the forward pass generates discrete masks by directly applying argtop_K over Gumbel-perturbed logits, while the backward pass propagates gradients through the continuous Gumbel-Softmax relaxation. This strategy enforces discretization earlier in training, which should be able to improve mask interpretability. However, our empirical results in Table 7 show that ST Gumbel-Top-K leads to slightly inferior performance compared to the pure soft relaxation.

	Qwen2.5-1.5B		Qwen2.5-3B	
Metric	w STE	w/o STE (Ours)	w STE	w/o STE (Ours)
PPL (↓)	30.58	**21.27**	23.75	**19.37**
Avg. Acc (↑)	53.10	**55.04**	55.26	**58.02**

Table 7: Comparison of pruning results with 2:4 sparsity, both with and without ST Gumbel-Top-K estimator (denoted as "w STE" and "w/o STE").

For instance, on Qwen2.5-1.5B, ST achieves 30.58 perplexity and 53.10% average accuracy, while the soft approach (SUSI) reaches 21.27 perplexity and 55.04% accuracy. Similarly, on Qwen2.5-3B, the gap widens (23.57 vs. 19.37 perplexity). These observations suggest that the bias introduced by the ST estimator hampers generalization, outweighing the potential benefits of earlier

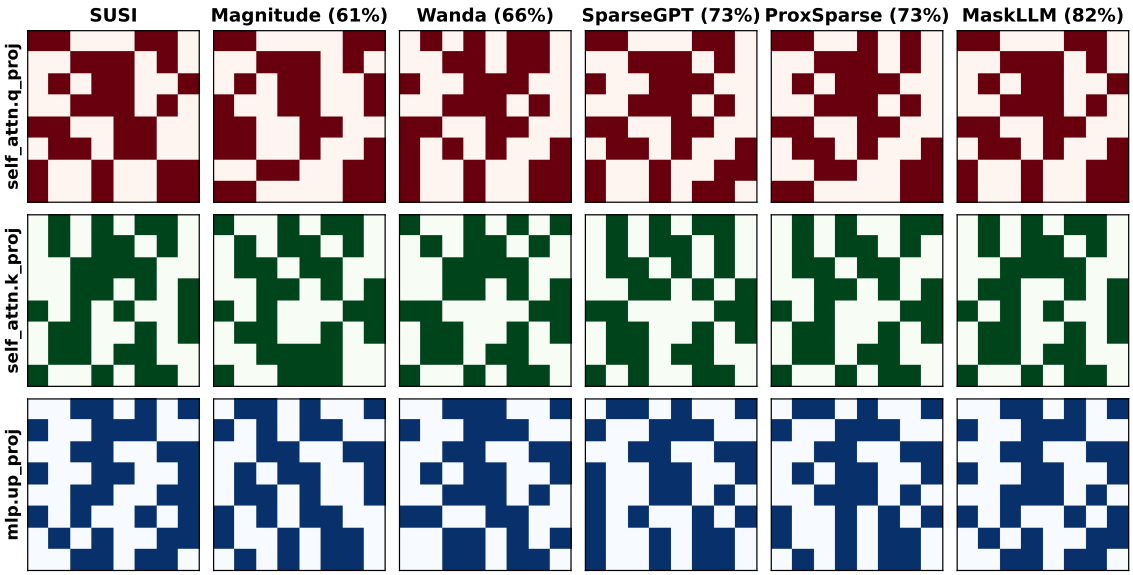

Figure 4: Mask difference analysis between SUSI and previous works on the Qwen2.5-3B model. Besides the name of each baseline, place an overlapping percentage indicating the similarity of the produced masks between that baseline and SUSI. Bold pixels indicate zero masks corresponding to pruned weights.

discretization. Overall, the soft Gumbel-Top-K relaxation used in SUSI provides a more effective balance between trainability and performance.

A.7 Throughput

Table 8 reports the decoding throughput of dense base models and their corresponding 2:4 sparse variants under different input/output token configurations. All models are deployed using vLLM with identical serving settings on a single NVIDIA B300 GPU to ensure a fair comparison.

I/O tokens	Models	Throughput	Acceleration
1024/4096	Qwen2.5-3B	190.60	1.0x
	Qwen2.5-3B 2:4	248.50	1.3x
	Qwen2.5-7B	165.00	1.0x
	Qwen2.5-7B 2:4	210.10	1.27x
2048/4096	Qwen2.5-3B	185.20	1.0x
	Qwen2.5-3B 2:4	241.40	1.3x
	Qwen2.5-7B	162.00	1.0x
	Qwen2.5-7B 2:4	204.40	1.26x

Table 8: Throughput comparison between dense base models and their 2:4 sparse variants. All models are deployed using vLLM on a single NVIDIA B300 GPU. Throughput is reported under different input/output token configurations. The 2:4 sparse models consistently achieve higher throughput than their dense counterparts, demonstrating effective exploitation of semi-structured sparsity at inference time.

Across both Qwen2.5-3B and Qwen2.5-7B, the 2:4 sparse variants consistently achieve higher throughput than their dense counterparts. For the 1024/4096 I/O setting, the 2:4 models yield up to 1.3× speedup for Qwen2.5-3B and 1.27× speedup for Qwen2.5-7B. Similar improvements are observed under the 2048/4096 configuration, indicating that the throughput gains are robust across different sequence lengths. These results demonstrate that semi-structured 2:4 sparsity can be effectively exploited at inference time, translating sparsity into tangible system-level acceleration without modifying the serving stack. Notably, the relative speedups remain stable across model scales, suggesting that the benefits of 2:4 sparsity generalize well from smaller to larger models.

A.8 Mask Difference Analysis

To investigate how different pruning strategies select weights, we measure the overlap between masks produced by various methods on the same model. Figure 4 presents a qualitative comparison of the sparsity masks produced by SUSI and several representative pruning baselines, including Magnitude, Wanda, SparseGPT, ProxSparse, and MaskLLM. For each method, we visualize the top-left 8×8 sub-matrix of the learned masks from three representative layers: `self_attn.q_proj`, `self_attn.k_proj`, and `mlp.up_proj`. Dark pixels denote zero entries corresponding to pruned weights. In addition, we report the overlapping percentage between the masks generated by each

baseline and SUSI, indicating the degree of structural similarity. Across all layers, SUSI exhibits a clear alignment with MaskLLM, as reflected by the higher overlap ratios. This suggests that SUSI is able to recover pruning patterns that are consistent with more expensive or data-intensive approaches, despite relying on a more efficient differentiable subset selection mechanism. In contrast, Magnitude pruning shows substantially lower overlap, indicating that purely weight-based criteria lead to structurally different sparsity patterns that deviate from those discovered by data-aware methods. Notably, ProxSparse achieves an overlap comparable to SparseGPT in certain layers, but still demonstrates noticeable structural discrepancies, particularly in attention projections. This observation is consistent with ProxSparse's deterministic and early-converging optimization behavior, which can restrict exploration of alternative sparsity configurations. Overall, this analysis highlights that SUSI not only improves quantitative performance but also produces structured masks that closely resemble those of strong, data-aware pruning baselines, reinforcing its effectiveness in learning meaningful semi-structured sparsity patterns.

A.9 Accuracy Results for 2:8 Sparsity

Table 9 re-illustrates the severity of the 2:8 sparsity constraint. On ARC-Challenge, pruned models of all sizes and methods fail to exceed the 25% random guess rate. Across the remaining benchmarks (Table 10), heuristic pruning methods remain statistically indistinguishable from random guessing, only approaches that explicitly learn the sparsity mask (SUSI and MaskLLM) are able to retain meaningful accuracy.

	Model Sizes			
Method	**0.5B**	**1.5B**	**3B**	**7B**
w/o Pruning	29.01	41.55	44.62	48.21
Magnitude	20.99	21.93	23.21	21.50
Wanda	19.80	19.62	19.37	19.88
SparseGPT	20.48	19.80	17.92	18.69
MaskLLM	18.88	21.51	20.81	21.41
SUSI (Ours)	18.26	21.33	20.48	21.33

Table 9: ARC-Challenge zero-shot accuracy scores across backbone models with 2:8 sparsity pattern setting. ProxSparse is excluded due to its inability to learn 2:8 sparsity patterns.

Among naive strategies, SparseGPT is the only method that preserves non-negligible performance, due to permitting parameter updates, and only on SciQ. These accuracy trends indicate that explicit mask learning is the most effective strategy. This observation is consistent with the 2:4 accuracy results in Table 1, as well as the 2:4 and 2:8 perplexity results reported in Tables 2 and 4. Collectively, these results further emphasize the advantages of SUSI over MaskLLM, showing that SUSI achieves competitive or superior performance at a fraction of the computational cost (Table 3), and remains robust across both tasks and sparsity regimes.

Method	W/U	ARC-E	BoolQ	HellaS.	PIQA	RACE	SciQ	Average ↑
Base Model: Qwen2.5-0.5B	-	64.48	61.80	40.54	70.51	34.74	92.90	60.83
Magnitude	✗	24.96	<u>38.72</u>	25.76	51.90	20.96	19.30	30.27
Wanda	✗	26.22	37.83	26.43	54.84	23.54	29.40	33.04
SparseGPT	✓	28.37	38.56	26.65	53.05	24.11	41.70	35.41
MaskLLM	✗	<u>42.75</u>	**61.65**	<u>26.92</u>	**59.42**	<u>25.86</u>	**73.80**	<u>48.40</u>
SUSI (Ours)	✗	**43.98**	**61.65**	**27.11**	<u>59.36</u>	**25.93**	<u>73.10</u>	**48.52**
Base Model: Qwen2.5-1.5B	-	75.21	72.48	50.18	75.52	36.65	94.20	67.37
Magnitude	✗	26.09	41.41	25.77	51.69	21.24	20.30	31.08
Wanda	✗	25.76	47.98	25.65	51.63	23.44	22.00	32.74
SparseGPT	✓	28.54	54.31	26.52	53.16	24.69	45.20	38.74
MaskLLM	✗	<u>50.12</u>	<u>58.87</u>	**30.20**	<u>63.32</u>	<u>27.80</u>	**77.40**	<u>51.29</u>
SUSI (Ours)	✗	**50.25**	**58.99**	<u>29.97</u>	**63.49**	**28.08**	<u>77.20</u>	**51.33**
Base Model: Qwen2.5-3B	-	77.57	77.22	55.04	78.29	38.47	95.90	70.42
Magnitude	✗	24.24	37.83	25.59	52.23	20.77	19.70	30.06
Wanda	✗	26.89	38.41	26.42	53.16	23.54	25.50	32.32
SparseGPT	✓	35.14	58.99	27.46	55.11	24.02	68.90	44.94
MaskLLM	✗	<u>50.57</u>	<u>61.91</u>	**30.70**	<u>64.08</u>	<u>28.22</u>	**82.00**	<u>52.91</u>
SUSI (Ours)	✗	**51.01**	**62.32**	<u>30.06</u>	**64.69**	**28.80**	<u>81.60</u>	**53.08**
Base Model: Qwen2.5-7B	-	80.43	84.65	59.98	78.78	41.91	96.60	73.73
Magnitude	✗	26.26	46.64	25.42	52.45	22.01	22.20	32.50
Wanda	✗	27.48	37.83	26.35	53.59	22.68	32.60	33.42
SparseGPT	✓	32.91	59.33	27.87	55.28	27.37	75.20	46.33
MaskLLM	✗	**51.20**	<u>60.98</u>	<u>29.32</u>	<u>62.90</u>	**28.98**	**83.60**	<u>52.83</u>
SUSI (Ours)	✗	<u>50.67</u>	**62.35**	**29.70**	**63.44**	<u>28.85</u>	<u>83.40</u>	**53.07**

Table 10: Comparative evaluation of zero-shot accuracy across multiple benchmark datasets for various pruning methods on the Qwen2.5 model family of different sizes with 2:8 sparsity pattern. **Bold** and <u>underlined</u> values indicate the highest and second-highest performance, respectively. ProxSparse is excluded due to its inability to learn 2:8 sparsity patterns.