SGA-SPARSE: TRAINING NATIVE SWIGLU GATES FOR ACTIVATION-SPARSE AND LOW-BIT LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Activation sparsity can reduce FFN weight traffic during single-user LLM decoding, but active channels must be identified early enough to avoid fetching unused weights. In pretrained SwiGLU models, training-free routing with the native gate performs substantially worse than routing with the up activation at high sparsity. We show that this ordering reverses after sparsity-aware continued pretraining. To the best of our knowledge, we present the first systematic study that converts pretrained SwiGLU LLMs into fixed-budget activation-sparse models by training the existing gate projection as the router. Our method, **SGA-Sparse**, updates all model parameters through continued pretraining while preserving the original SwiGLU activation and introducing no auxiliary router parameters. After approximately 30B training tokens, LLaMA models from 1B to 8B achieve uniform per-token 75–85% FFN channel sparsity while remaining within 0.2–2.7 pp of their original dense checkpoints in average zero-shot accuracy. We further study two complementary QAT recipes: continued-QAT largely recovers the performance at 4-bit weight quantization, while joint-QAT performs better for the more aggressive sub-4-bit target. To compare activation sparsity with dense low-bit quantization, we translate the sparsity-induced reduction in weight traffic into an equivalent bit-width. Under this common idealized metric, our 2.5-bit-equivalent models achieve average zero-shot accuracy comparable to dense 3-bit QAT models, while our 1.5-bit-equivalent models approach dense 2-bit models and outperform dense ternary models by 1.3 pp on average. These results establish native-gate training as a practical approach for combining high FFN sparsity with low-bit weights.

1 INTRODUCTION

Local inference of Large language model (LLM) is becoming increasingly important as on-device personal intelligence receives growing attention (Liu et al., 2024; Agarwal et al., 2025; Chen et al., 2026). The throughput of serving LLM locally is commonly bounded by the autoregressive decoding for a single user. It's speed is limited by moving model weights rather than by peak arithmetic throughput. Feed-forward networks (FFNs) account for a large fraction of these weights, although only a token-dependent subset of their intermediate channels may be important (Liu et al., 2023). Activation sparsity exploits this property by loading only the weights for the active input/output channels (Liu et al., 2025; Szatkowski et al., 2026). The key challenge is to identify these channels early enough to reduce the weight traffic without degrading the quality of the dense model.

Selecting these channels is particularly challenging in SwiGLU-based FFNs. The post-gating activation, i.e., the input to the down projection, provides the most accurate channel-importance signal (Szatkowski et al., 2026). However, it is available only after both the up and gate projections have been computed, so it can save only the down projection. An auxiliary predictor (Liu et al., 2023) can select channels before evaluating any of the three FFN projections. However, it introduces additional parameters and inference-time overhead. Existing predictor-based approaches have also been demonstrated primarily on ReLU-based FFNs, where exact zeros provide a clear prediction target (Shrestha et al., 2026; Song et al., 2025). The native gate and up signals provide an intermediate choice: one branch is computed densely and used as the router to select a common channel set for the other branch and the down projection.

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

Recent training-free studies expose an important limitation of gate-based routing (Liu et al., 2025; Szatkowski et al., 2026). In a SwiGLU FFN, the gate signal ignores the up-projection output. A channel with a small gate value can therefore remain important when its up activation has a large magnitude. Existing methods address this problem in different ways. CATS applies training-free magnitude thresholding to the gate signal (Lee et al., 2024), but its performance degrades substantially in the high-sparsity setting. Other methods replace SiLU with ReLU and retrain the model to induce intrinsic zeros (Mirzadeh et al., 2024; Song et al., 2025), at the cost of modifying the pretrained model architecture.

Based on our initial study, we observe the same limitation in the training-free setting, where the gate signal performs substantially worse than the up signal at high sparsity. Surprisingly, we find this ordering changes under sparsity-aware training. Our method, **SGA-Sparse** , continues training all model parameters using the native gate as a fixed top-k router. **SGA-Sparse** preserves the pretrained SwiGLU activation function and introduces no additional router parameters. Across models ranging from 1B to 8B parameters, **SGA-Sparse** achieves 75–85% FFN sparsity with approximately 30B training tokens while remaining within 2.7 pp of the dense models.

We further find that, unlike observations in prior training-free settings (Liu et al., 2025; Xu et al., 2025; Yin et al., 2026), post-training quantization (PTQ) does not preserve the performance of sparsity-trained checkpoints. We therefore study two complementary strategies: continued-QAT, which applies quantization-aware training (QAT) after sparsity-aware training, and joint-QAT, which trains sparsity and quantization together. Finally, we convert activation sparsity into an equivalent weight-traffic bit width. This allows us to compare sparse W_4 and mixed W_2/W_4 **SGA-Sparse** models directly with dense sub-4-bit QAT models on a common accuracy-traffic Pareto frontier. Our key contributions are as follows:

- To the best of our knowledge, we present the first systematic study that converts pretrained SwiGLU LLMs into fixed-budget activation-sparse models by training the original native gate as the router. **SGA-Sparse** updates the full model through continued pretraining while preserving the pretrained SwiGLU activation and introducing no auxiliary router parameters. With approximately 30B training tokens, **SGA-Sparse** achieves uniform per-token 75–85% FFN channel sparsity across LLaMA-3 models ranging from 1B to 8B parameters, with only a 0.2–2.7 pp accuracy gap from the corresponding dense models.
- We show that the best native routing signal changes with training. Although the up signal performs substantially better on frozen models, the native gate becomes the strongest deployable signal after activation-sparsity-aware continued pretraining.
- We show that PTQ consistently degrades sparsity-trained checkpoints and develop two QAT recipes: continued-QAT and joint-QAT—to combine activation sparsity with low-bit weights. By expressing sparsity as an equivalent weight-traffic bit width, we compare quantized **SGA-Sparse** and dense QAT models on a common accuracy–traffic Pareto curve. Our 2.5- and 1.5-bit-equivalent **SGA-Sparse** models introduce new Pareto-efficient operating points among the evaluated models, achieving accuracy comparable to dense 3- and 2-bit QAT models, respectively.

2 Related Work

Activation sparsity exploits token-dependent activation patterns to reduce computation and weight traffic during memory-bound LLM decoding (Liu et al., 2023; Mirzadeh et al., 2024; Liu et al., 2025). Existing methods differ mainly in where they obtain the sparsity signal and whether they modify the pretrained FFN architecture.

Auxiliary predictors and activation modification. ReLU-based Transformers naturally contain exact zeros in their FFN activations (Li et al., 2022). DejaVu (Liu et al., 2023) trains auxiliary predictors with a local binary classification objective to identify active channels before executing the FFN projections. However, auxiliary predictors introduce additional parameters and inference overhead, and extending them to SwiGLU FFNs remains challenging (Shrestha et al., 2026). ReLUfication (Mirzadeh et al., 2024) instead replaces SiLU with ReLU and continues training the model to produce intrinsic activation sparsity. ProSparse (Song et al., 2025) and TwELL (Cetin et al., 2026) further increase this sparsity through additional regularization. These approaches can support

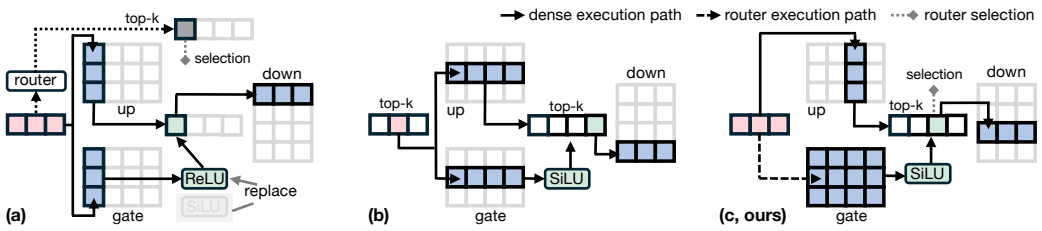


Figure 1: Comparison of different FFN activation-sparsity strategies: (a) auxiliary prediction with activation modification, (b) input activation sparsity, and (c) native-gate routing used by **SGA-Sparse** .

high activation sparsity, but they change the activation function of the pretrained model. Figure 1(a) illustrates this family of methods.

Input activation sparsity. TEAL (Liu et al., 2025) sparsifies the inputs to the up, gate, and down projections using activation magnitude. R-Sparse (Zhang et al., 2025) and SPON (Xu et al., 2025) compensate for the resulting information loss using low-rank residual paths and learned biases, respectively. DuoGPT (Yin et al., 2026) combines input activation sparsity with per-channel weight sparsity. These methods operate without additional training. Q-Sparse (Wang et al., 2024) instead incorporates input activation sparsity during low-bit model training. As illustrated in Figure 1(b), applying input sparsity to every FFN projection sparsifies both the input and output dimensions of the up and gate projections. We refer to this as *sparsity overlap*. Our experiments show that input-sparsity methods experience larger accuracy degradation than native-gate routing at high sparsity.

Native-gate routing. CATS (Lee et al., 2024) is the closest prior work to ours. It uses the magnitude of the native post-SiLU gate activation and a calibrated threshold to select FFN channels without training. The gate projection is evaluated densely, and its output is reused to execute selected channels of the up and down projections. This approach preserves the original SwiGLU activation and requires no auxiliary router parameters. However, prior studies find that the gate is less accurate than the up activation as a routing signal in the training-free setting (Liu et al., 2025; Szatkowski et al., 2026). **SGA-Sparse** studies a different setting: we apply fixed-budget top-k routing and continue training all model parameters. We show that this training can reverse the signal ordering and turn the native gate into an effective router at high sparsity, without changing the pretrained SwiGLU activation.

Activation sparsity and quantization. Several training-free sparsity methods report that activation sparsity can be combined with PTQ with little additional degradation (Zhang et al., 2025; Liu et al., 2025; Xu et al., 2025; Yin et al., 2026). Q-Sparse (Wang et al., 2024) jointly trains input activation sparsity with binary or ternary weights, but does not study native-gate routing or the effect of quantizing a sparsity-trained checkpoint. ParetoQ (Liu et al., 2026) provides strong dense low-bit QAT models across several precision levels. Our work studies how PTQ and QAT interact with sparsity-aware native-gate training and compares the resulting models with dense models trained with ParetoQ using a common equivalent weight-traffic metric.

3 Preliminaries

Notation. Throughout this paper, column vectors are denoted by bold lowercase letters and matrices by bold uppercase letters. We denote a token hidden state by $\mathbf{x} \in \mathbb{R}^d$. The projection matrices are $\mathbf{W}^{\text{up}}, \mathbf{W}^{\text{gate}} \in \mathbb{R}^{n \times d}$ and $\mathbf{W}^{\text{down}} \in \mathbb{R}^{d \times n}$, where d and n denote the hidden and FFN intermediate dimensions, respectively.

Router-based activation sparsity in SwiGLU FFNs. A dense SwiGLU FFN computes

$$\mathbf{y} = \mathbf{W}^{\text{down}} \left(\text{SiLU}(\mathbf{W}^{\text{gate}}\mathbf{x}) \odot \mathbf{W}^{\text{up}}\mathbf{x} \right). \tag{1}$$

We denote the gate activation, up activation, and post-gating activation by $\mathbf{g} = \text{SiLU}(\mathbf{W}^{\text{gate}}\mathbf{x})$, $\mathbf{u} = \mathbf{W}^{\text{up}}\mathbf{x}$, and $\mathbf{h} = \mathbf{g} \odot \mathbf{u}$, respectively. We apply activation sparsity by multiplying the post-gating activation $\mathbf{h}$ element-wise by a binary mask $\mathbf{m} \in 0, 1^n$. The post-gating signal $\mathbf{h}$ contains information from both branches, but it becomes available too late to skip either input projection. We therefore

consider the native gate and up activations as early routing signals. For gate-based routing, we select

$$\mathcal{I}_g = \text{top-k}(|\mathbf{g}|, k), \tag{2}$$

where k is the number of retained intermediate channels. The resulting sparse FFN computation is:

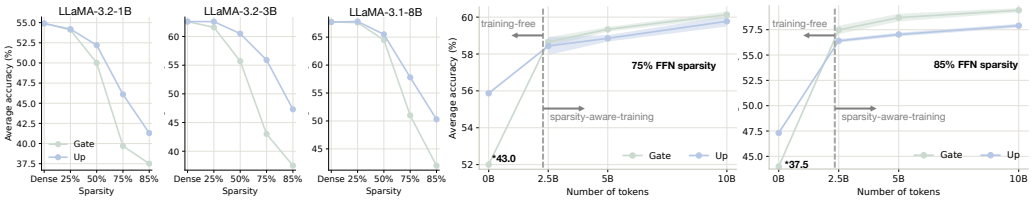
$$\mathbf{y} = \mathbf{W}^{\text{down}}[:, \mathcal{I}_g]\,(\mathbf{g}[\mathcal{I}_g] \odot \mathbf{W}^{\text{up}}[\mathcal{I}_g, :]\mathbf{x})\,. \tag{3}$$

The gate projection is evaluated densely to obtain $\mathcal{I}_g$, while only the selected gate values and corresponding up and down weights participate in the remaining computation. Up-based routing is defined analogously by exchanging the roles of the gate and up signals. We define the FFN channel sparsity as $s = 1 - k/n$.

4 SGA-SPARSE

4.1 GATE VS. UP ROUTING WITHOUT TRAINING

Prior work finds that the gate activation is less robust to sparsification than the up activation in frozen SwiGLU models (Liu et al., 2025; Szatkowski et al., 2026). Because $\mathbf{h}_i = \mathbf{g}_i\mathbf{u}_i$, ranking channels using $|\mathbf{g}|$ alone can discard channels whose gate activations are small but whose up activations have large magnitudes. Consequently, gate-based routing can introduce a larger FFN output error than up-based routing.



(a) (b)

Figure 2: (a) Training-free average zero-shot accuracy using gate- and up-based routing across FFN sparsity levels. (b) Average zero-shot accuracy during sparsity-aware training with gate- and up-based routing at 75% and 85% sparsity on LLaMA-3.2-3B, over 4 different seeds. For illustrational purpose, we clip the 0B gate points.

We verify this behavior across model sizes and sparsity levels. Figure 2(a) shows that the up signal begins to substantially outperform the gate signal at 50% sparsity, and the gap increases at higher sparsity. On LLaMA-3.2-3B at 85% sparsity, up-based routing improves average accuracy over gate-based routing by 9.8 pp. These results suggest that the native gate is a poor router when the pretrained model is kept frozen. We next examine whether this ordering remains after the model adapts to sparse execution.

4.2 ACTIVATION-SPARSITY-AWARE TRAINING

Rather than requiring the routing signal to reproduce the dense FFN activation pattern, we train the full model with routed FFNs enabled. Starting from pretrained parameters θ_0, we select a target sparsity s and minimize the causal language-modeling objective:

$$\mathcal{L}(\theta; s) = \mathbb{E}_{z_{1:T}\sim\mathcal{D}}\left[-\frac{1}{T}\sum_{t=1}^{T}\log p_{\theta,s}(z_t \mid z_{<t})\right], \tag{4}$$

where $z_{1:T}$ is a sequence of T tokens sampled from the training distribution $\mathcal{D}$, and $p_{\theta,s}$ denotes the next-token distribution under sparsity s.

During training, we compute the gate and up projections densely and construct the gate-based index set $\mathcal{I}_g$ using Equation 2. We then form the binary mask $\mathbf{m}$ from $\mathcal{I}_g$ and apply it to the post-gating activation:

$$\mathbf{y} = \mathbf{W}^{\text{down}}(\mathbf{h} \odot \mathbf{m}), \text{where } m_i = \begin{cases} 1, & i \in \mathcal{I}_g \\ 0, & i \notin \mathcal{I}_g \end{cases}. \tag{5}$$

Equation 5 produces the same output as Equation 3, while allowing training to use dense matrix multiplications without explicitly gathering rows and columns.

For backpropagation, let $\overline{\mathbf{y}} = \partial\mathcal{L}/\partial\mathbf{y}$ and $\mathbf{q} = (\mathbf{W}^{\text{down}})^\top\overline{\mathbf{y}}$. For a mask fixed during the forward pass, the activation gradients are $\partial\mathcal{L}/\partial\mathbf{g} = \mathbf{u} \odot \mathbf{m} \odot \mathbf{q}$ and $\partial\mathcal{L}/\partial\mathbf{u} = \mathbf{g} \odot \mathbf{m} \odot \mathbf{q}$. The down-projection gradient is $\partial\mathcal{L}/\partial\mathbf{W}^{\text{down}} = \overline{\mathbf{y}}(\mathbf{h} \odot \mathbf{m})^\top$. Thus, selected channels receive the standard gradient of the masked network, whereas dropped channels receive zero gradient for that token. A dropped channel can nevertheless be selected and updated by other tokens. We stop gradients through the hard top-k operation and do not use a pseudo-gradient (Hou et al., 2025) or auxiliary routing loss. We discuss alternative pseudo-gradients in Appendix B.1.

4.3 Gate vs. Up Routing after Training

We next compare gate- and up-based routing under matched sparsity-aware training. Figure 2(b) shows that training quickly closes the large initial gap between the two signals. On LLaMA-3.2-3B, the reversal is clearest at 85% sparsity, where gate-based routing overtakes up-based routing after approximately 2.5B training tokens and the gap widens to 1.5 pp in average accuracy. At 75% sparsity the two signals become comparable, with overlapping seed ranges. The reversal therefore emerges in the high-sparsity regime where the frozen-model gap was largest. Figure 2(b) reports the mean of 4 seeds, with shading showing the min–max range. These results show that routing quality in a frozen model does not necessarily predict routing quality after sparsity-aware training. We therefore use the native gate as the router for all subsequent **SGA-Sparse** experiments.

5 Activation Sparsity with Quantization

5.1 PTQ after Sparsity-Aware Training

Prior training-free studies report that activation sparsity and post-training weight quantization can be combined with little additional degradation (Zhang et al., 2025; Liu et al., 2025; Yin et al., 2026; Xu et al., 2025). However, this observation does not necessarily transfer after the model parameters have adapted to sparse execution. As shown in Section 6, applying W_4 GPTQ (Frantar et al., 2022) directly to our sparsity-trained checkpoints reduces average accuracy by 0.7–1.3 pp. Applying GPTQ after an additional 5B tokens of BF16 sparsity-aware training does not consistently reduce this loss.

Quantization can affect both the retained weight values and the top-k routing decisions. We do not attribute the observed degradation to either effect alone. Instead, we study how quantization should be incorporated into sparsity-aware training.

5.2 Continued-QAT and Joint-QAT

We consider two weight-only quantization targets. The first is uniform 4-bit quantization (W_4). The second is a more aggressive mixed-precision setting ($\text{W}_{2\&4}$), in which the up and down projections remain at 4 bits while the other quantized linear layers, including the gate projection, use 2 bits.

We study two complementary QAT recipes. *Continued-QAT* starts from a BF16 **SGA-Sparse** checkpoint. We enable fake weight quantization, retain the same gate-based top-k routing, and continue training for a smaller token budget, such as 5B tokens. This recipe allocates most of the total training budget to BF16 sparsity-aware training and uses a short QAT stage to adapt the router and remaining weights to low precision. We find that continued-QAT is effective for the W_4 target.

Joint-QAT starts from the pretrained dense checkpoint and enables both gate-based activation sparsity and fake weight quantization from the first training step. The model therefore learns the routing pattern and quantized weights together. Under the same total training-token budget, joint-QAT performs better for the more aggressive $\text{W}_{2\&4}$ target. The preferred recipe therefore depends on the quantization target and the available QAT budget.

5.3 Equivalent Bit-Width and Weight Traffic

Activation sparsity and quantization reduce weight traffic in different ways. Quantization reduces the number of bits used by every fetched weight, whereas activation sparsity reduces the fraction of up

and down weights fetched for each token. We convert these two effects into a common equivalent bit width.

Let p_q denote the fraction of model parameters that are quantized but fetched densely, p_s the fraction that are both quantized and sparsely fetched, and p_o the fraction that remains dense and BF16, such that $p_q + p_s + p_o = 1$. Let b_q and b_s denote the precision of the first two groups. We define b_{eq} as the uniform precision of the quantized parameter portion in a dense reference model with the same total idealized weight traffic. Because the p_o parameters remain BF16 in both models, equating their traffic gives

$$16p_o + (1 - p_o)b_{\text{eq}}(s) = 16p_o + p_q b_q + p_s b_s(1 - s), \tag{6}$$

and therefore

$$b_{\text{eq}}(s) = \frac{p_q b_q + p_s b_s(1 - s)}{1 - p_o}. \tag{7}$$

The dense gate and other densely fetched quantized weights belong to p_q, whereas the sparsely fetched up and down weights belong to p_s. This idealized calculation excludes quantization metadata, routing indices, and top-k overhead. For LLaMA-3.2-3B, combining 75% FFN sparsity with W_4 and $W_{2\&4}$ quantization gives $b_{\text{eq}} = 2.5$ and $b_{\text{eq}} = 1.5$, respectively. We use **SGA-Sparse-2.5** and **SGA-Sparse-1.5** as shorthand for these two configurations. These labels use the 3B calculation; exact values may vary slightly with model architecture.

6 Experiments

Table 1: Comparison of 75% and 85% FFN sparsity on LLaMA-3.2-1B, 3.2-3B, and 3.1-8B models, with other training-free baselines.

Model	Method	Wiki2(↓)	ARC-C	ARC-E	BoolQ	HS	OBQA	PIQA	SIQA	WG	Avg Acc.(↑)
LLaMA-3.2-1B	Dense	9.7	36.4	60.4	63.9	63.6	36.8	74.5	42.9	60.3	54.9
	CATS(75%)	174.7	25.5	35.7	50.8	33.9	28.2	57.8	34.4	51.5	39.7
	TEAL(75%)	30.7	26.2	41.6	52.9	41.8	28.8	62.0	37.8	51.1	42.8
	R-Sparse(75%)	31.3	26.0	42.1	**62.1**	40.8	28.6	59.7	37.8	51.7	43.6
	SGA-S$_{30BT}$(75%)	**12.4**	**38.1**	**67.4**	61.3	**59.0**	**39.8**	**73.2**	**42.4**	**56.6**	**54.7**
	CATS(85%)	17209.7	25.9	28.3	52.8	28.3	28.8	52.6	33.5	49.6	37.5
	TEAL(85%)	487.1	24.9	30.5	50.7	28.9	27.4	54.1	35.4	50.6	37.8
	R-Sparse(85%)	328.3	24.2	31.3	59.3	29.3	23.6	53.7	34.4	49.9	38.2
	SGA-S$_{30BT}$(85%)	**12.6**	**37.4**	**67.5**	**60.1**	**58.4**	**38.8**	**72.7**	**40.7**	**57.5**	**54.1**
LLaMA-3.2-3B	Dense	7.8	45.7	71.6	72.9	73.6	42.8	77.5	47.1	69.5	62.6
	CATS(75%)	128.8	30.0	40.0	47.8	45.6	31.6	60.9	35.9	52.1	43.0
	TEAL(75%)	17.7	33.7	51.6	54.5	55.3	35.4	67.6	41.3	55.9	49.4
	R-Sparse(75%)	18.1	33.0	51.3	63.9	54.3	30.6	68.6	40.6	55.6	49.7
	SGA-S$_{30BT}$(75%)	**9.6**	**47.1**	**73.7**	**71.0**	**68.7**	**44.0**	**77.5**	**45.7**	**64.5**	**61.5**
	CATS(85%)	5122.2	27.3	32.2	43.2	30.7	28.2	55.7	33.3	49.3	37.5
	TEAL(85%)	167.1	22.8	34.8	52.0	33.1	26.6	56.1	35.9	50.5	39.0
	R-Sparse(85%)	93.9	23.4	35.5	60.6	34.7	26.8	57.7	36.1	52.7	40.9
	SGA-S$_{30BT}$(85%)	**9.7**	**46.3**	**73.2**	**70.9**	**68.1**	**43.2**	**77.0**	**44.9**	**65.8**	**61.2**
LLaMA-3.1-8B	Dense	6.2	53.3	81.06	82.1	78.9	44.4	81.1	47.0	73.5	67.7
	CATS(75%)	28.4	34.0	50.6	55.4	63.5	37.4	72.0	40.2	54.5	51.0
	TEAL(75%)	12.5	37.8	59.5	69.6	65.5	40.0	74.5	42.9	60.3	56.3
	R-Sparse(75%)	13.0	39.2	63.2	71.5	64.8	38.2	73.6	43.0	62.4	57.0
	SGA-S$_{30BT}$(75%)	**7.9**	**52.6**	**78.2**	**77.6**	**75.9**	**46.4**	**79.4**	**47.2**	**70.6**	**66.0**
	CATS(85%)	956.8	29.0	36.9	48.8	41.6	31.0	62.1	35.3	51.5	42.0
	TEAL(85%)	46.6	27.2	41.1	52.1	42.1	29.6	59.6	36.7	55.3	43.0
	R-Sparse(85%)	46.7	28.3	43.0	63.9	44.0	27.4	61.5	38.4	51.7	44.8
	SGA-S$_{30BT}$(85%)	**8.1**	**49.9**	**77.4**	**76.6**	**75.0**	**45.0**	**79.3**	**46.9**	**69.4**	**65.0**

6.1 Experimental Setup

We evaluate **SGA-Sparse** on LLaMA-3.2-1B, LLaMA-3.2-3B, and LLaMA-3.1-8B (Grattafiori et al., 2024), all of which use SwiGLU FFNs. We report perplexity on WikiText-2 (Merity et al., 2016) with a sequence length of 2048 and average accuracy over eight zero-shot commonsense reasoning tasks: ARC-Easy, ARC-Challenge (Clark et al., 2018), BoolQ (Clark et al., 2019), HellaSwag (Zellers et al., 2019), OpenBookQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), Social IQa (Sap et al., 2019), and WinoGrande (Sakaguchi et al., 2021). We additionally evaluate generalization to Qwen-3.5-4B (Qwen Team, 2026) and report five-shot MMLU (Hendrycks et al., 2020) and GSM8K (Cobbe et al., 2021) results in the ablation study.

We initialize each model from its pretrained checkpoint and apply gate-based routing to every FFN block. We train on FineWeb-100BT (Penedo et al., 2024) for up to 60,000 steps, corresponding to approximately 30B tokens. We use AdamW (Loshchilov & Hutter, 2017) with a weight decay of 0.05 and train on 32 GPUs with a batch size of 8 per GPU. Additional hyperparameters are provided in Table 7. For W_4 and W_3 QAT, we use learned quantization scales (Esser et al., 2019) with a group size of 128. For W_2 and ternary quantization, we use the SEQ formulation from ParetoQ (Liu et al., 2026), also with a group size of 128. Embedding and output-layer weights remain in BF16. Additional QAT settings are provided in Table 8.

Table 2: Comparison with activation-sparsity-aware training methods. All methods use a 20B-token training budget.

Model Size	Method	Wiki2(↓)	Avg Acc.(↑)
1B	Dense	9.7	54.9
	Q-Sparse$_{20BT}$(75%)	13.3	52.7
	SGA-S$_{20BT}$(75%)	**12.5**	**54.7**
	QSparse$_{20BT}$(85%)	14.3	51.6
	ReLUfication$_{20BT}$(85%)	18.8	51.3
	TwELL$_{20BT}$(85%)	15.4	50.3
	SGA-S$_{20BT}$(85%)	**12.8**	**53.9**
3B	Dense	7.8	62.6
	Q-Sparse$_{20BT}$(75%)	10.0	58.2
	SGA-S$_{20BT}$(75%)	**9.8**	**60.9**
	Q-Sparse$_{20BT}$(85%)	10.9	57.6
	ReLUfication$_{20BT}$(85%)	10.2	59.8
	SGA-S$_{20BT}$(85%)	**9.9**	**60.8**

We compare against controlled top-k implementations of the training-free methods CATS (Lee et al., 2024), TEAL (Liu et al., 2025), and R-Sparse (Zhang et al., 2025), and the training-based methods Q-Sparse (Wang et al., 2024), ReLUfication (Mirzadeh et al., 2024), and TwELL (Cetin et al., 2026). Except for TwELL, we implement and evaluate all baselines in the same framework. We use per-token top-k selection to enforce an exact and uniform sparsity level. These controlled implementations may differ from the adaptive-threshold configurations used in the original training-free methods, they provide a like-for-like comparison at fixed sparsity.

6.2 Main Results

We first compare **SGA-Sparse** with the training-free baselines in Table 1. At 75% and 85% FFN sparsity, every training-free baseline loses more than 10 pp of average accuracy relative to the corresponding dense model. R-Sparse performs best among these baselines, aided by its additional low-rank residual paths. Nevertheless, at 85% sparsity, it remains 21.7 and 22.9 points below the dense 3B and 8B models, respectively. Sparsity-aware training substantially reduces these gaps. At 75% sparsity, **SGA-Sparse** remains within 0.2, 1.1, and 1.7 points of the dense 1B, 3B, and 8B models, respectively. At 85% sparsity, the corresponding gaps are 0.8, 1.4, and 2.7 points. The larger gaps at 8B suggest that larger models may require more training than the fixed 30B-token budget used here.

We next compare against training-based methods under a matched 20B-token budget in Table 2. **SGA-Sparse** achieves the best average accuracy at both sparsity levels on the 1B and 3B models. Compared with Q-Sparse, it improves the 3B accuracy by 2.7 points at 75% sparsity and 3.2 points at 85% sparsity. We attribute part of this difference to the sparsity overlap introduced by applying input sparsity to consecutive FFN projections. At 85% sparsity, **SGA-Sparse** outperforms ReLUfication and TwELL by at least 2.6 pp on the 1B model. On the 3B model, it outperforms ReLUfication by 1.0 pp while preserving the original SwiGLU activation.

Table 3: W_4 quantization of 75%-sparse **SGA-Sparse** models. We compare GPTQ after 30B or 35B tokens of sparse training with continued-QAT using 30B tokens of sparse training followed by 5B tokens of QAT.

Model	Method	Wiki2(↓)	Avg Acc.(↑)
3.2-1B	**SGA-S$_{30BT}$(75%)** + BF16	12.4	54.7
	SGA-S$_{30BT}$(75%)+ GPTQ(W_4)	13.1	53.8
	SGA-S$_{35BT}$(75%)+ GPTQ(W_4)	13.0	54.1
	SGA-S$_{30BT}$(75%) + QAT$_{5BT}$(W_4)	**12.6**	**54.9**
3.2-3B	**SGA-S$_{30BT}$(75%)** + BF16	9.6	61.5
	SGA-S$_{30BT}$(75%)+ GPTQ(W_4)	10.8	60.2
	SGA-S$_{35BT}$(75%)+ GPTQ(W_4)	10.0	59.9
	SGA-S$_{30BT}$(75%) + QAT$_{5BT}$(W_4)	**9.8**	**61.6**
3.1-8B	**SGA-S$_{30BT}$(75%)** + BF16	7.9	66.0
	SGA-S$_{30BT}$(75%)+ GPTQ(W_4)	8.2	65.3
	SGA-S$_{35BT}$(75%)+ GPTQ(W_4)	8.2	65.2
	SGA-S$_{30BT}$(75%) + QAT$_{5BT}$(W_4)	**8.0**	**65.5**

Table 3 compares W_4 GPTQ (Frantar et al., 2022) with continued-QAT for 75%-sparse **SGA-Sparse** checkpoints. Applying GPTQ with group size of 128 directly after 30B tokens of sparsity-aware training reduces average accuracy by 0.7–1.3 points. Performing another 5B tokens of BF16 sparsity-aware training before GPTQ does not consistently reduce this loss. In contrast, continued-QAT recovers the BF16 sparse accuracy on the 1B and 3B models and remains within 0.5 pp on the 8B model.

We next compare the two QAT recipes for the more aggressive 1.5-bit-equivalent target. Continued-QAT uses 25B tokens of BF16 sparsity-aware training followed by 5B tokens of QAT, whereas joint-QAT enables sparsity and quantization throughout all 30B training tokens. As shown in Table 4, joint-QAT improves average accuracy by 1.3, 0.7, and 1.1 points on the 1B, 3B, and 8B models, respectively. It also reduces perplexity at every model size. These results show that the aggressive mixed-$W_{2\&4}$ target benefits from learning the routing pattern and quantized weights jointly.

Table 4: Comparison of continued-QAT and joint-QAT for 75%-sparse mixed-$W_{2\&4}$ models under the same 30B-token budget.

Model	Method	Wiki2(↓)	Avg Acc(↑)
3.2-1B	Continued-QAT (25B + 5B)	14.5	51.8
	Joint-QAT (30B)	**13.4**	**53.1**
3.2-3B	Continued-QAT (25B + 5B)	10.9	59.3
	Joint-QAT (30B)	**10.1**	**60.0**
3.1-8B	Continued-QAT (25B + 5B)	9.0	63.5
	Joint-QAT (30B)	**8.6**	**64.6**

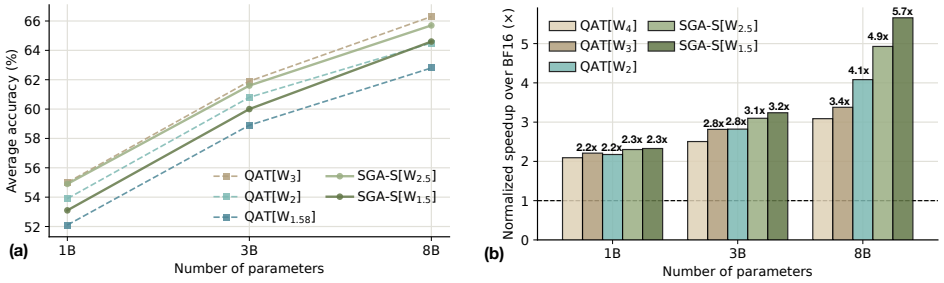


Figure 3: (a) Average zero-shot accuracy across model sizes between dense QAT models and quantized SGA-S models. (b) Speedup over dense BF16 on an NVIDIA A100 80GB GPU. We benchmark the gate, up, and down projection's GEMVs separately and report their summed latency.

Finally, we compare quantized **SGA-Sparse** models with dense low-bit QAT models using equivalent weight traffic. As shown in Figure 3(a), the **SGA-Sparse-2.5** models achieve accuracy comparable to dense W_3 models across model sizes ranging from 1B to 8B parameters. The 1.5-bit-equivalent models similarly approach the accuracy of dense W_2 models and consistently outperform dense ternary models by 1.3 pp on average. These results introduce new Pareto-efficient operating points among the evaluated models. Figure 3(b) further evaluates the corresponding GPU efficiency. We benchmark the gate, up, and down GEMVs separately and report their summed latency. The measurement excludes top-k selection and elementwise operations. Both W_4+75% sparsity and $W_{2\&4}$+75% sparsity achieve greater projection-level speedup than the dense W_2 implementation.

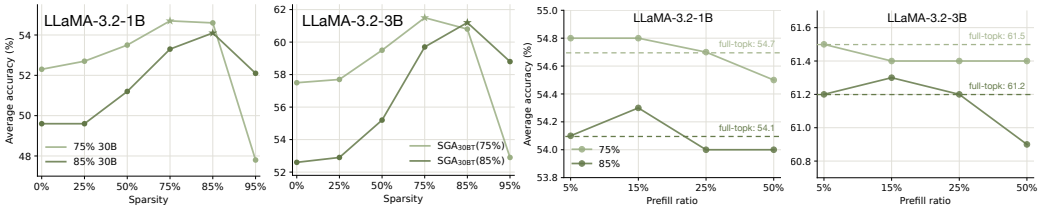


Figure 4: (a) Ablation study on the sparsity-aware-trained model's performance across different sparsity levels after the training. (b) The study of using the top-k free threshold-based method on the prefill tokens. On non-generated tasks, we sweep the prefill ratio (Liu et al., 2025; Zhang et al., 2025) from 5% to 50%.

6.3 Ablation Studies

Figure 4(a) evaluates each checkpoint at sparsity levels different from its training target. Each model performs best near the sparsity level used during training, and accuracy decreases as the evaluation sparsity moves away from this target. Nevertheless, evaluating the 85%-trained checkpoints at 95% sparsity reduces average accuracy by only ap-

proximately two points on the 1B and 3B models. This suggests that further training from an 85%-sparse checkpoint may provide an efficient path toward more extreme sparsity. Furthermore, according to Figure 4(a), applying dense execution to prefill tokens also creates a mismatch with the sparse computation used during training. We therefore apply an offline calibrated threshold during prefill. As shown in Figure 4(b), threshold-based prefill preserves accuracy across the evaluated settings without performing a top-k operation for every prefill token. The resulting number of active channels can vary across tokens, but our primary deployment acceleration target is single-token decoding, where we retain fixed-budget top-k routing.

Table 5: Comparison of gate- and up-based routing on Qwen-3.5-4B.

Model	Method	Wiki2(↓)	Avg Acc.(↑)
Qwen-3.5-4B	Dense	8.0	66.6
	SGA-S$_{10BT}$(75%)-up	9.1	64.7
	SGA-S$_{10BT}$(75%)-gate	**8.8**	**65.6**
	SGA-S$_{10BT}$(85%)-up	9.5	63.4
	SGA-S$_{10BT}$(85%)-gate	**8.9**	**64.7**

Table 5 extends our evaluation to Qwen-3.5-4B. With only 10B training tokens, gate-based **SGA-Sparse** remains within 1.0 and 1.9 pp of the dense model at 75% and 85% sparsity, respectively. It also outperforms up-based routing by 0.9 and 1.3 pp at the two sparsity levels. These results show that both **SGA-Sparse** and the training-induced preference for gate-based routing extend beyond the LLaMA family to another pretrained SwiGLU model family. Finally, Table 6 compares **SGA-Sparse** and Q-Sparse on LLaMA-3.2-3B models trained with 30B tokens on five-shot MMLU and GSM8K, where **SGA-Sparse** provides the largest gains at 85% sparsity.

Table 6: Performance comparison with Q-Sparse on few-shot reasoning tasks.

Method	5-shot MMLU	5-shot GSM8K	Avg. 8-task Acc.
Q-Sparse$_{30BT}$(75%)	39.9	9.0	59.4
SGA-S$_{30BT}$(75%)	40.5	9.2	61.5
Q-Sparse$_{30BT}$(85%)	33.2	5.8	58.5
SGA-S$_{30BT}$(85%)	38.1	8.2	**61.2**

7 Limitations

Our study focuses on signal-based intermediate-channel sparsity in SwiGLU FFNs. Our main experiments cover LLaMA 3 models from 1B to 8B parameters, with one Qwen model as cross-family validation, and use a single continued pretraining distribution. Whether our findings generalize to ReLU-based FFNs, attention sparsity, and mixture-of-experts routing remains an important direction for future work.

8 Conclusion

We present **SGA-Sparse**, a method for converting pretrained SwiGLU LLMs into fixed-budget activation-sparse models without changing the pretrained activation function or introducing auxiliary router parameters. To the best of our knowledge, this is the first systematic study that trains the existing SwiGLU gate as the router through full-model continued pretraining. Our key finding is that the best routing signal changes with training: although the up activation is substantially better in frozen models, the native gate becomes the stronger signal after sparsity-aware training. With approximately 30B training tokens, **SGA-Sparse** achieves uniform per-token 75–85% FFN sparsity across LLaMA models from 1B to 8B parameters while remaining within 0.2–2.7 percentage points of their original dense checkpoints. This result also extends to Qwen-3.5-4B.

We further show that quantization must be considered together with sparsity-aware training and proposing two QAT recipes: Continued-QAT and joint-QAT. Finally, we translate sparsity-induced weight-traffic reductions into an equivalent bit width, enabling direct comparison with dense low-bit QAT models. Our 2.5-bit-equivalent models achieve accuracy comparable to dense W_3 models, while our 1.5-bit-equivalent models approach dense W_2 performance and outperform dense ternary models by 1.3 points on average. Together, these results establish native-gate training as a practical way to combine high activation sparsity with low-bit weights. More broadly, we hope **SGA-Sparse** helps establish activation-sparsity-aware training, alongside QAT, as a practical model-optimization paradigm for efficient on-device LLM deployment.

AI USE STATEMENT

In this work, we have not used generative AI tools for: Generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design research and provide feedback on research methodology or experiments, implement methods, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis, interpret results.

Additionally, we used generative AI tools for sourcing/searching for information, edit a research paper to improve readability.

We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We try to provide as much as the training details and configurations for the reproducibility. Our training code will be made publicly available upon acceptance.

REFERENCES

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Edoardo Cetin, Stefano Peluchetti, Emilio Castillo, Akira Naruse, Mana Murakami, and Llion Jones. Sparser, faster, lighter transformer language models. *arXiv preprint arXiv:2603.23198*, 2026.

Yanbei Chen, Hanxian Huang, Ernie Chang, Jacob Szwejbka, Digant Desai, Zechun Liu, Vikas Chandra, and Raghuraman Krishnamoorthi. Mobilemoe: Scaling on-device mixture of experts. *arXiv preprint arXiv:2605.27358*, 2026.

Christopher Clark et al. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

Peter Clark et al. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Karl Cobbe et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Steven K Esser et al. Learned step size quantization. *arXiv preprint arXiv:1902.08153*, 2019.

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

Aaron Grattafiori et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Dan Hendrycks et al. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Bairu Hou, Qibin Chen, Jianyu Wang, Guoli Yin, Chong Wang, Nan Du, Ruoming Pang, Shiyu Chang, and Tao Lei. Instruction-following pruning for large language models. *arXiv preprint arXiv:2501.02086*, 2025.

Donghyun Lee, Jaeyong Lee, Genghan Zhang, Mo Tiwari, and Azalia Mirhoseini. Cats: Context-aware thresholding for sparsity in large language models. In *First Conference on Language Modeling*, 2024.

Zonglin Li et al. The lazy neuron phenomenon: On emergence of activation sparsity in transformers. *arXiv preprint arXiv:2210.06313*, 2022.

James Liu et al. Training-free activation sparsity in large language models. *The Thirteenth International Conference on Learning Representations*, 2025.

Zechun Liu, Changsheng Zhao, Forrest Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, et al. Mobilellm: Optimizing sub-billion parameter language models for on-device use cases. *arXiv preprint arXiv:2402.14905*, 2024.

Zechun Liu, Changsheng Zhao, Hanxian Huang, Sijia Chen, Jing Zhang, Jiawei Zhao, Scott Roy, Lisa Jin, Yunyang Xiong, Yangyang Shi, et al. Paretoq: Improving scaling laws in extremely low-bit llm quantization. *Advances in Neural Information Processing Systems*, 38:91311–91336, 2026.

Zichang Liu et al. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*, pp. 22137–22176. PMLR, 2023.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.

Iman Mirzadeh et al. Relu strikes back: Exploiting activation sparsity in large language models. In *International Conference on Learning Representations*, volume 2024, pp. 25378–25400, 2024.

Guilherme Penedo et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Maarten Sap et al. Social iqa: Commonsense reasoning about social interactions. In *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. 4463–4473, 2019.

Susav Shrestha, Bradley Settlemyer, Nikoli Dryden, and Narasimha Reddy. Polar sparsity: High throughput batched llm inferencing with scalable contextual sparsity. *Advances in Neural Information Processing Systems*, 38:84842–84870, 2026.

Chenyang Song, Xu Han, Zhengyan Zhang, Shengding Hu, Xiyu Shi, Kuai Li, Chen Chen, Zhiyuan Liu, Guangli Li, Tao Yang, et al. Prosparse: Introducing and enhancing intrinsic activation sparsity within large language models. In *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 2626–2644, 2025.

Filip Szatkowski et al. Universal properties of activation sparsity in modern large language models. In *International Conference on Learning Representations*, volume 2026, pp. 146247–146269, 2026.

Hongyu Wang, Shuming Ma, Ruiping Wang, and Furu Wei. Q-sparse: All large language models can be fully sparsely-activated. *arXiv preprint arXiv:2407.10969*, 2024.

Haotian Xu, Jiannan Yang, Tian Gao, Tsui-Wei Weng, and Tengfei Ma. Resting neurons, active insights: Robustifying activation sparsity in llms via spontaneity. *arXiv preprint arXiv:2512.12744*, 2025.

Ruokai Yin, Yuhang Li, Donghyun Lee, and Priyadarshini Panda. Duogpt: Training-free dual sparsity through activation-aware pruning in llms. *Advances in Neural Information Processing Systems*, 38: 146883–146912, 2026.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

Zhenyu Zhang et al. R-sparse: Rank-aware activation sparsity for efficient llm inference. In *The Thirteenth International Conference on Learning Representations*, 2025.

Table 7: Hyperparameters for the main sparsity-aware continued-pretraining runs. All models are trained on 32 GPUs with a sequence length of 2,048 and a linear learning-rate schedule.

Model	Method	Steps	Tokens/Step	Peak/Min. LR	WD	Warmup	Batch/GPU	Seed
LLaMA-3.2-1B	**SGA-S**$_{30BT}$(75%)	60000	524288	1e-4/0	5e-2	5%	8	428
	SGA-S$_{30BT}$(85%)	60000	524288	1e-4/0	5e-2	5%	8	428
LLaMA-3.2-3B	**SGA-S**$_{30BT}$(75%)	60000	524288	1e-4/0	5e-2	5%	8	428
	SGA-S$_{30BT}$(85%)	60000	524288	8e-5/0	5e-2	5%	8	428
LLaMA-3.1-8B	**SGA-S**$_{30BT}$(75%)	60000	524288	4e-5/0	5e-2	5%	8	428
	SGA-S$_{30BT}$(85%)	60000	524288	4e-5/0	5e-2	5%	8	428

Table 8: Hyperparameters for the dense ParetoQ QAT baselines. All models are trained on 32 GPUs with a sequence length of 2,048 and a linear learning-rate schedule. Following the original implementation, learnable quantization scales use zero weight decay and share the base learning rate of the weights.

Model	# of Bits	Steps	Tokens/Step	Peak/Min. LR	Group Size	WD	Warmup	Batch/GPU	Seed
1B/3B/8B	W_3	20000	524288	2e-5/0	128	5e-2	5%	8	428
	W_2	40000	524288	4e-5/0	128	5e-2	5%	8	428
	$W_{1.58}$	40000	524288	4e-5/0	128	5e-2	5%	8	428

A Reproducibility Details

Tables 7 and 8 summarize the training configurations used in our main experiments and dense ParetoQ baselines, respectively. For the ParetoQ baselines, we use one BF16 quantization scale per group of 128 weights. We find that this group-wise scaling converges faster than the per-channel scaling used in the original work. We therefore train the W_2 and ternary models for 20B tokens, at which point they reach performance comparable to that reported in the original paper.

For continued-QAT, we use 1e-5 as the learning rate with all other settings remain the same. For joint-QAT, we use the same learning rate as the non-quantized version. For Qwen-3.5-4B, both gate and up-based candidates use a learning rate of 4e-5.

B Additional Method Details

B.1 Gradient Through Hard Top-k Routing

For gate-based routing, let $\mathbf{r} = |\mathbf{g}|$ and $\mathbf{m} = M_k(\mathbf{r})$, where $[M_k(\mathbf{r})]_i = 1$ if $i \in \text{top-k}(\mathbf{r}, k)$ and is zero otherwise. Let $\mathbf{q} = (\mathbf{W}^{\text{down}})^\top \overline{\mathbf{y}}$. Since $\partial \mathcal{L}/\partial \mathbf{m} = \mathbf{h} \odot \mathbf{q}$, the gradient with respect to the gate activation can be decomposed as

$$\frac{\partial \mathcal{L}}{\partial \mathbf{g}} = \underbrace{\mathbf{u} \odot \mathbf{m} \odot \mathbf{q}}_{\text{direct activation path}} + \underbrace{\operatorname{sgn}(\mathbf{g}) \odot J_{M_k}(\mathbf{r})^\top (\mathbf{h} \odot \mathbf{q})}_{\text{routing path}}, \tag{8}$$

where $J_{M_k} = \partial \mathbf{m}/\partial \mathbf{r}$. The first term is the direct activation gradient reported in Section 4.2; the second term is the routing gradient. For hard top-k, M_k is piecewise constant, so its Jacobian is zero almost everywhere and undefined when channels tie at the selection boundary. We therefore treat the mask as constant during backpropagation and do not supply a surrogate gradient. The gate projection still receives gradients through selected gate activations, so its routing decisions can change between optimization steps.

A differentiable top-k relaxation (Hou et al., 2025) can replace the hard mask during training. Alternatively, a straight-through estimator can use the hard mask in the forward pass and a surrogate Jacobian in the backward pass. We leave these alternatives to future work.

Table 9: WikiText-2 perplexity and zero-shot accuracy for training-free gate- and up-based routing across model sizes and FFN sparsity levels.

Model	Method	Wiki2(↓)	ARC-C	ARC-E	BoolQ	HS	OBQA	PIQA	SIQA	WG	Avg.(↑)
LLaMA-3.2-1B	Dense	9.7	36.4	60.4	63.9	63.6	36.8	74.5	42.9	60.3	54.9
	up-25%	9.9	35.1	60.0	62.2	63.1	36.6	74.3	42.6	59.8	54.2
	up-50%	11.5	33.2	57.6	58.0	60.2	37.2	72.4	41.6	57.7	52.2
	up-75%	22.8	29.3	48.4	54.3	47.3	31.4	65.6	39.3	53.4	46.1
	up-85%	50.8	27.1	39.9	50.0	37.2	29.2	59.5	36.0	51.5	41.3
	gate-25%	10.1	35.8	59.7	63.0	62.4	36.8	73.6	42.5	58.6	54.1
	gate-50%	15.0	33.4	54.0	59.3	55.5	32.2	69.7	40.8	55.3	50.0
	gate-75%	174.7	25.5	35.7	50.8	33.9	28.2	57.8	34.4	51.5	39.7
	gate-85%	17209.7	25.9	28.3	52.8	28.3	28.8	52.6	33.5	49.6	37.5
LLaMA-3.2-3B	Dense	7.8	45.7	71.6	72.9	73.6	42.8	77.5	47.1	69.5	62.6
	up-25%	7.8	46.6	71.0	72.9	73.7	42.4	77.8	47.1	69.5	62.6
	up-50%	8.5	42.3	69.2	69.4	72.3	40.4	77.9	46.2	65.9	60.5
	up-75%	13.3	37.9	60.5	66.6	64.9	40.0	72.1	43.0	61.9	55.9
	up-85%	25.7	32.3	49.5	54.6	52.5	30.0	66.1	40.5	52.8	47.3
	gate-25%	8.0	44.8	70.6	69.5	73.8	41.8	77.5	47.0	67.5	61.6
	gate-50%	11.2	42.2	64.7	48.8	70.3	40.6	75.8	43.4	59.7	55.7
	gate-75%	128.8	30.0	40.0	47.8	45.6	31.6	60.9	35.9	52.1	43.0
	gate-85%	5122.2	27.3	32.2	43.2	30.7	28.2	55.7	33.3	49.3	37.5
LLaMA-3.1-8B	Dense	6.2	53.3	81.06	82.1	78.9	44.4	81.1	47.0	73.5	67.7
	up-25%	6.3	53.8	80.2	81.9	78.8	45.2	81.1	47.3	73.9	67.8
	up-50%	7.2	51.5	76.8	77.2	77.8	43.4	79.5	45.9	72.0	65.5
	up-75%	13.4	39.6	62.6	62.9	71.2	38.6	76.3	45.6	65.4	57.8
	up-85%	27.9	34.6	53.3	48.9	60.0	35.6	70.1	41.9	58.4	50.3
	gate-25%	6.3	53.5	80.3	81.8	78.9	44.4	81.2	47.3	73.4	67.6
	gate-50%	7.8	48.8	74.3	77.9	76.8	43.0	79.8	45.0	70.1	64.5
	gate-75%	28.4	34.0	50.6	55.4	63.5	37.4	72.0	40.2	54.5	51.0
	gate-85%	956.8	29.0	36.9	48.8	41.6	31.0	62.1	35.3	51.5	42.0

Table 10: W_4 GPTQ and continued-QAT results for checkpoints trained with 75% or 85% FFN sparsity. Dense denotes the original pretrained model.

Model	Method	Wiki2(↓)	ARC-C	ARC-E	BoolQ	HS	OBQA	PIQA	SIQA	WG	Avg.(↑)
LLaMA-3.2-1B	Dense	9.7	36.4	60.4	63.9	63.6	36.8	74.5	42.9	60.3	54.9
	SGA-S$_{35BT}$(75%)+ GPTQ(W_4)	13.0	38.3	66.4	59.2	58.2	39.4	73.2	41.4	57.1	54.1
	SGA-S$_{30BT}$(75%)+ QAT$_{5BT}$(W_4)	12.6	37.6	66.3	61.9	58.5	41.4	73.2	41.9	58.3	54.9
	SGA-S$_{35BT}$(85%)+ GPTQ(W_4)	13.3	37.6	65.6	59.0	58.1	38.0	72.8	41.1	56.6	53.6
	SGA-S$_{30BT}$(85%)+ QAT$_{5BT}$(W_4)	12.8	37.0	66.5	59.4	58.4	39.8	72.3	40.5	58.6	54.1
LLaMA-3.2-3B	Dense	7.8	45.7	71.6	72.9	73.6	42.8	77.5	47.1	69.5	62.6
	SGA-S$_{35BT}$(75%)+ GPTQ(W_4)	10.0	44.0	71.7	69.7	68.2	42.0	75.9	43.6	63.9	59.9
	SGA-S$_{30BT}$(75%)+ QAT$_{5BT}$(W_4)	9.8	**48.7**	**74.0**	**72.1**	**68.8**	**43.2**	**77.5**	44.0	**64.4**	**61.6**
	SGA-S$_{35BT}$(85%)+ GPTQ(W_4)	10.2	43.2	70.4	69.3	67.8	42.2	76.1	44.2	62.4	59.5
	SGA-S$_{30BT}$(85%)+ QAT$_{5BT}$(W_4)	9.9	45.7	72.3	71.0	68.1	44.0	76.4	44.5	64.1	60.9
LLaMA-3.1-8B	Dense	6.2	53.3	81.06	82.1	78.9	44.4	81.1	47.0	73.5	67.7
	SGA-S$_{35BT}$(75%)+ GPTQ(W_4)	8.2	51.4	77.1	77.6	74.9	45.4	80.0	46.4	69.1	65.2
	SGA-S$_{30BT}$(75%)+ QAT$_{5BT}$(W_4)	8.0	52.3	77.4	76.1	75.8	45.8	79.3	46.3	70.8	65.5
	SGA-S$_{35BT}$(85%)+ GPTQ(W_4)	8.7	50.4	77.2	73.5	73.8	44.0	77.9	45.9	66.5	63.6
	SGA-S$_{30BT}$(85%)+ QAT$_{5BT}$(W_4)	8.3	50.4	77.1	75.1	74.8	44.4	80.0	46.8	67.9	64.6

C Additional Results

Unless otherwise stated, Dense denotes the original pretrained checkpoint, and the training-free rows use the controlled fixed-top-k protocol described in Section 6.

C.1 Training-Free Gate and Up Routing

Table 9 provides the per-task results summarized in Figure 2(a).

C.2 Quantization Results

Table 10 reports the per-task W_4 GPTQ and continued-QAT results. Table 11 provides the dense ParetoQ baselines used in the equivalent-traffic comparison.

Table 11: WikiText-2 perplexity and zero-shot accuracy of the dense ParetoQ QAT baselines.

Model	Method	Wiki2(↓)	ARC-C	ARC-E	BoolQ	HS	OBQA	PIQA	SIQA	WG	Avg.(↑)
LLaMA-3.2-1B	Dense	9.7	36.4	60.4	63.9	63.6	36.8	74.5	42.9	60.3	54.9
	ParetoQ$_{10BT}$(W_3)	11.2	37.0	65.5	63.7	61.0	37.8	73.5	42.4	59.7	55.1
	ParetoQ$_{20BT}$(W_2)	11.5	35.8	64.4	61.3	57.5	39.0	73.1	41.4	58.7	53.9
	ParetoQ$_{20BT}$($W_{1.58}$)	12.0	34.0	62.7	60.6	53.0	36.6	71.1	40.8	58.1	52.1
LLaMA-3.2-3B	Dense	7.8	45.7	71.6	72.9	73.6	42.8	77.5	47.1	69.5	62.6
	ParetoQ$_{10BT}$(W_3)	8.2	46.2	73.8	71.3	71.4	41.4	77.7	45.6	67.4	61.9
	ParetoQ$_{20BT}$(W_2)	8.6	43.8	72.1	70.9	69.1	43.4	77.1	44.5	65.3	60.8
	ParetoQ$_{20BT}$($W_{1.58}$)	9.3	43.2	70.3	68.0	66.2	41.0	75.3	44.9	62.1	58.9
LLaMA-3.1-8B	Dense	6.2	53.3	81.06	82.1	78.9	44.4	81.1	47.0	73.5	67.7
	ParetoQ$_{10BT}$(W_3)	7.2	51.5	78.8	78.7	77.2	46.8	80.1	47.4	71.3	66.5
	ParetoQ$_{20BT}$(W_2)	7.2	50.3	76.0	74.2	74.6	43.4	79.3	47.1	71.4	64.5
	ParetoQ$_{20BT}$($W_{1.58}$)	8.2	50.4	75.0	65.9	72.7	43.8	77.7	47.0	69.5	62.8

Table 12: Per-task results for activation-sparsity-aware training methods under a 10B-token budget.

Model Size	Method	Wiki2(↓)	ARC-C	ARC-E	BoolQ	HS	OBQA	PIQA	SIQA	WG	Avg.(↑)
1B	Dense	9.7	36.4	60.4	63.9	63.6	36.8	74.5	42.9	60.3	54.9
	Q-Sparse$_{10BT}$(75%)	13.7	33.2	61.6	58.4	54.0	36.0	70.7	41.6	56.4	51.5
	SGA-S$_{10BT}$(75%)	12.6	36.5	65.3	61.3	57.6	38.4	73.1	40.6	56.8	53.7
	Q-Sparse$_{10BT}$(85%)	15.1	32.2	59.5	59.1	50.6	34.4	70.0	41.2	55.5	50.3
	ReLUfication$_{10BT}$(85%)	18.1	32.6	57.3	60.2	56.0	36.0	70.7	39.6	53.4	50.7
	SGA-S$_{10BT}$(85%)	13.0	35.3	64.2	60.2	56.1	37.8	73.5	40.5	58.2	53.2
3B	Dense	7.8	45.7	71.6	72.9	73.6	42.8	77.5	47.1	69.5	62.6
	Q-Sparse$_{10BT}$(75%)	10.2	42.0	67.9	64.8	65.8	39.6	75.4	43.9	61.3	57.6
	SGA-S$_{10BT}$(75%)	9.7	44.2	71.9	70.0	68.0	41.6	76.8	44.6	63.4	60.1
	Q-Sparse$_{10BT}$(85%)	11.2	40.0	66.0	66.1	62.4	38.4	73.6	43.3	60.1	56.2
	ReLUfication$_{10BT}$(85%)	10.3	43.8	71.0	69.4	68.2	40.6	76.1	43.8	63.6	59.6
	SGA-S$_{10BT}$(85%)	10.1	45.9	70.7	70.2	67.2	41.2	76.1	45.4	64.8	60.2

C.3 Training-Budget Comparisons

Tables 12 and 13 provide the per-task comparisons at 10B and 20B training tokens, respectively.

C.4 Cross-Family Evaluation

Table 14 provides the complete Qwen-3.5-4B results used in the cross-family ablation.

Table 13: Per-task results for activation-sparsity-aware training methods under a 20B-token budget.

Model Size	Method	Wiki2(↓)	ARC-C	ARC-E	BoolQ	HS	OBQA	PIQA	SIQA	WG	Avg.(↑)
1B	Dense	9.7	36.4	60.4	63.9	63.6	36.8	74.5	42.9	60.3	54.9
	Q-Sparse$_{20BT}$(75%)	13.3	35.2	63.7	59.8	55.5	37.2	70.9	**42.2**	56.9	52.7
	SGA-S$_{20BT}$(75%)	**12.5**	**37.4**	**66.6**	**61.6**	**58.7**	**40.2**	**73.3**	41.0	**58.9**	**54.7**
	Q-Sparse$_{20BT}$(85%)	14.3	33.1	61.6	60.3	52.6	35.6	71.3	**41.0**	**57.5**	51.6
	ReLUfication$_{20BT}$(85%)	18.8	33.3	56.8	**61.2**	56.3	37.0	70.7	39.2	55.5	51.3
	TwELL$_{20BT}$(85%)	15.4	34.0	63.5	62.3	44.6	37.2	68.7	39.2	53.2	50.3
	SGA-S$_{20BT}$(85%)	**12.8**	**37.3**	**66.1**	60.2	**58.0**	**38.0**	**73.7**	40.9	57.2	**53.9**
3B	Dense	7.8	45.7	71.6	72.9	73.6	42.8	77.5	47.1	69.5	62.6
	Q-Sparse$_{20BT}$(75%)	10.0	41.9	68.3	67.1	67.1	40.2	75.1	43.4	62.4	58.2
	SGA-S$_{20BT}$(75%)	**9.8**	**47.1**	**73.3**	**69.9**	**68.6**	**42.2**	**76.2**	**45.3**	**64.4**	**60.9**
	Q-Sparse$_{20BT}$(85%)	10.9	41.7	68.8	66.1	64.1	41.4	74.8	43.6	60.1	57.6
	ReLUfication$_{20BT}$(85%)	10.2	44.2	70.7	69.5	**68.7**	40.6	**76.6**	44.3	63.5	59.8
	SGA-S$_{20BT}$(85%)	**9.9**	**46.0**	**72.5**	**70.3**	68.1	**44.2**	76.3	**44.6**	**64.1**	**60.8**

Table 14: Per-task results for gate- and up-based routing on Qwen-3.5-4B after 10B training tokens.

Model	Method	Wiki2(↓)	ARC-C	ARC-E	BoolQ	HS	OBQA	PIQA	SIQA	WG	Avg.(↑)
Qwen-3.5-4B	Dense	8.0	55.2	76.3	83.9	75.3	42.4	79.5	49.2	71.1	66.6
	SGA-S$_{10BT}$(75%)-up	9.1	55.4	79.3	76.5	71.4	43.6	77.4	45.8	68.0	64.7
	SGA-S$_{10BT}$(75%)-gate	**8.8**	**56.4**	**80.6**	**78.7**	**71.9**	**44.2**	**77.7**	**46.2**	**69.2**	**65.6**
	SGA-S$_{10BT}$(85%)-up	9.5	51.2	77.1	76.1	69.7	43.4	76.6	45.3	67.5	63.4
	SGA-S$_{10BT}$(85%)-gate	**8.9**	**55.0**	**80.0**	**77.4**	**71.4**	**43.4**	**77.5**	**45.3**	**67.6**	**64.7**