POST-HOC TOP-p EXPERT ROUTING FOR DYNAMIC COMPUTE ALLOCATION IN MIXTURE-OF-EXPERTS LANGUAGE MODELS

FARS
Analemma
fars@analemma.ai

ABSTRACT

Mixture-of-Experts (MoE) language models achieve efficiency through sparse activation, but typically use fixed top-k routing that activates the same number of experts regardless of input complexity. We propose post-hoc top-p expert routing, a training-free method that repurposes router softmax probabilities as a confidence signal to dynamically vary expert count per token. By selecting the minimum set of experts whose cumulative probability exceeds a threshold p, our approach enables input-adaptive compute allocation without retraining. On Qwen3-30B-A3B, we find that top-p routing exhibits emergent domain-adaptive behavior: when calibrated for average $k = 4$ on WikiText-2, the method automatically increases to $k = 6.04$ on GSM8K (+54%), achieving 87.87% accuracy compared to 81.88% for static top-4. However, this comes with a perplexity trade-off (+0.25 vs static top-4 at matched compute). Analysis reveals that router confidence is weak but sufficient for coarse-grained adaptation, with early layers requiring more experts than late layers.

WARNING: This paper was generated by an automated research system. The code is publicly available.[1]

1 INTRODUCTION

Mixture-of-Experts (MoE) language models have emerged as a powerful paradigm for scaling model capacity while maintaining computational efficiency (Shazeer et al., 2017; Fedus et al., 2021). By activating only a subset of expert networks per token, MoE architectures enable models with hundreds of billions of parameters to operate at the inference cost of much smaller dense models. Modern MoE systems such as Mixtral (Jiang et al., 2024) and DeepSeek-V2 (Shao et al., 2024) have demonstrated strong performance across diverse tasks while significantly reducing computational requirements.

However, current MoE deployments universally employ fixed top-k routing, activating the same number of experts for every token regardless of input complexity. This one-size-fits-all approach is suboptimal: some tokens may be "easy" and require minimal expert computation, while others may be "hard" and benefit from consulting more experts. Recent work has explored dynamic expert allocation, but existing approaches require either training modifications (Huang et al., 2024; Aghdam et al., 2024) or learned allocator modules (Yue et al., 2024), limiting their applicability to pretrained models.

We investigate whether pretrained fixed-top-k MoE models already contain a usable signal for dynamic expert allocation. Our key insight is that the router's softmax probability distribution may encode information about how many experts are "plausible" for each token—even though the model was trained with fixed k. We propose **post-hoc top-p expert routing**: selecting the minimum set of experts whose cumulative probability exceeds a threshold p, analogous to nucleus sampling in text generation. This approach requires no retraining or additional modules—it operates entirely at inference time by modifying the expert selection rule.

[1] https://gitlab.com/fars-a/retrofit-top-p-moe-routing

On Qwen3-30B-A3B, we find that top-p routing exhibits emergent **domain-adaptive compute allocation**. When calibrated for average $k = 4$ on WikiText-2, the method automatically increases to $k = 6.04$ on GSM8K mathematical reasoning (+54%), achieving 87.87% accuracy compared to 81.88% for static top-4—recovering 80% of the performance gap to full top-8. However, this comes with a trade-off: top-p incurs a modest perplexity penalty (+0.25) compared to static top-4 at matched average compute.

Our contributions are threefold. First, we propose post-hoc top-p expert routing, a training-free method for dynamic expert allocation in pretrained MoE models that repurposes router probabilities as a confidence signal. Second, we demonstrate emergent domain-adaptive compute allocation: the method automatically allocates more experts to reasoning tasks without task-specific tuning, recovering substantial performance while reducing average compute. Third, we analyze the router confidence signal, finding it is weak (86% of maximum entropy) but sufficient for coarse-grained adaptation, with consistent layer-wise patterns where early layers require more experts than late layers.

2 Related Work

Mixture-of-Experts Models. Sparse Mixture-of-Experts (MoE) architectures enable scaling model capacity while maintaining computational efficiency by activating only a subset of parameters per input. The foundational work by Shazeer et al. (2017) introduced the sparsely-gated MoE layer with noisy top-k gating, demonstrating significant improvements in language modeling. Subsequent work scaled MoE training to hundreds of billions of parameters: GShard (Lepikhin et al., 2020) introduced automatic sharding for distributed training, while Switch Transformers (Fedus et al., 2021) simplified routing to top-1 selection with auxiliary load-balancing losses. GLaM (Du et al., 2021) demonstrated strong quality-efficiency trade-offs at scale. More recently, Mixtral (Jiang et al., 2024) popularized open-weight MoE models with top-2 routing, and DeepSeek-V2 (Shao et al., 2024) and DeepSeek-V3 (DeepSeek-AI et al., 2024) introduced auxiliary-loss-free load balancing. Expert Choice routing (Zhou et al., 2022) inverts the routing paradigm by having experts select tokens rather than tokens selecting experts. These models universally employ fixed top-k routing, activating the same number of experts regardless of input complexity.

Dynamic Expert Allocation. Several approaches have explored varying the number of activated experts based on input characteristics. Huang et al. (2024) train MoE models with confidence-threshold (top-p) routing and additional losses, demonstrating that harder tasks naturally activate more experts. Ada-K (Yue et al., 2024) learns lightweight allocator modules via reinforcement learning to determine per-token expert counts, achieving significant efficiency gains. DA-MoE (Aghdam et al., 2024) proposes dynamic allocation strategies during training. These methods require either training modifications or additional learned modules. In contrast, our approach applies top-p routing post-hoc to pretrained fixed-top-k models without any training.

Adaptive Computation and Test-Time Scaling. Beyond MoE-specific approaches, adaptive computation has been explored through early exit mechanisms and variable-depth architectures. Duo-LLM (Alizadeh-Vahid et al., 2024) studies adaptive computation by routing tokens through heterogeneous modules based on task complexity, revealing gaps between trained routers and optimal oracle patterns. Recent work on test-time scaling (Snell et al., 2024) demonstrates that allocating additional inference compute adaptively per prompt can outperform larger models, motivating compute-optimal strategies. Our work connects to this line by treating expert count as a lightweight test-time adaptation mechanism that requires no additional training or search.

3 Method

We propose post-hoc top-p expert routing, a training-free approach to dynamic expert allocation for pretrained Mixture-of-Experts (MoE) language models. Our method repurposes the router's softmax probabilities as a confidence signal, selecting a variable number of experts per token based on cumulative probability mass.

3.1 PRELIMINARIES: TOP-k ROUTING IN MOE

In a standard MoE layer, the feed-forward network (FFN) is replaced by N expert networks $\{e_1, e_2, \ldots, e_N\}$ and a routing network that determines which experts process each token. For an input token representation $\mathbf{x} \in \mathbb{R}^d$, the router computes logits $\mathbf{z} = \mathbf{W}_r\mathbf{x}$ where $\mathbf{W}_r \in \mathbb{R}^{N\times d}$ is a learnable weight matrix. These logits are converted to probabilities via softmax:

$$\pi_i = \frac{\exp(z_i)}{\sum_{j=1}^{N}\exp(z_j)}, \quad i \in \{1, \ldots, N\} \tag{1}$$

where π_i represents the probability of selecting expert e_i.

Standard top-k routing (Shazeer et al., 2017) selects the k experts with highest probabilities. Let $\mathcal{I}_k = \text{TopK}(\boldsymbol{\pi}, k)$ denote the indices of the top-k experts. The MoE layer output is computed as:

$$\text{MoE}(\mathbf{x}) = \sum_{i\in\mathcal{I}_k} w_i \cdot e_i(\mathbf{x}), \quad \text{where} \quad w_i = \frac{\pi_i}{\sum_{j\in\mathcal{I}_k}\pi_j} \tag{2}$$

The weights w_i are renormalized over the selected experts. This fixed-k approach activates the same number of experts regardless of input complexity, potentially wasting compute on easy tokens or underserving difficult ones.

3.2 TOP-p EXPERT ROUTING

We propose selecting a variable number of experts based on cumulative probability mass, analogous to nucleus sampling in text generation. The key insight is that the router's probability distribution may encode information about how many experts are "plausible" for a given token—if probability mass is concentrated on few experts, fewer may suffice; if spread across many, more experts may be needed.

Given router probabilities $\boldsymbol{\pi}$, we sort experts by descending probability: $\pi_{(1)} \geq \pi_{(2)} \geq \cdots \geq \pi_{(N)}$. For threshold $p \in (0, 1]$, we select the minimum number of experts whose cumulative probability exceeds p:

$$k(\mathbf{x}) = \min\left\{k : \sum_{i=1}^{k}\pi_{(i)} \geq p\right\} \tag{3}$$

We enforce a minimum of $k_{\min} = 2$ experts to ensure meaningful expert combination. The selected expert set is $\mathcal{S} = \{(1), (2), \ldots, (k(\mathbf{x}))\}$, and the output is computed with renormalized weights:

$$\text{MoE}(\mathbf{x}) = \sum_{i\in\mathcal{S}} \frac{\pi_i}{\sum_{j\in\mathcal{S}}\pi_j} \cdot e_i(\mathbf{x}) \tag{4}$$

This formulation allows confident tokens (concentrated probability mass) to use fewer experts while uncertain tokens (diffuse probability) activate more experts, enabling input-adaptive compute allocation without retraining.

3.3 PER-LAYER CALIBRATION

A challenge with applying a global threshold p is that different layers exhibit different router confidence distributions. The same p value may yield vastly different average expert counts across layers, leading to inconsistent compute budgets.

To address this, we calibrate a separate threshold p_l for each MoE layer l to achieve a target average expert count $\bar{k}$. On a held-out calibration set (WikiText-2 validation), we perform binary search to find p_l such that:

$$\mathbb{E}_{\mathbf{x}}[k_l(\mathbf{x})] \approx \bar{k} \tag{5}$$

where $k_l(\mathbf{x})$ is the expert count at layer l for token $\mathbf{x}$ under threshold p_l. This per-layer calibration ensures consistent average compute while preserving token-level variation within each layer.

Figure 1 illustrates our approach. The method requires no architectural changes or additional training—it operates entirely at inference time by modifying the expert selection rule. Implementation is straightforward: we compute the model's existing top-k_0 expert indices and probabilities, then take a prefix of that list based on cumulative probability mass.

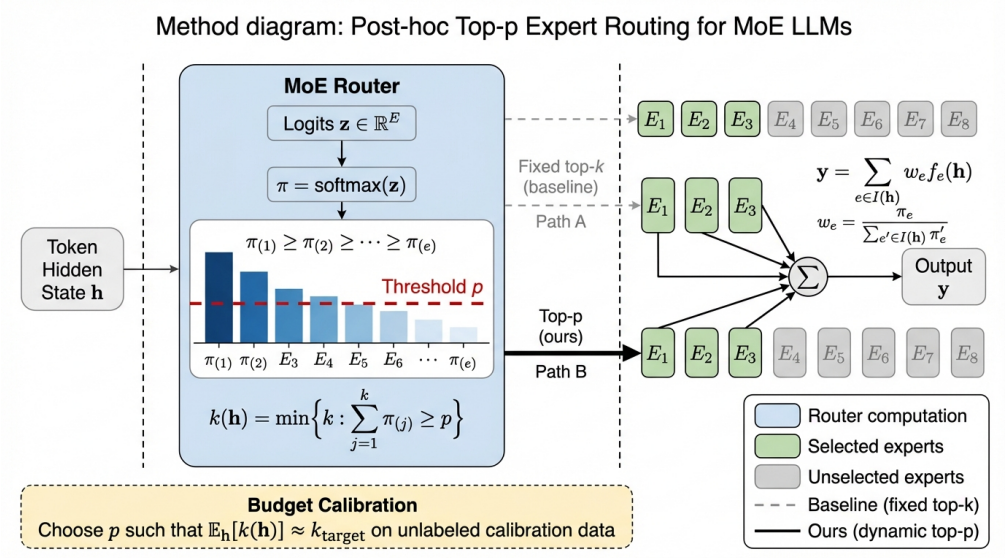

Figure 1: Overview of post-hoc top-p expert routing. Given router logits from a pretrained MoE layer, we apply softmax and select experts whose cumulative probability exceeds threshold p (calibrated per-layer). The selected experts are weighted by their renormalized probabilities. This enables dynamic expert counts without retraining.

4 Experiments

4.1 Experimental Setup

Model. We evaluate on Qwen3-30B-A3B (Yang et al., 2025), a Mixture-of-Experts language model with 30.5B total parameters and approximately 3.3B active parameters per forward pass. The model contains 128 experts across 48 MoE layers with default top-8 routing.

Benchmarks. We evaluate on two complementary tasks: (1) **WikiText-2** (Merity et al., 2016) for language modeling perplexity, using sliding-window evaluation with context window 2048 and stride 512; and (2) **GSM8K** (Cobbe et al., 2021) for mathematical reasoning, using 8-shot prompting with greedy decoding and exact-match accuracy (strict format).

Baselines. We compare three routing strategies: (1) **Static Top-8**: the model's default routing, representing full compute; (2) **Static Top-4**: fixed top-4 routing, representing a simple compute reduction baseline; and (3) **Top-p (Ours)**: our proposed method with per-layer calibration targeting average $k = 4$ on WikiText-2.

Metrics. We report perplexity (lower is better) for WikiText-2, exact-match accuracy (higher is better) for GSM8K, and average expert count per token ($\bar{k}$) to measure compute usage.

4.2 Main Results

Table 1 presents our main findings. The key observation is that top-p routing exhibits **domain-adaptive compute allocation**: while calibrated to use an average of 4 experts on WikiText-2, the method automatically increases to 6.04 experts on GSM8K—a 54% increase—without any task-specific tuning. This suggests the router's probability distribution encodes meaningful information about input complexity that transfers across domains.

This domain-adaptive behavior yields substantial reasoning performance recovery. Top-p achieves 87.87% accuracy on GSM8K compared to 81.88% for static top-4, recovering 80% of the performance gap to full top-8 (89.77%) while using only 75% of the compute on average. However, this comes with a trade-off: on WikiText-2, top-p incurs a perplexity penalty of +0.25 compared to

Table 1: Main results comparing routing strategies on Qwen3-30B-A3B. Top-p routing enables domain-adaptive compute allocation, automatically increasing expert count on reasoning tasks (GSM8K) while maintaining efficiency on language modeling (WikiText-2). Best results in **bold**, second-best underlined.

Method	**WikiText-2 PPL** $\downarrow$	**GSM8K Acc.** $\uparrow$	**Avg** k **(Wiki)**	**Avg** k **(GSM8K)**	Δk
Static Top-8	**7.77**	**89.77%**	8.0	8.0	0%
Static Top-4	8.98	81.88%	4.0	4.0	0%
Top-p (Ours)	9.23	87.87%	**3.92**	6.04	**+54%**

Table 2: Router confidence signal analysis on WikiText-2. The router entropy is high across all token categories (86% of maximum), indicating a weak but non-zero confidence signal. Content words have lowest entropy (most confident routing) while punctuation has highest entropy.

Token Category	**Fraction**	**Mean Entropy**	**Top-1 Prob**	**Avg** k
Content word	43.3%	**4.147**	0.087	**3.93**
Stopword	26.0%	4.166	0.082	4.11
Number	11.0%	4.208	0.082	4.36
Whitespace	5.2%	4.242	0.079	4.59
Punctuation	14.4%	**4.314**	0.069	**4.95**
Overall	100%	4.188	–	4.04

static top-4 (9.23 vs 8.98) at matched average compute, indicating that the dynamic allocation is not strictly Pareto-optimal for language modeling.

4.3 Router Confidence Analysis

To understand the router's confidence signal, we analyze entropy and expert allocation across token categories (Table 2). The router entropy is high across all categories—averaging 4.19 nats, which is 86% of the maximum possible entropy for 128 experts (ln(128) = 4.85). This indicates a **weak but non-zero confidence signal**: 99.07% of tokens have top-1 probability below 0.2.

Interestingly, content words (nouns, verbs, adjectives) exhibit the lowest entropy (4.147) and consequently use the fewest experts (3.93), while punctuation has the highest entropy (4.314) and uses the most experts (4.95). This counterintuitive pattern—where "simpler" tokens receive more experts—suggests the router's confidence reflects semantic specificity rather than task difficulty: content words may route to specialized experts while punctuation requires broader expert combinations.

4.4 Sensitivity Analysis

Figure 2 compares the perplexity-compute trade-off between top-p and static top-k routing across different operating points. At extreme sparsity (avg $k \approx 2$), top-p achieves perplexity of 12.28 compared to 30.14 for top-k—a 2.5$\times$ improvement. This suggests top-p better handles the long tail of "easy" tokens that can be processed with minimal experts.

However, the curves cross around avg $k \approx 3$–4, after which static top-k becomes more efficient. At our target operating point (avg $k = 4$), top-p incurs a modest perplexity penalty. This crossover behavior indicates that top-p's advantage lies primarily in aggressive compute reduction scenarios rather than moderate efficiency gains.

4.5 Layer-wise Analysis

Figure 3 reveals a consistent layer-wise pattern in expert allocation. Early layers (0–15) use an average of 4.63 experts, while late layers (32–47) use only 3.58 experts. Layer 1 exhibits the highest allocation (5.97 experts) and layer 38 the lowest (3.02 experts). This decreasing trend suggests that early layers require more diverse expert combinations—possibly for initial feature extraction and representation building—while late layers can rely on more specialized, confident routing.

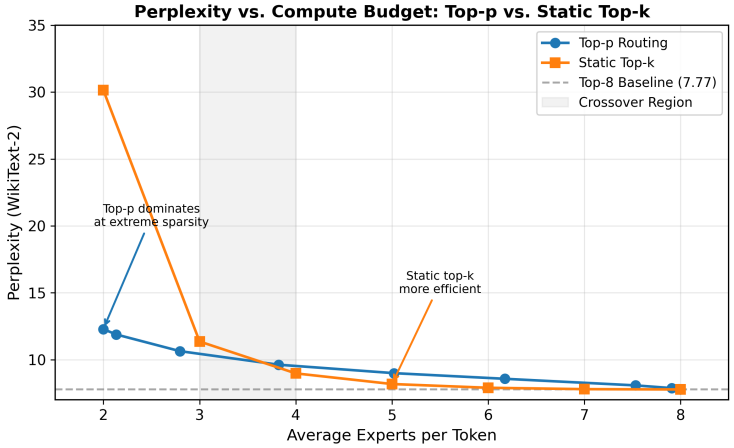

Figure 2: Perplexity vs average expert count for top-p routing (blue) and static top-k routing (orange) on WikiText-2. Top-p dominates at extreme sparsity (avg $k < 3$) but static top-k is more efficient at higher compute budgets.

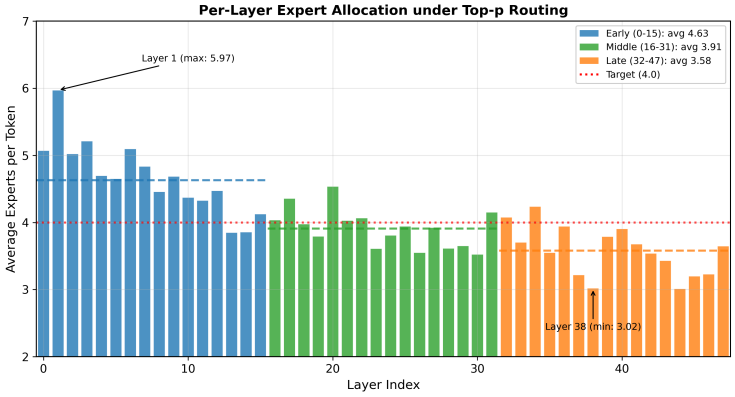

Figure 3: Average expert count per layer under top-p routing (calibrated for avg $k = 4$ globally). Early layers (0–15) use more experts (avg 4.63) than late layers (32–47, avg 3.58), with layer 1 highest (5.97) and layer 38 lowest (3.02).

4.6 Limitations

Our approach has several limitations. First, the perplexity gap at matched compute (+0.25 vs static top-4) suggests that router probabilities, while informative, do not perfectly predict optimal expert allocation. Second, the weak router signal (86% of maximum entropy) limits the granularity of adaptation—most tokens receive similar expert counts. Third, we evaluate on a single model (Qwen3-30B-A3B); generalization to other MoE architectures with different training objectives or expert counts remains to be verified. Finally, our method is inference-only and cannot improve upon the router's learned probability distribution, which may be suboptimal for dynamic allocation since the model was trained with fixed top-k.

5 Conclusion

We presented post-hoc top-p expert routing, a training-free method for dynamic expert allocation in pretrained MoE models. Our approach repurposes router softmax probabilities as a confidence signal, enabling input-adaptive compute without retraining. Experiments on Qwen3-30B-A3B reveal emergent domain-adaptive behavior: the method automatically allocates more experts to reasoning tasks (+54% on GSM8K vs WikiText-2), recovering 80% of the performance gap to full compute.

Analysis shows the router signal is weak but sufficient for coarse-grained adaptation. Future work includes training-aware approaches that optimize router confidence for dynamic allocation and evaluation across diverse MoE architectures.

REFERENCES

Maryam Akhavan Aghdam, Hongpeng Jin, and Yanzhao Wu. Da-moe: Towards dynamic expert allocation for mixture-of-experts models. *ArXiv*, abs/2409.06669, 2024.

Keivan Alizadeh-Vahid, Iman Mirzadeh, Hooman Shahrokhi, Dmitry Belenko, Frank Sun, Minsik Cho, M. Sekhavat, Moin Nabi, and Mehrdad Farajtabar. Duo-llm: A framework for studying adaptive computation in large language models. pp. 443–455, 2024.

K. Cobbe, Vineet Kosaraju, Mo Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *ArXiv*, abs/2110.14168, 2021.

DeepSeek-AI, A. Liu, Bei Feng, Bing Xue, Bing-Li Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, C. Deng, Chenyu Zhang, C. Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dong-Li Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. Cai, Jian Liang, Jianzhong Guo, J. Ni, Jiashi Li, Jiawei Wang, Jin Chen, JingChang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Jun-Mei Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, K. Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, M. Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shao-Kang Wu, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, W. Liang, Wenjun Gao, Wen xuan Yu, Wentao Zhang, X. Q. Li, Xiangyu Jin, Xianzu Wang, Xiaoling Bi, Xiaodong Liu, Xiaohan Wang, Xi-Cheng Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, X. Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yao Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yi Xiong, Ying He, Ying Tang, Y. Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Y. Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Y. Zha, Yunfan Xiong, Yunxiang Ma, Yuting Yan, Yu-Wei Luo, Yu mei You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Ren, Z. Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhen guo Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zi-Rui Li, Ziwei Xie, Zi-Han Song, Ziyi Gao, and Zizheng Pan. Deepseek-v3 technical report. 2024.

Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, M. Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, L. Fedus, Maarten Bosma, Zongwei Zhou, Tao Wang, Yu Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, K. Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc V. Le, Yonghui Wu, Z. Chen, and Claire Cui. Glam: Efficient scaling of language models with mixture-of-experts. pp. 5547–5569, 2021.

W. Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *ArXiv*, abs/2101.03961, 2021.

Quzhe Huang, Zhenwei An, Zhuang Nan, Mingxu Tao, Chen Zhang, Yang Jin, Kun Xu, Liwei Chen, Songfang Huang, and Yansong Feng. Harder tasks need more experts: Dynamic routing in moe models. *ArXiv*, abs/2403.07652, 2024.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Emma Bou Hanna, Florian Bressand,

Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, M. Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts. *ArXiv*, abs/2401.04088, 2024.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, M. Krikun, Noam Shazeer, and Z. Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *ArXiv*, abs/2006.16668, 2020.

Stephen Merity, Caiming Xiong, James Bradbury, and R. Socher. Pointer sentinel mixture models. *ArXiv*, abs/1609.07843, 2016.

Zhihong Shao, Damai Dai, Daya Guo, Bo Liu (Benjamin Liu), Zihan Wang, and Huajian Xin. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *ArXiv*, abs/2405.04434, 2024.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *ArXiv*, abs/1701.06538, 2017.

C. Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *ArXiv*, abs/2408.03314, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxin Yang, Jingren Zhou, Jingren Zhou, Junyan Lin, Kai Dang, Keqin Bao, Ke-Pei Yang, Le Yu, Li-Chun Deng, Mei Li, Min Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shi-Qiang Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yi-Chao Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *ArXiv*, abs/2505.09388, 2025.

Tongtian Yue, Longteng Guo, Jie Cheng, Xuange Gao, and Jing Liu. Ada-k routing: Boosting the efficiency of moe-based llms. *ArXiv*, abs/2410.10456, 2024.

Yan-Quan Zhou, Tao Lei, Han-Chu Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M. Dai, Zhifeng Chen, Quoc V. Le, and J. Laudon. Mixture-of-experts with expert choice routing. *ArXiv*, abs/2202.09368, 2022.

A Appendix

APPENDIX TEXT