SlimQwen: Exploring the Pruning and Distillation in Large MoE Model Pre-training

Anonymous Authors[1]

Abstract

Structured pruning and knowledge distillation (KD) are typical techniques for compressing large language models, but it remains unclear how they should be applied at pretraining scale, especially to recent mixture-of-experts (MoE) models. In this work, we systematically study MoE compression in large-scale pretraining, focusing on three key questions: whether pruning provides a better initialization than training from scratch, how expert compression choices affect the final model after continued training, and which training strategy is most effective. We have the following findings: First, across depth, width, and expert compression, pruning a pretrained MoE consistently outperforms training the target architecture from scratch under the same training budget. Second, different one-shot expert compression methods converge to similar final performance after large-scale continual pretraining. Motivated by this, we introduce a simple partial-preservation expert merging strategy that improves downstream performance across most benchmarks. Third, combining KD with the language modeling loss outperforms KD alone, particularly on knowledge-intensive tasks. We further propose multi-token prediction (MTP) distillation, which yields consistent gains. Finally, given the same training tokens, progressive pruning schedules outperform one-shot compression, suggesting that gradual architecture transitions lead to better optimization trajectories. Putting it all together, we compress Qwen3-Next-80A3B to a 23A2B model that retains competitive performance. These results offer practical guidance for efficient MoE compression at scale.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

1. Introduction

Mixture-of-Experts (MoE) (Shazeer et al., 2017) has become a dominant architecture for scaling large language models (Jiang et al., 2024; Team, 2024; Yang et al., 2025a; Team, 2025a; 2026), but modern MoE LLMs remain expensive to pretrain and serve. Compressing a pretrained MoE into a smaller model that retains most of its capability at pretraining scale is therefore an important practical problem.

Structured pruning compresses models by removing entire architectural components (e.g., layers, attention heads, or experts) and delivers wall-clock speedups without specialized sparse kernels. Because pruning alone could degrade performance, knowledge distillation (KD) is commonly used to recover the loss by transferring knowledge from the teacher to the pruned student, and is widely believed to outperform continued pretraining with the standard language modeling (LM) objective. Despite extensive progress on dense models (Muralidharan et al., 2024), extending these compression paradigms to MoE models presents unique challenges. Specifically, MoE models introduce an additional compression dimension: experts, which can be pruned or merged. While recent studies (Jaiswal et al., 2025) thoroughly evaluate the one-shot performance of various expert compression methods, their efficacy following large-scale continual pretraining remains unexplored.

To bridge this gap, we revisit structured pruning and post-compression training for MoE LLMs by systematically investigating several practical questions: **(1) Initialization.** Does pruning a pretrained MoE model provide a stronger initialization than training an identical target architecture from scratch? **(2) Compression Strategy.** How do different expert compression strategies impact final performance after extensive continual pretraining? **(3) Training Recipe.** What is the optimal post-compression training recipe to facilitate performance recovery?

By exploring MoE-based LLM compression across depth, width, and experts via extensive continual pretraining, we present our key findings as follows: First, under the matched training tokens, pruning a pretrained MoE model to a target architecture provides a significantly better initialization than training from scratch, consistently improving both rea-

soning and generation performance. Second, we conduct a comprehensive empirical analysis of expert compression and propose a partial-preservation strategy. By comparing various pruning and merging criteria (e.g., routing frequency or scores, expert activations) under a 400B-token continual pretraining setting, we find that the final performance differences among one-shot expert pruning or merging methods are marginal, with no single approach dominating. Motivated by this observation and the critical need to balance pretrained expert specialization against the consolidation of discarded experts, we propose a strategy that explicitly retains the top half of target experts intact while merging the less critical remainder into them. This prevents representation homogenization and consistently enhances downstream evaluation performance. Third, we demonstrate that hybridizing next-token knowledge distillation (NTP KD) with a standard language modeling (LM) loss, regulated by a linear decay schedule, yields superior recovery on knowledge-intensive benchmarks compared to pure KD. To further elevate the compacted model, we propose multi-token prediction (Gloeckle et al., 2024) distillation (MTP KD). This paradigm extends the distillation objective beyond single tokens, fundamentally enhancing the backbone's training dynamics and representation quality, and improving the acceptance rate in multi-token speculative decoding. Finally, we study how to schedule pruning and distillation progressively when transitioning from a base architecture to a target architecture. Given a target configuration, we systematically compare direct one-stage compression against three progressive pruning schedules: depth-first, width-first, and joint. Across all configurations, progressive strategies consistently surpass one-shot pruning under an identical token budget. This confirms that staged capacity reduction provides a significantly smoother optimization trajectory for knowledge transfer.

Empirically, we demonstrate that our pruning and distillation recipe can compress the Qwen3-Next-80A3B (Team, 2025b) to a 23A2B model (approximately $4\times$ compression) with competitive downstream performance after continual pretraining across a broad suite of evaluations, including MMLU variants, BBH, GSM8K, coding, and Chinese benchmarks. Overall, our results provide practical guidance for compute-efficient MoE compression at pre-training scale (Team, 2026), clarifying (i) how structured pruning across depth/width/experts should be applied, (ii) how progressive schedules affect recovery, and (iii) which training objective is most effective during long post-compression training. Our main contributions are:

- We present a systematic study of large-scale MoE compression at pretraining scale, covering structured pruning initialization, expert compression, post-compression continual pretraining objectives, and progressive pruning schedules. We show that structured pruning provides a strong initialization, and that after large-scale continual pretraining, different one-shot expert pruning/merging methods yield similar final performance. We further propose a simple partial-preservation expert merging strategy that shows consistent improvement across benchmarks.

- We introduce the multi-token knowledge distillation that improves backbone model training and speculative decoding, and investigate different pretraining loss choices. Our experiments show that incorporating LM loss improves performance on knowledge-intensive benchmarks, while MTP KD yields consistent gains across the major benchmarks.

- We compare progressive pruning schedules and find that all progressive pruning strategies consistently outperform one-shot compression under the same final sparsity and total training tokens. Empirically, we compress Qwen3-Next-80A3B into a 23A2B model that achieves competitive performance across a wide range of benchmarks, including general reasoning, mathematics, and coding.

2. Related Work

Structured Pruning and Post-Compression Training in LLMs. Structured pruning improves LLM efficiency without specialized hardware, while post-compression training is often required to recover performance after pruning (Ma et al., 2023; Wang et al., 2025). Existing work studies width pruning in dense LLMs (Xia et al., 2024b; Ashkboos et al., 2024), depth pruning by removing transformer blocks (Men et al., 2024; Kim et al., 2024), and expert pruning/merging for MoE models (Li et al., 2024b; Lasby et al., 2025b; Lu et al., 2024). Recovery is typically performed with distillation or language modeling objectives for dense or partially compressed MoE models (Muralidharan et al., 2024; Tang et al., 2025; Li et al., 2025). In this work, we jointly combine depth/width pruning with expert pruning/merging for high-ratio MoE compression, and study post-compression continual pretraining with a focus on pruning initialization, expert merging, and recovery strategies. Moreover related works can be found in the appendix.

3. Method

3.1. Background and Notation.

Qwen3-Next (Team, 2025b) is a hybrid-attention MoE-based model with L layers, each block includes Gated DeltaNet (Yang et al., 2025b) or Gated Attention modules (Qiu et al., 2025b) with ratio $(L_{linear} : L_{full})$, MoE module with N_e regular experts and N_s shared experts, and RMSNorm modules.

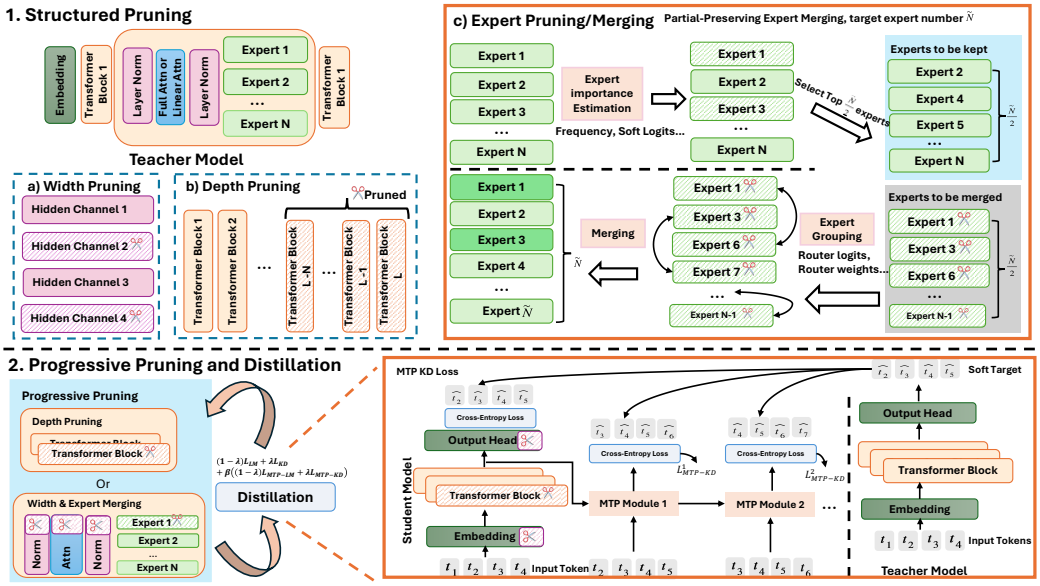

Figure 1. Overview of the SlimQwen. We first perform structured pruning on a teacher MoE model, including width pruning, depth pruning, and expert pruning/merging based on importance estimation and similarity, with a proposed partial-preservation strategy. We then adopt progressive pruning and distillation to gradually transform the teacher into the target architecture via staged pruning schedules (depth-first, width-first, or joint). Finally, we introduce a multi-token prediction (MTP) distillation, which extends standard next-token distillation by supervising multiple future tokens, improving training effectiveness.

For the MoE module, given an input token $x \in \mathbb{R}^{1\times d}$, we define n experts in total, including n_{routed} routed experts and n_{shared} shared experts ($n = n_{\text{routed}} + n_{\text{shared}}$). Each expert is a SwiGLU MLP:

$$\text{Expert}(x) = (\text{SiLU}(xW_{1e}) \odot (xW_{2e}))W_{3e}, \quad (1)$$

where $W_{1e}, W_{2e} \in \mathbb{R}^{d\times d_{\text{ff}}}$ and $W_{3e} \in \mathbb{R}^{d_{\text{ff}}\times d}$. The router produces top-k gating scores over the routed experts: $z(x) = \text{softmax}\big(\text{TopK}(xW^G, k)\big)$, $W^G \in \mathbb{R}^{d\times n_{\text{routed}}}$. In addition, we apply a separate shared gate $z_{\text{s}}(x) = \sigma(xw_{\text{sh}}) \in \mathbb{R}^{n_{shared}}$, $w_{\text{s}} \in \mathbb{R}^{d\times n_{shared}}$ for shared experts. The MoE output is

$$\text{MoE}(x) = \sum_{e=1}^{n_{\text{routed}}} z_e(x)\text{Expert}_e(x) + \sum_{s=1}^{n_{shared}} z_{\text{s}}(x)\text{Expert}_s(x). \quad (2)$$

Qwen3-Next uses the RMSNorm (Zhang & Sennrich, 2019) normalizing function

$$\text{RMSNorm}(X) = \frac{X}{\text{RMS}(X)} \odot \gamma, \quad (3)$$

$$\text{RMS}(X)_i = \sqrt{\frac{1}{d}\sum_{j=1}^{d} X_{ij}^2 + \epsilon} \quad (4)$$

where $\text{RMS}(X) \in \mathbb{R}^{n\times 1}$ is the root mean square computed over the hidden dimension for each token, and $\gamma \in \mathbb{R}^{1\times d}$ is the learnable scale parameter. The constant ϵ is added for numerical stability. The details of Gated DeltaNet and Gated Attention can be found in the Appendix Sec. A.1.

3.2. MoE-based Model Compression

In this work, we focus on exploring MoE-based Model compression across three dimensions: depth, width, and experts. We introduce the details of strategy for each dimension below.

Depth Pruning. Considering a model with L sequential layers $\{f_\ell\}_{\ell=1}^{L}$, we directly drop the last N layers of an L-layer model (Sun et al., 2026)[1]:

$$\mathcal{L}_{\text{keep}} = \{1, \ldots, L-N\}, \qquad \tilde{L} = L - N. \quad (5)$$

In our experiments, we prune the last 25% layers.

Width Pruning. For width pruning, we reduce the hidden dimension across the entire architecture, encompassing the hybrid attention, MoE, and normalization modules. We estimate the importance of each hidden dimension using activation statistics computed on a sampled calibration dataset $\mathcal{D}$ from our training dataset. Let $Z \in \mathbb{R}^{B\times n\times m}$ denote the output activation of a module for a batch size B, sequence length n, and hidden dimension m. We aggregate along the batch and sequence dimensions using mean absolute activation: $\text{Mean}(Z) := \frac{1}{Bn}\sum_{b=1}^{B}\sum_{t=1}^{n}\big|Z_{b,t,:}\big| \in \mathbb{R}^{m}$. Let $Y = \text{RMSNorm}(X) \in \mathbb{R}^{B\times n\times d}$ be the RMSNorm output

[1]We provide the performance comparison and discussion of different depth pruning methods in Appendix Sec. A.6. The last-layer pruning achieves better performance on both one-shot and continual pretraining settings.

. The hidden dimension importance are formulated as:

$$I_{\text{norm}}^{(k)} = \Big[\frac{\sum_{i=0}^{L} \text{Mean}\big(\text{RMSNorm}(X)\big)}{L}\Big]_k,\ k = 1, \ldots, d. \tag{6}$$

Given the target hidden size d_t, we retain the d_t hidden dimensions with the highest importance scores.

Expert Compression. Regarding expert compression, we compare various compression strategies, including pruning and merging. The initial step involves quantifying expert importance with various criteria. Given a set of calibration data, frequency-based criteria records the activated frequency while soft-logits method further weights frequency with the logits of router outputs for each expert. We also consider the router-weighted expert output activation (REAP) (Lasby et al., 2025a). Formally, for each MoE layer, let there be N routed experts $\mathcal{E} = \{E_1, \ldots, E_N\}$ and a router $R : \mathbb{R}^d \to \mathbb{R}^N$ that outputs routing logits $z(x) = R(x) \in \mathbb{R}^N, x \in \mathbb{R}^d$. For each token representation x, we select the top-k experts $\mathcal{A}(x) = \text{TopK}(z(x), k) \subseteq \{1, \ldots, N\}$. let $E_j(x)$ be the expert output. We can compute the frequency-based, soft-logits and REAP expert importance via:

$$I_i^{\text{Freq}} = \mathbb{E}_{x\sim\mathcal{C}}\Big[\mathbb{I}\big[i \in \mathcal{A}(x)\big]\Big], \tag{7}$$

$$I_i^{\text{Soft}} = \mathbb{E}_{x\sim\mathcal{C}}\Big[\frac{\mathbb{I}[i \in \mathcal{A}(x)] \cdot z_i(x)}{\sum_{j\in\mathcal{A}(x)} z_j(x)}\Big], \tag{8}$$

$$I_i^{\text{REAP}} = \frac{1}{|\mathcal{X}_i|} \sum_{x\in\mathcal{X}_i} z_i(x)\, \big\|E_i(x)\big\|_2, \qquad i = 1, \ldots, N, \tag{9}$$

where $\mathbb{I}[\cdot]$ is the indicator function. In practice, the expectation is computed by mean over all tokens in the calibration set.

For expert merging, we need to identify both the target clusters and the interpolation weights. We first quantify inter-expert similarities using router logits $z(x)$, router weights and output activation $E_j(x)$ among each expert. Given the above expert-importance scores, we preserve the highest-ranked experts. Each discarded expert is then merged into its nearest retained neighbor, using its importance score as the scaling factor. A central challenge in expert compression is striking an optimal balance between knowledge preservation and expert consolidation. Exclusively retaining top-ranked experts preserves highly salient knowledge but risks discarding experts that are individually less prominent yet functionally complementary. Conversely, constructing all target experts through aggressive merging can homogenize pretrained expert specialization, hindering performance recovery during continual pretraining. To navigate this trade-off, we propose a simple **partial-preservation merging strategy**: we retain half of the target experts intact, and construct the remainder by merging the discarded experts into selected merge bases. Formally, given a target number of retained experts $\tilde{N} < N$, we keep half target of experts with the largest importance scores: $\mathcal{S}_{\text{keep}} = \arg\text{topk}_{i\in\{1,\ldots,N\}} I_i$ with $|\mathcal{S}_{\text{keep}}| = \lfloor \tilde{N}//2 \rfloor$ and the pruned expert index is $\mathcal{S}_{\text{prune}} = \{1, \ldots, N\} \setminus \mathcal{S}_{\text{keep}}$. Finally, we select another $\tilde{N}/2$ experts from the remaining experts as merge bases, denoted by $\mathcal{S}_{\text{base}}$. For each $i \in \mathcal{S}_{\text{base}}$, we find its most similar partner $m(i) = \arg\max_{j\in\mathcal{S}_{\text{merge}}} \text{CosineSim}(i, j)$, and merge the two experts as

$$\tilde{E}_i = \frac{I_i}{I_i + I_{m(i)}} E_i + \frac{I_{m(i)}}{I_i + I_{m(i)}} E_{m(i)}. \tag{10}$$

The final compressed expert set is composed of the preserved experts and the merged experts. For both expert pruning and expert merging, we prune the corresponding router weight for continual pretraining. A detailed algorithm description can be found in Algorithm 1. We choose half of the target experts as a simple and symmetric design choice. Intuitively, preserving too few experts weakens parameter inheritance, whereas preserving too many leaves limited room for consolidation. Keeping roughly half provides a robust compromise in our evaluated setting. We discuss this more in Limitation section.

3.3. Distillation Pretraining

MTP Distillation Loss. We use Multi-Token Prediction (MTP) modules (Gloeckle et al., 2024) to predict additional future tokens. The MTP module consists of a embedding layer $\text{Emb}(\cdot)$ and a output head $\text{OutHead}(\cdot)$, which are shared with the backbone models. Moreover, a Transformer block $\text{TRM}_k(\cdot)$ and a projection matrix $M_k \in \mathbb{R}^{d\times 2d}$ are included in the MTP module. For the i-th input token t_i, at prediction depth $k \in \{1, \ldots, D\}$, we first combine the representation of the i-th token at depth $k-1$, denoted by $h_i^{k-1} \in \mathbb{R}^d$, with the embedding of the $(i+k)$-th token $\text{Emb}(t_{i+k}) \in \mathbb{R}^d$ via a linear projection:

$$h_i'^k = M_k\Big[\text{RMSNorm}(h_i^{k-1});\ \text{RMSNorm}\big(\text{Emb}(t_{i+k})\big)\Big], \tag{11}$$

where $[\cdot;\cdot]$ denotes concatenation. In particular, when $k = 1$, h_i^0 refers to the token representation produced by the main model. The combined representation is then fed into the k-th Transformer block to produce the current-depth representation: $h_{1:T-k}^k = \text{TRM}_k\big(h_{1:T-k}'^k\big)$, where T is the sequence length and $1{:}T-k$ denotes slicing. Finally, given h_i^k as input, the shared output head computes the probability distribution for the k-th additional prediction token: $p_{i+k}^k = \text{OutHead}(h_i^k) \in \mathbb{R}^V$, where V is the vocabulary size. The output head $\text{OutHead}(\cdot)$ linearly maps h_i^k to logits and applies $\text{Softmax}(\cdot)$ to obtain probabilities.

For each prediction depth $k \in \{1, \ldots, D\}$, the k-th MTP module produces a student distribution $p_{i+k}^k \in \mathbb{R}^V$ for

position $i+k$. The MTP LM loss can be written as:

$$\mathcal{L}_{\text{MTP-LM}} = \frac{1}{D}\sum_{k=1}^{D}(-\frac{1}{T-k}\sum_{i=1}^{T-k}\log p_{i+k}^{k}[t_{i+k}]). \quad (12)$$

Besides using ground-truth one-hot labels, we distill from a teacher model that provides a soft target distribution $q_{i+k} \in \mathbb{R}^V$ at the same position. We minimize the KL-divergence between teacher and student:

$$\mathcal{L}_{\text{MTP-KD}} = -\frac{1}{D}\sum_{k=1}^{D}(\frac{1}{T-k}\sum_{i=1}^{T-k}\sum_{v=1}^{V} q_{i+k}[v]\log p_{i+k}^{k}[v]). \quad (13)$$

where T is the input sequence length and V is the vocabulary size. Therefore, we train the model with four terms: (i) standard language modeling loss $\mathcal{L}_{\text{LM}}$ and knowledge distillation loss $\mathcal{L}_{\text{KD}}$ on the backbone output, MTP LM loss $\mathcal{L}_{\text{MTP-LM}}$ and MTP distillation loss $\mathcal{L}_{\text{MTP-KD}}$. The total objective is

$$\mathcal{L} = (1-\lambda)\,\mathcal{L}_{\text{LM}} + \lambda\,\mathcal{L}_{\text{KD}} + \beta\,((1-\lambda)\mathcal{L}_{\text{MTP-LM}} + \lambda\mathcal{L}_{\text{MTP-KD}}). \quad (14)$$

where λ and β are hyperparameters, which balance KD and LM loss, and backbone loss and MTP loss respectively.

Progressive Pruning and Distillation. Directly compressing a teacher model to a compact target architecture often induces substantial knowledge loss. To ensure a smoother transfer of pretrained capabilities, we explore three progressive, two-stage distillation schedules. Each schedule interleaves structural pruning with a fixed-token distillation phase, differing primarily in their reduction priorities for depth and width. **Depth-first** allocates half of the layer reduction to the first stage while maintaining the original width, leaving the remaining depth and the entire width reduction for the second stage. Conversely, **Width-first** executes half of the width reduction in the first stage while keeping the depth intact, completing the remaining width and the full depth reduction in the final stage. Finally, the **Joint** strategy simultaneously reduces both depth and width by half of their respective targets in the first stage, with the remaining halves pruned in the second stage to reach the final configuration. Through this exploration, we aim to identify the optimal structural reduction trajectory that maximizes performance recovery during continual pretraining.

4. Experiments

4.1. Experimental Setup

Base Model and Pruning Setup. Unless otherwise noted, our experiments are conducted based on an 80A3B hybrid MoE-based model, which includes 48 transformer blocks with 12 full attention and 36 linear attention layers. Each full attention has 16 query heads and 2 key/value heads with 256 head dim. The gated attention (Qiu et al., 2025b) is incorporated. For the MoE layers, each module contains a total of 512 experts, with 10 routed experts and 1 shared expert activated per token. The intermediate size is 512 and the hidden size is 2048. The model is trained with the multi-token prediction (MTP) module. More architecture details can be found in Appendix Table 2. For depth pruning, we remove 12 transformer blocks (3 full, 9 linear attention). In the remaining layers, we reduce the hidden size from 2048 to 1536. Additionally, we merge the 512 experts into 256 per MoE module and the compacted model activates only 8 routed experts with 1 shared expert per token. We randomly use 1024 samples as calibration set to compute the importance metric.

Training Settings. We evaluate our models under two training budgets: 120B and 400B high-quality, diverse tokens, with global batch sizes of 512 and 1024, respectively. The peak learning rate is set to 4e-4, decaying to 3e-5 via a cosine schedule with 2000 warmup steps. The distillation loss weight λ decays linearly from 1 to 0.75, while the MTP distillation weight β follows a cosine decay from 0.3 to 0.1. We explain the detailed experiment settings in each section and details can be found in Appendix Table 3.

Evaluation. We evaluate the few-shot performance of our models across a wide range of benchmarks. These include MMLU (Hendrycks et al., 2021), MMLU-Redux (Gema et al., 2025) and MMLU-Pro (Wang et al., 2024) for general knowledge; BBH (Suzgun et al., 2022) for reasoning; GSM-8K (Cobbe et al., 2021) for mathematics; EvalPlus (Liu et al., 2023) for coding, C-Eval (Huang et al., 2023) and CMMLU (Li et al., 2024a) for Chinese proficiency. We provide more evaluation in Appendix Sec. A.10.

4.2. Results

Q1: Does pruning provide a better initialization for MoE in large-scale pretraining? We first validate the effectiveness of training from a pruned MoE model in pretraining. As detailed in Table 1, both setups are trained for 120B tokens using knowledge distillation (KD) from the Qwen3-Next teacher. Compared to random initialization, the pruned model demonstrates striking superiority, achieving an average score of 73.45 against 61.66 (+11.79 points). This consistent improvement spans diverse domains, including knowledge (MMLU), math (GSM-8K), and coding (EvalPlus). Remarkably, the pruned architecture recovers 86.5% of the teacher's performance (73.45 vs. 82.68) despite being 3.4× smaller, suggesting that structured pruning successfully preserves task-critical weights to form an informative starting point. Furthermore, the training trajectories (Figure 2) corroborate these findings: pruned initialization yields considerably faster convergence and lower language modeling (LM) loss than random initialization, with the

Table 1. The result comparison of models trained from scratch and initialized from pruned weights. The results show training from a pruned model brings benefits to the final model under the same training budget. †Here, the KD loss refers to the combined loss in Eq. 14.

Method	MMLU	MMLU-Pro	MMLU-Redux	BBH	GSM-8K	EvalPlus	C-Eval	CMMLU	Avg.
Qwen3-Next-80A3B	85.22	62.86	84.45	85.12	90.07	74.12	90.33	89.27	82.68
Random Init + KD Loss†	65.06	34.54	65.66	56.01	73.35	58.67	70.11	69.85	61.66
Pruned + LM Loss	72.76	48.24	71.89	64.94	81.84	67.05	76.51	76.51	69.96
Pruned + KD Loss†	**75.67**	**51.19**	**74.37**	**72.29**	**83.17**	**69.30**	**80.67**	**80.95**	**73.45**

combined "Pruned + KD" recipe achieving the lowest loss.

Q2: How do different expert compression strategies impact final performance? To evaluate various expert compression strategies, we compress a 24A2B MoE model to a 6A1B architecture and continually pretrain for 400B tokens. As summarized in Table 4, *no single one-shot pruning or merging method establishes consistent superiority across all downstream tasks,* even if some models show higher performance on certain benchmarks (e.g. frequency-based router logits grouping method achieves 60.17 on BBH). A possible explanation is that one-shot expert compression (coarse-grained pruning or merging) methods are unable to preserve the performance of all benchmarks consistently. Furthermore, partial expert preservation during the merging experts yields consistent improvements across major benchmarks, including MMLU, MMLU-Pro, and GSM8K.

Q3: What constitutes an effective training recipe for compressed MoEs? To establish an effective post-compression continual training recipe under pre-training setting, we evaluate various loss configurations on a 23A2B model pruned from Qwen3-Next-80A3B and trained for 120B tokens (Table 6). Our analysis reveals several findings: Combining next-token prediction knowledge distillation (NTP KD) with a standard language modeling (LM) loss outperforms pure distillation, particularly on knowledge-intensive benchmarks such as MMLU (from 74.16 to 74.93) and MMLU-Pro (from 50.97 to 51.44). Furthermore, ablations demonstrate that integrating multi-token prediction knowledge distillation (MTP KD) into either pure NTP KD or a comprehensive joint objective (NTP KD + LM + MTP loss) improves the performance on several knowledge-intensive benchmarks. Beyond backbone quality, MTP KD yields substantial efficiency gains for speculative decoding across both pretraining and supervised fine-tuning (SFT), as shown in Table 7. We report the results on benchmarks including, HumanEval, GSM8K, WMT22(Kocmi et al., 2022) for pretraining stage, RepoQA(Liu et al., 2024), MT-Bench(Chen et al., 2026) and SpecBench(Xia et al., 2024a) for SFT stage. The results show that MTP KD consistently improves the multi-token acceptance rate from *acc_1* to *acc_4* on all benchmarks. A notable pattern is that the gains from MTP KD are often larger for longer accepted token sequence. This suggests that MTP KD is particularly helpful for improving the efficiency of multi-token generation, making the drafted tokens more likely to be accepted by the verifier model during speculative decoding. Overall, these results indicate MTP KD not only improves backbone training quality, but also brings practical benefits for speculative decoding.

Progressive pruning and distillation. Building upon the one-shot strategy, we further explore the efficacy of progressive pruning and distillation. Given the final target architectural configuration, we progressively prune the base model using three strategies: depth-first, width-first, and joint pruning, each conducted in two stages as described in Section 3.3. In the first stage, the intermediate pruned model is trained with 40B tokens. We then further prune it to the final target configuration and continue training on the remaining 360B tokens. The results are shown in Table 8. Overall, progressive pruning and distillation consistently outperform one-stage pruning trained directly on 400B tokens, demonstrating the benefit of gradual model compression during continual pretraining. In particular, MMLU improves from 75.86 (one-stage) to 77.39 (depth-first) and 77.14 (width-first), while MMLU-Redux shows substantial improvements, from 75.41 to 78.01 and 77.07. These findings confirm that a progressive trajectory mitigates information loss and better transfers pretrained knowledge. We further provide the results for more fine-grained stage schedules in Appendix Sec. A.9. However, However, more fine-grained stage partitions do not provide additional benchmark performance gains. Given the superior overall performance, we officially designate the depth-first progressive model as **SlimQwen**.

5. Conclusion

In this paper, we explore the pruning and distillation in MoE model pretraining. We show that structured pruning, even at high compression ratios, provides a strong initialization for continual pretraining, while different expert pruning and merging metrics exhibit only minor differences after large-scale pretraining. We further propose a simple partial-preservation expert merging strategy and demonstrate consistent performance improvements across major benchmarks. For distillation, we investigate the effectiveness of progressive pruning and distillation, as well as the role of LM loss as a complementary training objective. We propose a novel multi-token prediction (MTP) distillation objective for pretraining, demonstrating consistent performance gains across major benchmarks.

References

Ashkboos, S., Croci, M. L., do Nascimento, M. G., Hoefler, T., and Hensman, J. Slicegpt: Compress large language models by deleting rows and columns, 2024. URL https://arxiv.org/abs/2401.15024.

Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q., and Sutton, C. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.07732.

Cao, M., Li, G., Ji, J., Zhang, J., Ma, X., Liu, S., and Yin, L. Condense, don't just prune: Enhancing efficiency and performance in moe layer pruning, 2025. URL https://arxiv.org/abs/2412.00069.

Chen, J., Feng, A., Zhao, Z., Garza, J., Nurbek, G., Qin, C., Maatouk, A., Tassiulas, L., Gao, Y., and Ying, R. Mtbench: A multimodal time series benchmark for temporal reasoning and question answering, 2026. URL https://arxiv.org/abs/2503.16858.

Chen, W., Lin, Y., Zhou, Z., Huang, H., Jia, Y., Cao, Z., and Wen, J.-R. Icleval: Evaluating in-context learning ability of large language models, 2024. URL https://arxiv.org/abs/2406.14955.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Gema, A. P., Leang, J. O. J., Hong, G., Devoto, A., Mancino, A. C. M., Saxena, R., He, X., Zhao, Y., Du, X., Madani, M. R. G., Barale, C., McHardy, R., Harris, J., Kaddour, J., van Krieken, E., and Minervini, P. Are we done with mmlu?, 2025. URL https://arxiv.org/abs/2406.04127.

Gloeckle, F., Idrissi, B. Y., Rozière, B., Lopez-Paz, D., and Synnaeve, G. Better & faster large language models via multi-token prediction. In *Forty-first International Conference on Machine Learning*, 2024.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

Huang, Y., Bai, Y., Zhu, Z., Zhang, J., Zhang, J., Su, T., Liu, J., Lv, C., Zhang, Y., Lei, J., Fu, Y., Sun, M., and He, J. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models, 2023. URL https://arxiv.org/abs/2305.08322.

Jaiswal, A., Wang, J., Li, Y., Li, P., Chen, T., Wang, Z., Wang, C., Pang, R., and Du, X. Finding fantastic experts in moes: A unified study for expert dropping strategies and observations, 2025. URL https://arxiv.org/abs/2504.05586.

Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D. S., de Las Casas, D., Hanna, E. B., Bressand, F., Lengyel, G., Bour, G., Lample, G., Lavaud, L. R., Saulnier, L., Lachaux, M., Stock, P., Subramanian, S., Yang, S., Antoniak, S., Scao, T. L., Gervet, T., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mixtral of experts. *CoRR*, abs/2401.04088, 2024.

Kim, B.-K., Kim, G., Kim, T.-H., Castells, T., Choi, S., Shin, J., and Song, H.-K. Shortened llama: Depth pruning for large language models with comparison of retraining methods, 2024. URL https://arxiv.org/abs/2402.02834.

Kocmi, T., Bawden, R., Bojar, O., Dvorkovich, A., Federmann, C., Fishel, M., Gowda, T., Graham, Y., Grundkiewicz, R., Haddow, B., Knowles, R., Koehn, P., Monz, C., Morishita, M., Nagata, M., Nakazawa, T., Novák, M., Popel, M., and Popović, M. Findings of the 2022 conference on machine translation (WMT22). In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, 2022. URL https://aclanthology.org/2022.wmt-1.1/.

Lasby, M., Lazarevich, I., Sinnadurai, N., Lie, S., Ioannou, Y., and Thangarasa, V. REAP the Experts: Why Pruning Prevails for One-Shot MoE compression, 2025a. URL https://arxiv.org/abs/2510.13999v1. arXiv:2510.13999v1 [cs].

Lasby, M., Lazarevich, I., Sinnadurai, N., Lie, S., Ioannou, Y., and Thangarasa, V. Reap the experts: Why pruning prevails for one-shot moe compression, 2025b. URL https://arxiv.org/abs/2510.13999.

Li, H., Zhang, Y., Koto, F., Yang, Y., Zhao, H., Gong, Y., Duan, N., and Baldwin, T. Cmmlu: Measuring massive multitask language understanding in chinese, 2024a. URL https://arxiv.org/abs/2306.09212.

Li, P., Zhang, Z., Yadav, P., Sung, Y.-L., Cheng, Y., Bansal, M., and Chen, T. Merge, then compress: Demystify efficient smoe with hints from its routing policy, 2024b. URL https://arxiv.org/abs/2310.01334.

Li, Z., Liang, C., Zhang, Z., Hong, I., Kim, Y. J., Chen, W., and Zhao, T. Slimmoe: Structured compression of large moe models via expert slimming and distillation, 2025. URL https://arxiv.org/abs/2506.18349.

Liu, J., Xia, C. S., Wang, Y., and Zhang, L. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation, 2023. URL https://arxiv.org/abs/2305.01210.

Liu, J., Tian, J. L., Daita, V., Wei, Y., Ding, Y., Wang, Y. K., Yang, J., and Zhang, L. Repoqa: Evaluating long context code understanding, 2024. URL https://arxiv.org/abs/2406.06025.

Lu, X., Liu, Q., Xu, Y., Zhou, A., Huang, S., Zhang, B., Yan, J., and Li, H. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models, 2024. URL https://arxiv.org/abs/2402.14800.

Ma, K., Du, X., Wang, Y., Zhang, H., Wen, Z., Qu, X., Yang, J., Liu, J., Liu, M., Yue, X., Huang, W., and Zhang, G. Kor-bench: Benchmarking language models on knowledge-orthogonal reasoning tasks, 2025. URL https://arxiv.org/abs/2410.06526.

Ma, X., Fang, G., and Wang, X. Llm-pruner: On the structural pruning of large language models. In *Advances in Neural Information Processing Systems*, 2023.

Men, X., Xu, M., Zhang, Q., Wang, B., Lin, H., Lu, Y., Han, X., and Chen, W. Shortgpt: Layers in large language models are more redundant than you expect, 2024. URL https://arxiv.org/abs/2403.03853.

Muralidharan, S., Sreenivas, S. T., Joshi, R., Chochowski, M., Patwary, M., Shoeybi, M., Catanzaro, B., Kautz, J., and Molchanov, P. Compact language models via pruning and knowledge distillation, 2024. URL https://arxiv.org/abs/2407.14679.

Peng, H., Lv, X., Bai, Y., Yao, Z., Zhang, J., Hou, L., and Li, J. Pre-training distillation for large language models: A design space exploration, 2024. URL https://arxiv.org/abs/2410.16215.

Qiu, Z., Huang, Z., Zheng, B., Wen, K., Wang, Z., Men, R., Titov, I., Liu, D., Zhou, J., and Lin, J. Demons in the detail: On implementing load balancing loss for training specialized mixture-of-expert models, 2025a. URL https://arxiv.org/abs/2501.11873.

Qiu, Z., Wang, Z., Zheng, B., Huang, Z., Wen, K., Yang, S., Men, R., Yu, L., Huang, F., Huang, S., Liu, D., Zhou, J., and Lin, J. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free, 2025b. URL https://arxiv.org/abs/2505.06708.

Romanou, A., Foroutan, N., Sotnikova, A., Chen, Z., Nelaturu, S. H., Singh, S., Maheshwary, R., Altomare, M., Haggag, M. A., Amayuelas, A., et al. Include: Evaluating multilingual language understanding with regional knowledge. *arXiv preprint arXiv:2411.19799*, 2024.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q. V., Hinton, G. E., and Dean, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *5th International Conference on Learning Representations*, 2017.

Shi, F., Suzgun, M., Freitag, M., Wang, X., Srivats, S., Vosoughi, S., Chung, H. W., Tay, Y., Ruder, S., Zhou, D., Das, D., and Wei, J. Language models are multilingual chain-of-thought reasoners, 2022.

Sun, W., Song, X., Li, P., Yin, L., Zheng, Y., and Liu, S. The curse of depth in large language models, 2026. URL https://arxiv.org/abs/2502.05795.

Suzgun, M., Scales, N., Schärli, N., Gehrmann, S., Tay, Y., Chung, H. W., Chowdhery, A., Le, Q. V., Chi, E. H., Zhou, D., and Wei, J. Challenging big-bench tasks and whether chain-of-thought can solve them, 2022. URL https://arxiv.org/abs/2210.09261.

Tang, S., Sieberling, O., Kurtic, E., Shen, Z., and Alistarh, D. Darwinlm: Evolutionary structured pruning of large language models, 2025. URL https://arxiv.org/abs/2502.07780.

Team, G. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *CoRR*, abs/2507.06261, 2025a. doi: 10.48550/ARXIV.2507.06261.

Team, P., Du, X., Yao, Y., Ma, K., Wang, B., Zheng, T., Zhu, K., Liu, M., Liang, Y., Jin, X., Wei, Z., Zheng, C., Deng, K., Gavin, S., Jia, S., Jiang, S., Liao, Y., Li, R., Li, Q., Li, S., Li, Y., Li, Y., Ma, D., Ni, Y., Que, H., Wang, Q., Wen, Z., Wu, S., Hsing, T., Xu, M., Yang, Z., Wang, Z. M., Zhou, J., Bai, Y., Bu, X., Cai, C., Chen, L., Chen, Y., Cheng, C., Cheng, T., Ding, K., Huang, S., Huang, Y., Li, Y., Li, Y., Li, Z., Liang, T., Lin, C., Lin, H., Ma, Y., Pang, T., Peng, Z., Peng, Z., Qi, Q., Qiu, S., Qu, X., Quan, S., Tan, Y., Wang, Z., Wang, C., Wang, H., Wang, Y., Wang, Y., Xu, J., Yang, K., Yuan, R., Yue, Y., Zhan, T., Zhang, C., Zhang, J., Zhang, X., Zhang, X., Zhang, Y., Zhao, Y., Zheng, X., Zhong, C., Gao, Y., Li, Z., Liu, D., Liu, Q., Liu, T., Ni, S., Peng, J., Qin, Y., Su, W., Wang, G., Wang, S., Yang, J., Yang, M., Cao, M., Yue, X., Zhang, Z., Zhou, W., Liu, J., Lin, Q., Huang, W., and Zhang, G. Supergpqa: Scaling llm evaluation across 285 graduate disciplines, 2025. URL https://arxiv.org/abs/2502.14739.

Team, Q. Qwen2.5: A party of foundation models, September 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

Team, Q. Qwen3-next: Towards ultimate training & inference efficiency, 2025b.

Team, Q. Qwen3.5: Accelerating productivity with native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Wang, Y., Ma, X., Zhang, G., Ni, Y., Chandra, A., Guo, S., Ren, W., Arulraj, A., He, X., Jiang, Z., Li, T., Ku, M., Wang, K., Zhuang, A., Fan, R., Yue, X., and Chen, W. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark, 2024. URL https://arxiv.org/abs/2406.01574.

Wang, Y., Ma, M., Wang, Z., Chen, J., Shan, L., Yang, Q., Xu, D., Liu, M., and Qin, B. CFSP: an efficient structured pruning framework for llms with coarse-to-fine activation information. In *Proceedings of the 31st International Conference on Computational Linguistics*, 2025.

Xia, H., Yang, Z., Dong, Q., Wang, P., Li, Y., Ge, T., Liu, T., Li, W., and Sui, Z. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. In *Findings of the Association for Computational Linguistics*, 2024a. URL https://aclanthology.org/2024.findings-acl.456.

Xia, M., Gao, T., Zeng, Z., and Chen, D. Sheared llama: Accelerating language model pre-training via structured pruning, 2024b. URL https://arxiv.org/abs/2310.06694.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Zhou, J., Lin, J., Dang, K., Bao, K., Yang, K., Yu, L., Deng, L., Li, M., Xue, M., Li, M., Zhang, P., Wang, P., Zhu, Q., Men, R., Gao, R., Liu, S., Luo, S., Li, T., Tang, T., Yin, W., Ren, X., Wang, X., Zhang, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Zhang, Y., Wan, Y., Liu, Y., Wang, Z., Cui, Z., Zhang, Z., Zhou, Z., and Qiu, Z. Qwen3 technical report, 2025a. URL https://arxiv.org/abs/2505.09388.

Yang, S., Kautz, J., and Hatamizadeh, A. Gated delta networks: Improving mamba2 with delta rule, 2025b. URL https://arxiv.org/abs/2412.06464.

Yang, Y., Cao, Z., and Zhao, H. Laco: Large language model pruning via layer collapse, 2024. URL https://arxiv.org/abs/2402.11187.

Zhang, B. and Sennrich, R. Root mean square layer normalization. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems*, 2019.

A. Appendix

A.1. Architecture Details

We provide the architecture details of the original teacher model and the pruned student models in Table 2. Specifically, for the Gated Attention, given input hidden states $X \in \mathbb{R}^{n\times d}$, where d is the model hidden size and h_q is the number of query head, Gated Attention can be formulated as:

$$\mathrm{GatedAttn}(X) = \mathrm{Concat}\big(\mathrm{head}_1 \odot g_i(X), \ldots, \mathrm{head}_{h_q} \odot g_{h_q}(X)\big)W_O,\ g_i(X) = \sigma(Xw_g^{(i)}) \in \mathbb{R}^{n\times 1}, \tag{15}$$

where $W_O \in \mathbb{R}^{(h_q d_{\mathrm{head}})\times d}$ is the output matrix, $\sigma(\cdot)$ indicates the sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$ and $w_g^{(i)} \in \mathbb{R}^{d\times 1}$ is a learnable gate weight. The attention head is computed by scaled dot-product attention: $\mathrm{head}_i = \mathrm{Attn}\big(Q^{(i)}, K^{(m(i))}, V^{(m(i))}\big), \mathrm{Attn}(Q,K,V) = \mathrm{softmax}(\frac{QK^\top}{\sqrt{d_{\mathrm{head}}}})V$. The per-head query, key and value projections are $Q^{(i)} = XW_Q^{(i)}, K^{(j)} = XW_K^{(j)}, V^{(j)} = XW_V^{(j)}$ with learnable parameters $W_Q^{(i)}, W_K^{(j)}, W_V^{(j)} \in \mathbb{R}^{d\times d_{\mathrm{head}}}$. We use Grouped-Query Attention (GQA) with h_q query heads and h_{kv} key/value heads. For the Gated DeltaNet, we maintain a linear state matrix $S_t \in \mathbb{R}^{d_v\times d_k}, q_t \in \mathbb{R}^{d_k}, k_t \in \mathbb{R}^{d_k}, v_t \in \mathbb{R}^{d_v}$. The gated delta rule updates the state as

$$S_t \;=\; S_{t-1}\Big(\alpha_t\big(I - \beta_t k_t k_t^\top\big)\Big) \;+\; \beta_t\, v_t k_t^\top, \qquad \alpha_t \in (0,1),\ \beta_t \in (0,1). \tag{16}$$

The token-mixing output is read out by $y_t = S_t q_t \in \mathbb{R}^{d_v}$. We can map it back to the model dimension d: $Y_t = y_t W_{\mathrm{out}} \in \mathbb{R}^d$, $W_{\mathrm{out}} \in \mathbb{R}^{d_v\times d}$. In our implementation, d_k corresponds to the Q/K hidden size and d_v corresponds to the V hidden size.

Table 2. Model configurations and parameter counts for different MoE variants.

Model	d_{model}	Self Attn					MoE				n_{MTP}
		n_{qhead}	n_{kvhead}	d_{head}	n_{layer}	Attn-Gate	d_{expert}	n_{expert}	$n_{\mathrm{shared\text{-}expert}}$	top-k	
80B-A3B	2048	16	2	256	12	Yes	512	512	1	10	1
SlimQwen-23A2B	1536	16	2	256	8	Yes	512	256	1	8	1
23B-A2B	2048	16	2	256	7	Yes	512	256	1	8	1
SlimQwen-6A1B	1280	16	2	256	5	Yes	512	128	1	8	1

Model	Linear Attn								# Total Param	# Act. Param
	n_{vhead}	n_{qkhead}	d_{vhead}	d_{qkhead}	d_{conv}	d_{inner}	n_{layer}	Attn-Gate		
80B-A3B	32	16	128	128	4	4096	36	No	80B	3.8B
SlimQwen-23A2B	32	16	128	128	4	4096	24	No	23B	2.0B
23B-A2B	32	16	128	128	4	4096	21	No	23B	2B
SlimQwen-6A1B	32	16	128	128	4	2560	15	No	6B	1B

A.2. Training Hyperparameters

We provide the detailed pretraining hyperparameters in Table 3.

A.3. Implementation Detail

Our codebase is built upon Megatron-LM. Following Qwen3 MoE models (Yang et al., 2025a), we apply the global-batch load balancing loss (Qiu et al., 2025a) for MoE. The calibration data is sampled from the pretraining data. For progressive pruning distillation, we train all models using a single-stage learning rate decay schedule, such that the second stage starts from the learning rate reached at the final step of the first stage. We use the AdamW optimizer and apply the default hyperparameter settings for the optimizer. For speculative decoding, we use the MTP module as the draft model and backbone model as the verification model, and report *acc_0* as the acceptance rate of generating one token with the MTP module, *acc_1* as that of generating two tokens, and so on. We also provide the pseudo-code of the partial-preservation expert merging strategy, as shown in Algorithm 1.

A.4. Training curve

The results are shown in Figure 2.

Table 3. Training hyperparameters for the 120B-token and 400B-token settings. The two settings differ only in the global batch size.

Hyperparameter	120B Setting	400B Setting
Training tokens	120B	400B
Global batch size	512	1024
Learning rate	4×10^{-4}	
LR schedule	Cosine decay	
Minimum learning rate	3×10^{-5}	
Warmup steps	2000	
KD loss weight λ	Linear decay from 1.0 to 0.75	
MTP distillation weight β	Cosine decay from 0.3 to 0.1	
Calibration samples	1024	
Training platform	Alibaba Cloud	

Algorithm 1 Partial-preservation Expert Merging Strategy

Require: Experts $\{E_i\}_{i=1}^{N}$, target expert number $\tilde{N}$, importance scores $\{S_i\}_{i=1}^{N}$
Ensure: Compressed experts $\tilde{\mathcal{E}}$
0: $S_{\text{keep}} \leftarrow \arg\text{topk}_{i \in \{1,\dots,N\}} S_i$, where $|S_{\text{keep}}| = \lfloor \tilde{N}/2 \rfloor$
0: Select $S_{\text{base}} \subset \{1,\dots,N\} \setminus S_{\text{keep}}$ such that $|S_{\text{base}}| = \tilde{N} - |S_{\text{keep}}|$
0: **for all** $i \in \{1,\dots,N\} \setminus (S_{\text{keep}} \cup S_{\text{base}})$ **do**
0: $\quad m(i) \leftarrow \arg\max_{j \in S_{\text{base}}} \text{CosineSim}(i,j)$
0: $\quad$ Assign i to the merge group of $m(i)$
0: **end for**
0: **for all** $j \in S_{\text{base}}$ **do**
0: $\quad$ Merge all experts assigned to j into $\tilde{E}_j$
0: **end for**
0: **return** $\tilde{\mathcal{E}} = \{E_i : i \in S_{\text{keep}}\} \cup \{\tilde{E}_j : j \in S_{\text{base}}\}$ =0

A.5. Results of Different Pruning/Merging Methods.

The results are shown in Table 4.

A.6. Comparison of Different Depth Pruning Methods

We compare different depth pruning methods including activation similarity based method and pruning the last several layers directly, as shown in Table 5. Formally, Let $h_\ell \in \mathbb{R}^{n \times d}$ be the activation of layer ℓ and $a_\ell = \frac{1}{n}\sum_{t=1}^{n} h_{\ell,t} \in \mathbb{R}^d$ be its token-mean pooled vector. We compute adjacent-layer cosine similarity:

$$c_\ell = \frac{\langle a_{\ell-1}, a_\ell \rangle}{\|a_{\ell-1}\|_2 \|a_\ell\|_2}, \qquad \ell = 2, \dots, L. \tag{17}$$

Let $\ell^\star = \arg\max_{\ell \in \{2,\dots,L\}} c_\ell$ be the starting index, and prune a contiguous chunk of N layers: $\mathcal{S}_{\text{prune}} = \{\ell^\star, \ell^\star + 1, \dots, \ell^\star + N - 1\}$ and $\mathcal{S}_{\text{keep}} = \{1,\dots,L\} \setminus \mathcal{S}_{\text{prune}}$.

We conduct the experiments on a pretrained 15A3B teacher model with 24 layers. We utilize the same 1024 calibration dataset discussed above to compute the layer activation and prune 4 layers in the one-shot setting. The activation-based pruning method tends to prune the middle layers. The results from the table indicate that directly pruning the last 4 layers causes only minor degradation (e.g from 75.62 to 73.86 on MMLU) while activation-based method shows substantially larger performance drops (e.g from 75.62 to 41.95 on MMLU). The results also align with the observation from (Sun et al., 2026). After 120B tokens of post-compression KD, last-layer pruning still recovers better performance than the activation-based method. One interesting phenomenon from the table is that the performance of the model trained with 120B tokens is worse than the one-shot counterpart on benchmarks such as MMLU and CMMLU. A possible explanation is that the one-shot performance is already close to that of the teacher model, leaving a relatively small knowledge gap to recover.

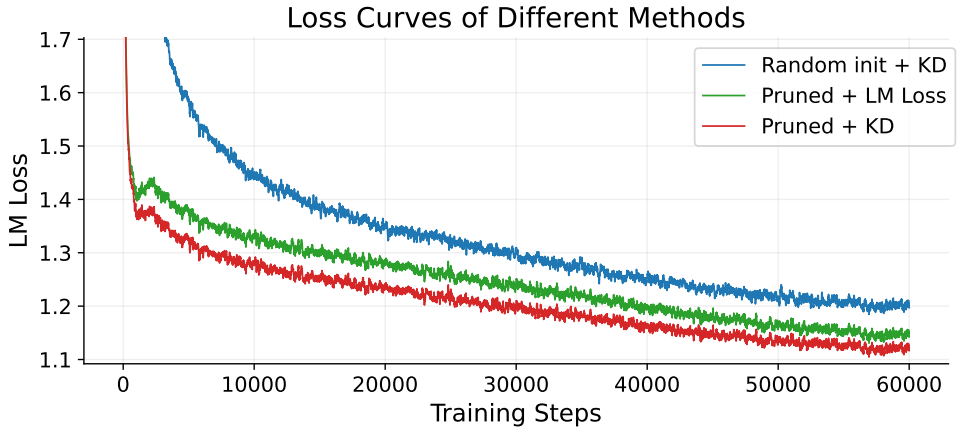

Figure 2. Training loss curves under different initialization and training objectives. Models initialized from pruned checkpoints converge faster and achieve lower LM loss than random initialization. Incorporating KD further improves optimization, with Pruned + KD consistently achieving the lowest loss, followed by Pruned + LM Loss, demonstrating the advantage of pruning-based initialization and distillation for efficient and effective training.

Table 4. Performance comparison of models with and without the partial-preservation expert merging strategy during expert pruning, and across different pruning and merging methods after continual pretraining. The results demonstrate that (1) the partial-preservation expert merging strategy leads to performance gains on major benchmarks, and (2) no single model exhibits uniformly superior performance across all evaluation tasks.

Prune/Merging	Imp. Metric	Group Method	Preserve	MMLU	MMLU-Pro	MMLU-Redux	BBH	GSM-8K	EvalPlus	C-Eval	CMMLU
Expert Merging	Soft Logits	Router Weights	No	69.05	42.62	68.47	59.12	71.08	50.35	71.08	72.20
Expert Merging	Soft Logits	Router Weights	Yes	**69.28**	**44.05**	67.64	**59.81**	**74.18**	48.00	**71.15**	**72.73**
Expert Pruning	Soft Logits	-	-	68.74	**43.23**	**69.11**	58.97	**74.30**	51.69	71.67	72.26
Expert Pruning	REAP	-	-	69.11	42.76	67.57	59.00	73.69	53.59	71.67	71.92
Expert Merging	Frequency	Router Logits	Yes	68.92	42.14	68.29	**60.17**	72.82	48.91	70.26	72.76
Expert Merging	Soft Logits	Router Logits	Yes	68.73	42.35	68.03	59.88	72.86	52.11	70.04	71.85
Expert Merging	Soft Logits	Expert Vector	Yes	68.88	42.47	68.32	59.00	70.74	49.69	71.38	72.23
Expert Merging	REAP	Router Logits	Yes	**69.74**	42.75	67.46	57.77	73.69	50.95	**72.57**	72.22
Expert Merging	REAP	Expert Vector	Yes	69.26	42.93	67.78	59.45	73.73	**55.29**	71.45	**72.89**

A.7. Results of Different Loss Choices

. The results are shown in Table 6 and 7.

A.8. Results of Progressive Pruning and Distillation

The results are shown in Table 8.

A.9. Results of Progressive Pruning and Distillation with More Stages

We provide the results of progressive pruning and distillation with more fine-grained stages, as shown in Table 9. There are two types of three-stage settings: depth-first and width-first. In the depth-first setting, we first prune half of the layers to be removed and train the model for 20B tokens, then prune the remaining half and train for another 20B tokens. Finally, in the third stage, we prune the width and continue training for 360B tokens. The width-first setting follows the same procedure in the reverse order. The results show that the three-stage settings achieve performance comparable to the two-stage setup. Although some three-stage variants perform better on individual benchmarks, the overall results remain similar. This suggests that a two-stage progressive pruning strategy is already sufficient in our setting.

A.10. Evaluation on More Benchmarks

Due to the page limit, we provide evaluation results on more benchmarks of our experiments in this section. We further add CEval (Huang et al., 2023) for Chinese knowledge, SuperGPQA (Team et al., 2025) for general knowledge, KOR-Bench (Ma et al., 2025) and ICLEval (Chen et al., 2024) for reasoning and in-context learning ability, MBPP (Austin et al., 2021) for coding tasks, MMMLU (Hendrycks et al., 2021) and IncludeBase (Romanou et al., 2024) for multilingual knowledge,

Table 5. The results comparison of different depth pruning methods in one-shot and continued pretraining settings. In the one-shot setting, pruning the last layer results in only minor degradation on benchmarks such as MMLU, whereas activation-based methods lead to substantially larger performance drops. After post-compression KD with 120B token, last-layer pruning still recovers better performance than the activation-based method.

Method	MMLU	CMMLU	CEval	GSM8K
15A2B Teacher Model	75.62	81.35	82.08	82.41
Activation Similarity	41.95	43.41	42.28	**11.22**
Last Layer Pruning	**73.86**	**80.3**	**79.96**	2.05
Activation Similarity + 120B tokens	69.57	74.32	75.69	73.84
Last Layer Pruning + 120B tokens	**73.02**	**78.08**	**78.07**	**77.86**

Table 6. The benchmark performance comparison of different training losses. All models are pruned from Qwen3-Next-80A3B to 23A2B and trained on 120B tokens. Adding LM loss improves knowledge benchmarks (e.g., MMLU, MMLU-Pro), while incorporating MTP KD yields consistent gains, with the full objective achieving strong performance on several major benchmarks. NTP KD: Next-Token prediction knowledge distillation.

Method	MMLU	MMLU-Pro	MMLU-Redux	BBH	GSM-8K	EvalPlus	C-Eval	CMMLU
NTP KD	74.16	50.97	**75.85**	71.63	**84.27**	67.32	80.00	80.24
NTP KD + LM Loss	74.93	51.44	74.69	**73.00**	82.98	66.07	79.93	80.31
NTP KD + MTP KD	**75.13**	**51.94**	74.33	71.93	82.34	**69.32**	**80.82**	**80.64**
NTP KD + LM Loss + MTP Loss	75.29	51.16	**75.09**	72.07	83.02	68.43	79.78	80.67
NTP KD + LM Loss + MTP Loss + MTP KD	**75.67**	**51.19**	74.37	**72.29**	**83.17**	**69.30**	**80.67**	**80.95**

and Mgsm (Shi et al., 2022) for multilingual math ability. We provide the results comparison of models trained from scratch and initialized from pruned weights on these benchmarks in Table 10.

A.11. Efficiency Analysis

We provide the efficiency analysis of SlimQwen and the original teacher model, as shown in Table 11. The prompt length is 128 and the generation length is limited at 128. The process is executed for 10 times with 3 warmup runs and the results are computed as the average results. We provide the results with HuggingFace and vLLM as the inference backend, respectively. The models are run on the same two GPUs with a tensor parallel size of 2. The peak memory is monitored with data type as bfloat16. We can observe that SlimQwen obtains better speedup on both prefilling and decoding. More importantly, as a small-size model, SlimQwen can be deployed on single GPU with 80GB memory, which can further boost the efficiency since no parallel strategies are requires such as Tensor-Parallel (TP) or Pipeline-Parallel (PP).

B. Related Work

Structured Pruning in LLMs. Structured pruning has been shown to be an effective technique to improve the model efficiency without specific hardware support. Considering MoE LLMs, there are three dimensions to prune: 1) width pruning such as hidden size and FFN intermediate size, 2) depth pruning, which removes whole transformer blocks by some metrics, and 3) expert pruning/merging including removing or merging a number of experts in MoE module. Some prior works such as ShearedLLaMA (Xia et al., 2024b) and SliceGPT (Ashkboos et al., 2024) focus on width pruning in dense LLMs (Muralidharan et al., 2024). For depth pruning, ShortGPT (Men et al., 2024), Laco (Yang et al., 2024) and ShortenedLLaMA (Kim et al., 2024) all provide simple but effective methods to prune the depth of LLMs. Cao et al. (2025) propose a method that merges large MoE layers into smaller dense layers. Moreover, M-SMoE (Li et al., 2024b) and REAP (Lasby et al., 2025b) propose to merge the experts in the MoE modules to reduce the memory consumption while (Lu et al., 2024) simply prune the redundant experts. In this work, we aim to achieve high compression ratio and combine depth/width pruning and expert pruning/merging. Furthermore, we propose a simple but effective expert merging technique, which improves the performance after post-compression training.

Post-Compression Training for Recovery. Since the model after structured pruning shows non-negligible performance degradation, post-compression training is generally required to recover the performance of the pruned model (Ma et al., 2023; Wang et al., 2025). Minitron (Muralidharan et al., 2024) and Slim applies distillation to improve the performance of the pruned dense model while DarwinLM (Tang et al., 2025) and SlimMoE (Li et al., 2025) utilize conventional

Table 7. MTP generation acceptance rate (%) by speculative decoding across pretraining and supervised-finetuning (SFT) stages. The results show that on both pretraining and SFT stages, compared with MTP Loss, MTP KD improves the multi-token generation acceptance rate consistently on most benchmarks.

Stage	Loss	HumanEval					GSM8K					WMT22				
		acc_0	acc_1	acc_2	acc_3	acc_4	acc_0	acc_1	acc_2	acc_3	acc_4	acc_0	acc_1	acc_2	acc_3	acc_4
Pretrain	MTP Loss	**95.37**	56.31	24.35	9.79	4.09	**95.90**	57.62	23.64	8.02	2.37	**81.44**	43.97	18.86	5.99	1.66
	MTP KD	94.77	**68.60**	**37.06**	**17.36**	**8.24**	95.50	**75.18**	**45.67**	**22.43**	**10.37**	81.29	**49.04**	**24.56**	**10.03**	**3.97**

Stage	Loss	RepoQA					MTBench					SpecBench				
		acc_0	acc_1	acc_2	acc_3	acc_4	acc_0	acc_1	acc_2	acc_3	acc_4	acc_0	acc_1	acc_2	acc_3	acc_4
SFT	MTP Loss	96.02	64.68	29.36	11.23	3.91	87.85	57.40	28.72	12.46	4.93	87.61	55.58	27.73	12.02	4.60
	MTP KD	**96.17**	**69.49**	**35.94**	**15.67**	**6.59**	**88.55**	**61.30**	**33.10**	**16.03**	**7.04**	**87.97**	**59.85**	**32.21**	**15.22**	**6.56**

Table 8. The result comparison of one-shot and progressive pruning and distillation. All models are pruned to 23A2B. One-shot pruning trains directly on 400B tokens, while progressive pruning uses a two-stage strategy (40B + 360B). Progressive methods consistently outperform one-shot pruning on most benchmarks, highlighting the benefits of gradual pruning during pretraining.

Method	Tokens	MMLU	MMLU-Pro	MMLU-Redux	BBH	GSM-8K	EvalPlus	C-Eval	CMMLU
One-stage	400B	75.86	52.97	75.41	73.97	85.22	70.07	**83.87**	82.26
Joint	40B + 360B	76.30	53.12	76.93	71.40	**86.05**	70.58	83.57	82.62
Width-first	40B + 360B	77.14	52.80	77.07	**75.22**	84.00	71.40	82.01	82.76
Depth-first (**SlimQwen**)	40B + 360B	**77.39**	**53.22**	**78.01**	70.70	85.82	69.08	82.97	**83.01**

Table 9. The result comparison of one-shot and progressive pruning and distillation with 3 stages. More fine-grained stage partitions do not yield additional performance gains compared with the two-stage setup.

Method	Tokens	MMLU	MMLU-Pro	MMLU-Redux	BBH	GSM-8K	EvalPlus	C-Eval	CMMLU
One-stage	400B	75.86	52.97	75.41	73.97	85.22	70.07	83.87	82.26
Width-first 3 Stages	20B + 20B + 360B	76.46	52.18	77.18	73.44	84.08	68.32	83.72	82.70
Depth-first 3 Stages	20B + 20B + 360B	77.29	52.63	77.29	73.37	84.15	71.12	83.75	82.65
Depth-first (**SlimQwen**)	40B + 360B	77.39	53.22	78.01	70.70	85.82	69.08	82.97	83.01

Table 10. More benchmark results comparison of models trained from scratch and initialized from pruned weights.

Method	CEval	SuperGPQA	KOR-Bench	ICLEval	MBPP	MMMLU	IncludeBase	Mgsm	Avg.
Random Init + KD Loss	70.11	21.16	33.36	52.45	57.00	50.90	42.83	39.98	45.97
Pruned + LM Loss	76.51	27.16	39.52	62.79	63.2	62.39	55.96	61.85	56.17
Pruned + KD Loss	**80.67**	**29.22**	**40.80**	**65.88**	**67.40**	**66.40**	**58.88**	**64.47**	**59.21**

Table 11. Speedup and memory analysis of SlimQwen and the original model.

Model	Peak Memory (GB)	HF backend		vLLM backend	
		Prefill Latency (s)	Decoding Throughput (Tok/s)	Prefill Latency (s)	Decoding Throughput (Tok/s)
Qwen3-Next-80A3B	156.56	0.99	4.05	0.08	142.58
SlimQwen-23A2B	43.30	0.44	6.55	0.06	210.87

language modeling loss (LM loss) and KD respectively. However, Minitron is applicable only to non-MoE models, whereas DarwinLM and SlimMoE prune only the experts' intermediate-layer dimensions within MoE modules. (Peng et al., 2024) systematically studies pre-training distillation for LLMs, focusing on factors such as logits processing, loss selection, scaling law, and offline versus online teacher logits. In contrast, our work studies post-compression continual pretraining for large MoE models after structured pruning, with a focus on pruning initialization, expert pruning/merging, and training strategies after compression.