Unlocking the Potential of Extremely Low-Bit Sparse Transformers through Adaptive Multi-bit Supermasks and Random Weights

Yasuyuki Okoshi [1] **Hikari Otsuka** [1] **Junnnosuke Suzuki** [1] **Daichi Fujiki** [1] **Masato Motomura** [1]

Abstract

We propose Adaptive Supermask (Ada-Sup), a scalable and efficient method for discovering high-quality multi-bit supermasks in an extended Strong Lottery Ticket framework. Building on this methods, we introduce TicketLLM, a Transformer-based model that combines pruning, quantization, and random weights to enable compact low-bit sparse representations. Experimental results show that Ada-Sup can find high quality supermasks with significantly reduced training cost in comparison to previous methods, both for binary and multi-bit supermask settings. Furthermore, TicketLLM outperforms BitNet b1.58 on a 1.3B parameter model with the same memory per connection, achieving 0.08 lower perplexity despite operating at a higher sparsity level (50% vs. 33%). These results demonstrate the potential of leveraging supermask and random weights as a practical and powerful alternative for building lightweight, scalable LLMs.

1. Introduction

As the number of parameters in large language models increases, model compression techniques are becoming more and more important for efficient deployment. In particular, directly analyzing scaling laws with model compression techniques has led to new insights into the trade-offs between performance and efficiency, enabling the discovery of new frontiers in model efficiency (Dettmers & Zettlemoyer, 2023; Wang et al., 2023; Ma et al., 2024; Chen et al., 2024; Kumar et al., 2024; Liu et al., 2025). However, existing analyses have focused almost on quantization or pruning, leaving other promising compression techniques underexplored.

[1]AI Computing Research Unit, Institute of Science Tokyo, Yokohama, Japan. Correspondence to: Yasuyuki Okoshi <okoshi.yasuyuki@artic.iir.isct.ac.jp>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

Table 1. Comparison of TicketLLM with other reproduced LLMs with 1.3B parameters. All models are trained with approximately 400B tokens with the same dataset sampled from FineWeb-Edu (Penedo et al., 2024). LLaMA* is configured LLaMA architecture by replacing linear projection layers with low-bit representations with RMSNorm.

Name	TicketLLM	BitNet b1.58	LLaMA
General Characteristics			
Base Arch.	LLaMA*	LLaMA*	LLaMA
Weight Distribution	{-3, -2, ..., 3}	{-1, 0, +1}	bf16
Compression Approach	SLT	Quantization	N/A
Training Method	Ada-Sup	QAT	N/A
Parameters			
Random Weights	{-1, +1}	N/A	N/A
Trained Weights	N/A	{-1, 0, +1}	bf16
Supermask	2-bit ({0,1,2,3})	N/A	N/A
Memory and Sparsity			
Memory per Connection	2-bit	2-bit	16-bit
Sparsity (%)	$\approx$ **50**	$\approx$ 30	N/A
Performance			
C4 PPL (1.3B)	**13.54**	13.62	11.68

The Strong Lottery Tickets (Zhou et al., 2019; Ramanujan et al., 2020) and their follow up study (Hirose et al., 2022) have introduced a novel compression paradigm that leverages randomness. This approach is characterized by three key features. First, it eliminates the need to store model weights by using fixed random weights, allowing model to utilize generated weights on-the-fly via a random number generator. Second, it introduces unstructured sparsity by uncovering an effective subnetwork within random weights using a pruning mask—known as a supermask. Third, since only the supermask needs to be stored, and it can be represented in binary, the overall model storage is significantly reduced. This novel paradigm has sparked considerable interests and found applications in diverse domains, including Graph Neural Networks (Huang et al., 2022), Folded Networks (García-Arias et al., 2023), one-layer Transformers (Shen et al., 2021), and large-scale vision applications (Okoshi et al., 2022).

Despite its potential, the impact of Strong Lottery Tickets (SLTs) on the scaling behavior of Transformers remains largely unexplored, primarily due to the lack of suitable optimization algorithms. In particular, there are two major challenges in applying SLTs to LLMs: (1) ensuring scala-

bility to complex tasks, and (2) reducing the training overhead associated with supermask optimization. Multicoated Supermasks (M-Sup) (Okoshi et al., 2022) addresses the scalability issue by introducing a multivalued mask, which allocates additional bits to leverage multiple supermasks with different sparsity levels. However, it still suffers from inefficiencies during training, mainly due to overhead for supermask generation.

To tackle these challenges, we propose Adaptive Supermasks (Ada-Sup), a scalable and efficient method for supermask optimization. Ada-Sup leverages a quantization-based approach for score parameters, which are updated during training to optimize the supermask. Building on this foundation, we introduce TicketLLM, a low-bit and sparse Transformer architecture enhanced with Ada-Sup, to investigate the scaling behavior of SLTs in LLMs. We first compare Ada-Sup with existing supermask optimization methods on Transformer architectures. Subsequently, we evaluate TicketLLM across varying amounts of training tokens and model sizes to assess its scalability and compare it against state-of-the-art low-bit sparse baselines. Our experimental findings can be summarized as follows:

- Compared with other supermask optimization methods, Ada-Sup achieves comparable performance with lower training cost in both the binary and multivalued supermask settings.
- Increasing supermask bits is effective in overtraining scenarios where the number of training tokens far exceeds the compute-optimal budget proposed by (Hoffmann et al., 2022).
- In situations where unstructured sparsity can be exploited in low-bit representations, allocating additional bits to the supermask is more effective than using them for sign representation.
- As summarized in Table 1, experiments on 1.3B Transformers shows that our proposal improves perplexity by 0.08 even with higher sparsity compared with BitNet b1.58, both using 2 bits per connection.

2. Strong Lottery Ticket

2.1. Formulation of SLT

As shown in Figure 1, SLT consists of random weights and supermask. Given an input row vector $\mathbf{x} \in \mathbb{R}^{1\times C_{\text{in}}}$, an output row vector $\mathbf{y} \in \mathbb{R}^{1\times C_{\text{out}}}$, and random weights $\mathbf{W}_{\text{rand}} \in \mathbb{A}^{C_{\text{in}}\times C_{\text{out}}}$, where $\mathbb{A}$ is an arbitrary distribution, the linear projection of SLT with the batch size of 1 is defined as follows:

$$\mathbf{y} = \mathbf{x}\left(\mathbf{W}_{\text{rand}} \odot \mathbf{M}\right), \tag{1}$$

where $\mathbf{M} \in \{0,1\}^{C_{\text{in}}\times C_{\text{out}}}$ is a supermask that uncovers the subnetwork of random weights. Instead of optimizing weights, SLT finds an accurate subnetwork by updating a pruning mask $\mathbf{M}$. Since it takes only zero or one, this optimization is NP-hard. To mitigate this issue, most prior works map score parameters into pruning masks. In this context, three notable techniques have emerged for supermask optimization: EdgePopup (Ramanujan et al., 2020), ProbMask (Zhou et al., 2021), and FixedTh (Koster et al., 2022).

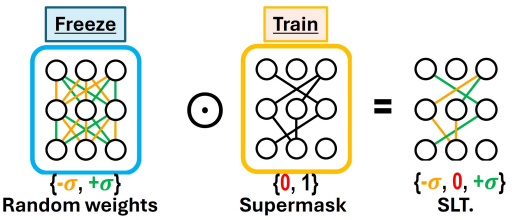


Figure 1. Structure of Strong Lottery Ticket (SLT) with randomly initialized weights and the supermask. Since random weights are sampled from Signed Kaiming Constant as discussed in Sec. 3.2, the subnetwork can represent ternary values with binary storage.

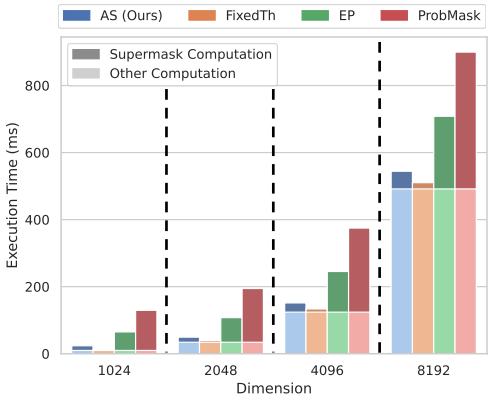


Figure 2. Overhead of supermask optimization from score parameters for each method. We include three baselines: EP (Ramanujan et al., 2020), ProbMask (Zhou et al., 2021), and FixedTh (Koster et al., 2022), in addition to our proposal (AS). We set the same input and output dimensions with all methods with batch size of 2048. Execution time is measured over 100 iterations using an NVIDIA GeForce RTX 3090.

Recently, Okoshi et al. has proposed Multicoated Supermasks (M-Sup), which expand supermask to take integer scalar values. In this method, Eq. (2) can be represented as

$$\mathbf{y} = \mathbf{x}\left(\mathbf{W}_{\text{rand}} \odot \sum_{i=0}^{N} \mathbf{M_i}\right), \tag{2}$$

where $\mathbf{M}_i$ is supermask whose sparsity is different and N is the number of supermasks. In this proposal, each supermask is optimized using EdgePopup using shared score parameters, introducing additional constraints of $(i+1)$-th supermask to be a subset of i-th supermask.

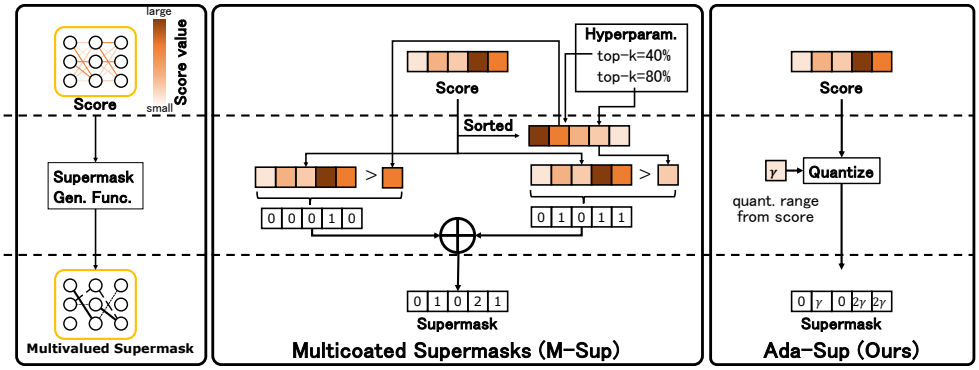


Figure 3. Overview of supermask generation methods. Supermask is generally optimized using score parameter assigned to each connectivity (left). M-Sup (Okoshi et al., 2022), a fundamental approach for multivalued mask, calculates supermask by selecting top-k% of score parameters, resulting in significant training overhead (center). Ada-Sup, in contrast, reduces computational overhead through score quantization-based approach (right).

2.2. Model Compression Perspective on SLT

This section briefly explains how SLTs are connected to the conventional model compression methods such as pruning and quantization.

SLTs can be interpreted as a special case of a concurrent blend of pruning and quantization. Specifically, SLTs sparsify the model by uncovering the subnetwork through the supermask, which determines the connectivity of random weights. This sparsity has a similar granularity to the unstructured sparsity in conventional pruning methods. SLT also achieves low-bit storage and computation through a mechanism different from quantization. Unlike conventional quantization, SLT achieves low-bit storage and computation through a fundamentally different mechanism. Storage efficiency is obtained by leveraging a binary supermask along with an appropriate architectural support for inference-time random weight generation, such as the one found Hirose et al., eliminating the need for explicit weight storage. Low-bit computation is enabled by sampling random weights from a binary distribution, which empirically yields optimal performance in SLT settings (Ramanujan et al., 2020). In consequence, SLT can effectively represent a ternary weighted model at the memory requirements of only the binary supermask. In the case of multivalued mask, only n-bits are required to represent a $(n+1)$-bits symmetric quantized distribution.

2.3. Motivation and Limitations of Existing SLT Methods

Existing frameworks suffer from either limited scalability to large-scale datasets such as ImageNet (e.g., FixedTh), poor training efficiency (e.g., EdgePopup), or both (e.g., ProbMask). Most prior works, including FixedTh and ProbMask, do not report results on ImageNet. Although EdgePopup is the only method among the three that provides ImageNet results, its accuracy using a single supermask is significantly lower than that of the original dense model (e.g., 68.6% vs. 77.1% on ResNet-50). While M-Sup pushes the limits of scalability by incorporating multivalued mask (e.g., 74.3% on ResNet-50), its reliance on EdgePopup still suffers from training overhead for supermask computation as shown in Figure 2.

These observations highlight a gap in existing SLT-based methods: achieving both model scalability and training efficiency remains challenging. To address this difficulty, we propose Adaptive Supermask (Ada-Sup), a novel supermask optimization framework that improves both the scalability and efficiency through quantized score-based mask generation.

3. Adaptive Supermask and its application to LLMs

3.1. Adaptive Supermask

In this section, we explain our proposal, Adaptive Supermasks (Ada-Sup) which is a quantization-based approach for optimizing supermasks. The brief overview of our proposal is described in Figure 3. As described in Sec. 2.1, supermask is optimized by updating score parameters. Given score parameters $\mathbf{S} \in \mathbb{R}_+^{C_{in} \times C_{out}}$, Ada-Sup calculates supermask by quantizing score parameters, as

$$\mathbf{M} = \gamma \lceil \mathrm{clip}(\mathbf{S}/\gamma, 0, 1) \rfloor. \tag{3}$$

Here, γ is a scaling factor that determines the clip range of scores, and $\lceil \cdot \rfloor$ is the round function. The $\mathrm{clip}(x, a, b)$ function clamps all elements to the range $[a, b]$. Please note supermask in Ada-Sup uses a scaled binary connectivity mask $\mathbf{M} \in \{0, \gamma\}$ different from original supermask to introduce the quantization scaling factor.

Ada-Sup can extend to a multivalued mask with n-bits representation by replacing the upper bound of the clip function

with $2^n - 1$ without additional operations:

$$\mathbf{M}_{\text{multi}} = \gamma \lceil \text{clip}(\mathbf{S}/\gamma, 0, 2^n - 1) \rfloor. \tag{4}$$

In the backward pass, we use the straight through estimators (Bengio et al., 2013) to compute the derivative of $\mathbf{M}$ concerning the score $\mathbf{S}$, as in EdgePopup.

Weight quantization often determines the scaling factor γ based on the distribution of weights. Following BitNet b1.58 (Ma et al., 2024), we compute γ as the mean of the absolute values of the parameters. Since score parameters are always non-negative, we can directly use their mean without taking absolute values:

$$\gamma = \frac{1}{MN} \sum_{i,j} |S_{ij}| = \frac{1}{MN} \sum_{i,j} S_{ij}. \tag{5}$$

3.2. Weight Initialization

In order to find accurate SLTs, the distribution of random weights is crucial. Previous research (Ramanujan et al., 2020; Okoshi et al., 2022) has demonstrated that the Signed Kaiming Constant (SKC), which samples from $\{-\sigma_{\text{KN}}, \sigma_{\text{KN}}\}$, yields the best performance (σ_{KN} is the standard deviation of the Kaiming Normal distribution (Han et al., 2015)). However, since the supermasks are already scaled in our method, as shown in Eq. (3), multiplying γ with σ_{KN} introduces a redundant operation. Therefore, we adopt a binary distribution $\{-1, +1\}$ for random weights.

3.3. Overall Architecture of TicketLLM

This section presents overall architecture of TicketLLM to verify effectiveness of Ada-Sup on LLMs. We follow the LLaMA (Touvron et al., 2023) to design the Transformer, including rotary positional embedding (RoPE) (Su et al., 2024) and gated linear unit (GLU) (Shazeer, 2020). The basic block of LLaMA consists of a multi-head attention (MHA) block and a feed-forward network (FFN) block with a residual connection. Different from the LLaMA architectures, we eliminate the pre-normalization layer for MHA and FFN since we introduce the RMSNorm to the input of the linear projection layer as discussed in Sec. 3.1. Thus, the output of a Transformer block is calculated as follows:

$$\begin{aligned} \mathbf{X}_{\text{mid}} &= \mathrm{X}_{\text{in}} + \text{MHA}(\mathbf{X}_{\text{in}}), \\ \mathbf{X}_{\text{out}} &= \mathrm{X}_{\text{mid}} + \text{FFN}(\mathbf{X}_{\text{mid}}). \end{aligned}$$

Here, $\mathbf{X}_{\text{in}}, \mathbf{X}_{\text{mid}}, \mathbf{X}_{\text{out}} \in \mathbb{R}^{T \times d}$ denote input sequence, output sequence of the MHA, and output sequence of FFN, respectively, where T is the sequence length and d is the model dimension.

TicketLLM replaces all linear projections in both MHA and FFN with Ada-Sup linear (ASL). Thus, given an input sequence $\mathbf{X}_{\text{in}}$, the MHA layer can be represented as

$$\text{MHA}(\mathbf{X}_{\text{in}}) = \phi\left(\frac{\text{ASL}_{\text{Q}}(\mathbf{X}_{\text{in}})\,(\text{ASL}_{\text{K}}(\mathbf{X}_{\text{in}}))^{\text{T}}}{\sqrt{d}}\right)\text{ASL}_{\text{V}}(\mathbf{X}_{\text{in}}),$$

where ϕ is a softmax function, and ASL_{Q}, ASL_{K}, and ASL_{V} are query, key, and value projections by Ada-Sup, respectively. Note that we omit the RoPE and assume the single-head attention for simplicity.

Based on the LLaMA architectures, we apply the FFN with GLU. Thus, $\mathbf{X}_{\text{out}} = \text{FFN}(\mathbf{X}_{\text{in}})$ is calculated using the following two steps:

$$\begin{aligned} \mathbf{X}_{\text{mid2}} &= \text{ASL}_1(\mathbf{X}_{\text{mid}}) \odot \sigma(\text{ASL}_2(\mathbf{X}_{\text{mid}})) \\ \mathbf{X}_{\text{out}} &= \text{ASL}_3(\mathbf{X}_{\text{mid2}}), \end{aligned}$$

where σ represents the sigmoid function. As an exception, we train only weights shared by the token embedding and the final linear projection.

4. Evaluation

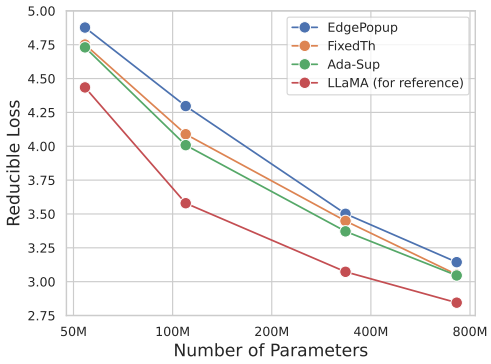
4.1. Experimental Setup

Transformer models are trained on randomly sampled subsets of FineWeb (Penedo et al., 2024) and evaluated on the C4 validation dataset (Raffel et al., 2020). All datasets are tokenized using the LLaMA2 tokenizer (Touvron et al., 2023), whose vocabulary size is $32K$. In order to ensure consistent training, tokens are concatenated into sequences of length 2048, where shorter sequences are combined and longer sequences are truncated.

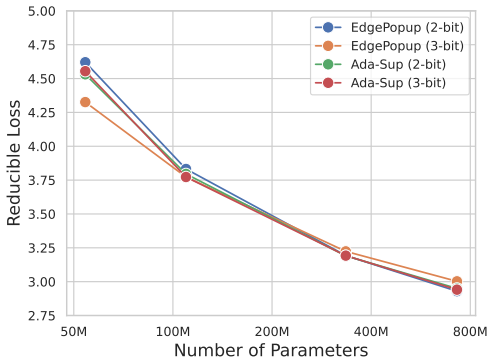
We vary the model size from 0.05B to 1.3B by increasing the number of layers and hidden dimensions while keeping the head dimension constant.

For pre-training, we determine the training token following a ratio of **t**okens **p**er model **p**arameters (TPP). We use Adam with decoupled weight decay (AdamW) (Loshchilov & Hutter, 2019), setting $\beta_1 = 0.95$, $\beta_2 = 0.99$, and a weight decay of 0.1. The learning rate is scaled with model size following Kaplan et al., and linearly decays to zero after completing the learning rate warmup in the first 1% of the total number of iterations. Although cosine decay is commonly used for pre-training, we adopt the recent learning rate schedule findings (Defazio et al., 2024; Anonymous, 2025). The batch size is 512, with gradient accumulation employed for larger models. Gradient clipping with 1.0 is also applied to stabilize training.

In addition to cross-entropy loss, we also evaluate perplexity and downstream accuracy for experiments involving models with larger parameters to provide a more comprehensive analysis of model performance in practical situations. Validation is performed once the training is completed using the latest checkpoint.



(a) Binary supermask



(b) Multivalued mask

Figure 4. Comparison of Ada-Sup with other supermask methods, including EdgePopup and FixedTh. To determine the γ in Eq. (3), we use the mean of scores for Ada-Sup. We also include the conventional weight learning (LLaMA) as a reference. We vary the number of parameters from 0.05B to 0.7B with a fixed TPP of 20.

We provide all model configurations and hyperparameters in Appendix A.

4.2. Comparison of Supermask Methods

This section compares Ada-Sup with other supermask methods, including EdgePopup and FixedTh. We also include reproduced LLaMA models as a baseline with the same training tokens. Following Ramanujan et al. and Koster et al., we set the sparsity to 50% for EdgePopup and use a fixed threshold of 0.01 for FixedTh, respectively. All models are trained with a fixed TPP of 20.

As shown in Figure 4 (a), Ada-Sup consistently outperforms other baselines except for LLaMA models across all parameters. Specifically, Ada-Sup reduces the loss by 0.05 compared to FixedTh and by 0.17 compared to EdgePopup on average, demonstrating that Ada-Sup discovers superior supermasks compared to other methods.

Figure 4 (b) compares Ada-Sup against M-Sup with different supermask bits. M-Sup is a multivalued mask optimization method where each supermask is optimized using EdgePopup. To align overall training iterations with our method, M-Sup with linear is used to determine multiple supermask sparsities of EdgePopup.

As shown in Figure 4 (b), both methods achieve comparable performance. For example, Ada-Sup achieves an evaluation loss of 2.95 compared to 2.92 from EdgePopup for 2-bit supermasks, while for 3-bit supermasks, the losses are 2.94 and 3.00, respectively. Despite comparable performance, Ada-Sup shows superior training efficiency. On 700M-parameter models trained with 20 TPPs, Ada-Sup takes a training time of approximately 40 H100 GPU hours for both 2-bit and 3-bit supermasks, while M-Sup with EdgePopup takes around 80 H100 GPU hours.

These results highlight that Ada-Sup not only matches performance in both the binary and multivalued mask settings but also significantly reduces the computational cost in multivalued mask methods, making it a more practical and scalable solution for applying supermask methods to LLMs.

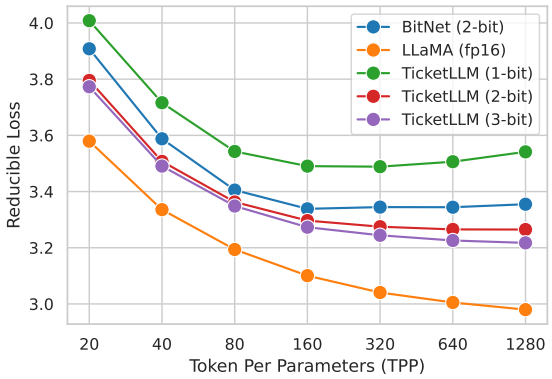


Figure 5. Model performance scale regarding training tokens. We compare different numerical representations with 0.1B parameters.

4.3. Exploring TicketLLM as Low-bit Sparse LLMs

Analysis of Dataset Scaling

This section explores how increasing the number of training tokens affects model performance across different low-bit sparse representations. To analysis dataset scaling, we compare the loss on the C4 validation dataset across a wide range of TPPs, from 20 to 1280, using different low-bit representations under a fixed model size of 0.1B parameters. We adopt BitNet b1.58 as the strong baseline for low-bit sparse representation in addition to the FP16 baseline of the LLaMA-based dense model.

We can observe three key findings from Figure 5. (1) When comparing TicketLLM models with different supermask bits, increasing supermask consistently improves perfor-

Table 2. Perplexity on C4-validation datasets and 0-shot evaluations on 10 downstream tasks. To support a wide variety of downstream tasks, we choose the 0-shot tasks from (Gadre et al., 2024) in addition to reported tasks in BitNet b1.58 (Ma et al., 2024) (denoted as BitNet in this table). All models are trained with 320 TPP. Sps. and PPL denote sparsity and perplexity.

Param.	Model	#Bit	Sps. (%)	PPL ↓	ARCe	ARCc	HS	OQ	BQ	PQ	WGe	COPA	MMLU	LLAMB	Avg. (%) ↑
0.1B	LLaMA	16	0	20.92	41.33	26.11	32.16	31.60	59.39	62.57	50.36	52.00	23.17	28.29	40.70
0.1B	BitNet	2	32	28.35	39.94	21.76	29.17	30.20	54.25	58.22	51.30	45.00	23.29	21.99	37.51
0.1B	TicketLLM	2	48	**26.44**	37.96	23.72	28.92	30.40	60.46	60.12	50.04	55.00	23.20	21.11	**39.19**
0.3B	LLaMA	16	0	16.14	50.29	29.61	42.53	36.20	53.43	67.14	51.93	59.00	23.17	38.09	45.34
0.3B	BitNet	2	32	20.22	44.40	26.88	36.01	31.20	59.45	64.36	53.28	58.00	23.38	30.18	42.51
0.3B	TicketLLM	2	48	**19.43**	43.73	25.85	36.14	34.00	58.04	63.17	50.04	58.00	23.29	30.22	**42.65**
0.7B	LLaMA	16	0	12.84	57.95	34.39	53.81	38.40	61.13	71.27	57.85	66.00	24.37	46.90	51.81
0.7B	BitNet	2	32	15.35	52.78	30.63	47.44	34.40	60.49	68.39	55.96	62.00	23.31	39.84	**47.42**
0.7B	TicketLLM	2	50	**15.06**	49.41	29.01	45.23	34.40	56.82	67.85	54.62	64.00	23.25	37.90	46.05
1.3B	LLaMA	16	0	11.68	62.92	37.29	59.87	40.20	62.26	73.01	59.67	69.00	24.18	51.52	53.99
1.3B	BitNet	2	33	13.62	55.39	32.68	54.51	38.40	59.24	70.35	57.30	66.00	23.80	41.78	**49.75**
1.3B	TicketLLM	2	50	**13.54**	54.29	33.02	52.03	36.40	59.76	70.35	54.93	66.00	23.64	42.89	49.53

mance. However, the performance gain from 2-bits to 3-bits is relatively small compared to the improvement from 1-bit to 2-bits. Our proposed method uses varying supermask bits to analyze the trade-offs between model size and performance.

(2) TicketLLM not only outperform the state-of-the-art (SOTA) quantization methods, but also shows better scaling trends regarding the dataset size under the same 2-bit model. While BitNet-b1.58 does not improve its performance beyond TPP=160, TicketLLM continues to benefit from the increased dataset. This comes from the extended search space of TicketLLM, fully leveraging 2-bit representation and expanded numerical representation.

(3) When comparing TicketLLM with a LLaMA-based dense model using FP16 weights, on the other hand, the loss gap gradually widens as the number of training tokens increases. This suggests that increasing the number of weight bits allows the model to retain more information from the training data. Improving model capacity to handle increasing data under low-bit representation remains a key challenge, and addressing this limitation is a promising direction for future research.

These results highlight the effectiveness of our proposal, particularly in low-bit settings with larger datasets.

Analysis of Parameter Scaling

This section compares the model performance when increasing parameters for the given TPP 320 to analyze parameter scaling on Transformer architecture. To achieve this goal, we evaluate perplexity on C4 validation datasets and 0-shot performance for ten downstream tasks, including ARC-easy, ARC-challenge (Yadav et al., 2019), HellaSwarg (Zellers et al., 2019), BoolQ (Clark et al., 2019), Open-bookQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), WinoGrande (Sakaguchi et al., 2021), COPA (Roemmele et al., 2011), MMLU (Hendrycks et al., 2021), and LAMBADA (Paperno et al., 2016). Table 2 summarize the results.

When comparing TicketLLM with BitNet b1.58, our method consistently outperforms in perplexity with higher sparsity. While BitNet b1.58 shows superior performance in downstream accuracy in certain parameters, these results highlight the strong potential of TicketLLM for efficient large language models.

Although our proposed method achieves lower perplexity compared to BitNet b1.58 across all model sizes, it still underperforms the reproduced LLaMA models in both perplexity and downstream accuracy. However, we observe distinct trends in performance scaling across perplexity and downstream tasks. For perplexity, the performance gap between our model and LLaMA narrows as the number of parameters increases, suggesting improved text generation quality at larger parameters. In contrast, on downstream tasks, the performance gap tends to widen with increasing parameters, implying that TicketLLM may currently have limitations in leveraging knowledge of pre-trained datasets as effectively as dense models. These results highlight the need for further development of scalable training strategies to achieve strong performance on both perplexity and downstream tasks while maintaining model efficiency.

5. Conclusion

We introduced Adaptive Supermasks (Ada-Sup), an efficient supermask optimization method designed to support extremely low-bit sparse representations. Being built on this method, we developed TicketLLM, a Transformer architecture that integrates Ada-Sup to achieve low-bit compression without training weights. Experimental results show that TicketLLM, powered by Ada-Sup, outperforms existing low-bit baselines such as BitNet b1.58 even with higher sparsity. These findings underscore the potential of SLTs for enabling efficient low-bit representations, offering a promising solution for scalable model compression in LLMs through the use of concurrent blend of pruning, quantization and random weights.

Acknowledgment

This work was carried out using the TSUBAME4.0 supercomputer at Institute of Science Tokyo. This work was supported in part by JSPS KAKENHI Grant Numbers JP23H05489, JP25K03092, and JP23KJ0955, and by JST-ALCA-Next Japan Grant # JPMJAN24F3.

References

Anonymous. Straight to zero: Why linearly decaying the learning rate to zero works best for LLMs. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=hrOlBgHsMI.

Bengio, Y., Léonard, N., and Courville, A. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Bisk, Y., Zellers, R., Gao, J., Choi, Y., et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, pp. 7432–7439, 2020.

Chen, X., Hu, Y., Zhang, X., Wang, Y., Li, C., Chen, H., and Zhang, J. P˙2 law: Scaling law for post-training after model pruning. *arXiv preprint arXiv:2411.10272*, 2024.

Clark, C., Lee, K., Chang, M.-W., Kwiatkowski, T., Collins, M., and Toutanova, K. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In Burstein, J., Doran, C., and Solorio, T. (eds.), *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)*, pp. 2924–2936, 2019.

Defazio, A., Yang, X. A., Khaled, A., Mishchenko, K., Mehta, H., and Cutkosky, A. The road less scheduled. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Dettmers, T. and Zettlemoyer, L. The case for 4-bit precision: k-bit inference scaling laws. In *International Conference on Machine Learning*, pp. 7750–7774. PMLR, 2023.

Gadre, S. Y., Smyrnis, G., Shankar, V., Gururangan, S., Wortsman, M., Shao, R., Mercat, J., Fang, A., Li, J., Keh, S., et al. Language models scale reliably with over-training and on downstream tasks. *arXiv preprint arXiv:2403.08540*, 2024.

García-Arias, Á. L., Okoshi, Y., Hashimoto, M., Motomura, M., and Yu, J. Recurrent residual networks contain stronger lottery tickets. *IEEE Access*, 11:16588–16604, 2023. doi: 10.1109/ACCESS.2023.3245808.

Han, S., Pool, J., Tran, J., and Dally, W. J. Learning both weights and connections for efficient neural networks. In *Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 1*, pp. 1135–1143, 2015.

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*, 2021.

Hirose, K., Yu, J., Ando, K., Okoshi, Y., García-Arias, L., Suzuki, J., Chu, T. V., Kawamura, K., and Motomura, M. Hiddenite: 4k-pe hidden network inference 4d-tensor engine exploiting on-chip model construction achieving 34.8-to-16.0tops/w for cifar-100 and imagenet. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, volume 65, pp. 1–3, 2022.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Vinyals, O., Rae, J. W., and Sifre, L. An empirical analysis of compute-optimal large language model training. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022.

Huang, T., Chen, T., Fang, M., Menkovski, V., Zhao, J., Yin, L., Pei, Y., Mocanu, D. C., Wang, Z., Pechenizkiy, M., and Liu, S. You can have better graph neural networks by not training weights at all: Finding untrained GNNs tickets. In *The First Learning on Graphs Conference*, 2022.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Koster, N., Grothe, O., and Rettinger, A. Signing the supermask: Keep, hide, invert. In *International Conference on Learning Representations*, 2022.

Kumar, T., Ankner, Z., Spector, B. F., Bordelon, B., Muennighoff, N., Paul, M., Pehlevan, C., Ré, C., and Raghunathan, A. Scaling laws for precision. *arXiv preprint arXiv:2411.04330*, 2024.

Liu, Z., Zhao, C., Huang, H., Chen, S., Zhang, J., Zhao, J., Roy, S., Jin, L., Xiong, Y., Shi, Y., et al. Paretoq: Scaling laws in extremely low-bit llm quantization. *arXiv preprint arXiv:2502.02631*, 2025.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Ma, S., Wang, H., Ma, L., Wang, L., Wang, W., Huang, S., Dong, L., Wang, R., Xue, J., and Wei, F. The era of 1-bit llms: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764*, 2024.

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 Riloff, E., Chiang, D., Hockenmaier, J., and Tsujii, J. (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, October-November 2018.

Okoshi, Y., García-Arias, A. L., Hirose, K., Ando, K., Kawamura, K., Van Chu, T., Motomura, M., and Yu, J. Multi-coated supermasks enhance hidden networks. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 17045–17055. PMLR, 17–23 Jul 2022.

Paperno, D., Kruszewski, G., Lazaridou, A., Pham, N. Q., Bernardi, R., Pezzelle, S., Baroni, M., Boleda, G., and Fernández, R. The LAMBADA dataset: Word prediction requiring a broad discourse context. In Erk, K. and Smith, N. A. (eds.), *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1525–1534, August 2016.

Penedo, G., Kydlíček, H., allal, L. B., Lozhkov, A., Mitchell, M., Raffel, C., Werra, L. V., and Wolf, T. The fineweb datasets: Decanting the web for the finest text data at scale. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

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.

Ramanujan, V., Wortsman, M., Kembhavi, A., Farhadi, A., and Rastegari, M. What's hidden in a randomly weighted neural network? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11893–11902, 2020.

Roemmele, M., Bejan, C. A., and Gordon, A. S. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *2011 AAAI spring symposium series*, 2011.

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106, August 2021.

Shazeer, N. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Shen, S., Yao, Z., Kiela, D., Keutzer, K., and Mahoney, M. What's hidden in a one-layer randomly weighted transformer? In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, November 2021.

Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding. *Neurocomput.*, 568(C), February 2024. ISSN 0925-2312.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Wang, H., Ma, S., Dong, L., Huang, S., Wang, H., Ma, L., Yang, F., Wang, R., Wu, Y., and Wei, F. Bitnet: Scaling 1-bit transformers for large language models. arxiv. *arXiv preprint arXiv:2310.11453*, 3, 2023.

Yadav, V., Bethard, S., and Surdeanu, M. Quick and (not so) dirty: Unsupervised selection of justification sentences for multi-hop question answering. In Inui, K., Jiang, J., Ng, V., and Wan, X. (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2578–2589. Association for Computational Linguistics, November 2019.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. HellaSwag: Can a machine really finish your sentence? In Korhonen, A., Traum, D., and Màrquez, L. (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800. Association for Computational Linguistics, July 2019.

Zhou, H., Lan, J., Liu, R., and Yosinski, J. Deconstructing lottery tickets: Zeros, signs, and the supermask. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Zhou, X., Zhang, W., Xu, H., and Zhang, T. Effective sparsification of neural networks with global sparsity constraint.

In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3599–3608, 2021.

A. Training configurations

We provide all model configurations in Table 3 and hyperparameters in Table 4.

Table 3. Model configuration. We scale the number of layers and model dimensions while head dimension keeps constant.

N	n_{layers}	n_{heads}	d_{model}	d_{head}	d_{FFN}
0.1B	12	12	768	64	2,048
0.3B	24	16	1,024	64	2,731
0.7B	24	24	1,536	64	4,096
1.3B	24	32	2,048	64	5,460

Table 4. Hyperparameters for training. Batch size is denoted as BS, while learning rate is described as LR. Warmup and Steps describe their respective number of iteration. Tokens means the number of training tokens.

Model	TPP	LR	BS	#Warmup	#Steps	#Tokens
0.1B	320	6.6e-4	512	334	33,438	32B
0.3B	320	5.0e-4	512	1,022	102,216	56B
0.7B	320	3.9e-4	512	2,224	222,434	223B
1.3B	320	3.1e-4	512	3,886	388,680	408B