CoSMoEs: Compact Sparse Mixture of Experts

Patrick Huber, Akshat Shrivastava, Ernie Chang, Chinnadhurai Sankar, Ahmed Aly and **Adithya Sagar**
Meta Reality Labs

Abstract

Sparse Mixture of Expert (MoE) models are widely used foundation architectures at large scale, yet remain under-explored at smaller sizes. In this work, we introduce Compact Sparse Mixture of Experts (CoSMoEs) for on-device inference, addressing three key challenges: Quality, Memory, and Latency. On the quality front, we conduct a fair evaluation (removing confounding factors) and show that MoE architectures outperform dense models at on-device scale. We further propose weight-decomposed experts, which improve MoE performance beyond the standard formulation. On the memory and latency front, we address the large parameter count of MoE models by improving expert offloading efficiency through a novel training-time loss, reducing inference latency for on-device deployment.

1 Introduction

Mixture of Experts (MoEs) have become a popular extension of the transformer architecture (Vaswani et al., 2023). The core idea is that each token in the input sequence is routed through a set of sub-networks, or "experts", whose outputs are combined via a gating mechanism that determines each expert's contribution.

While all experts are activated in the most general MoE formulation (Jacobs et al., 1991; Jordan and Jacobs, 1993), *sparse* Mixture of Expert models select only a subset of experts per token (Cai et al., 2024), as used in Qwen (Bai et al., 2023; Yang et al., 2024), OLMoE (Muennighoff et al., 2024), Mixtral (Jiang et al., 2024), and DeepSeek (DeepSeek-AI, 2024). These large-scale MoE models are optimized for highly parallelized server-side inference.

In contrast, this work focuses on small-scale MoEs for edge devices, which face a distinct set of challenges around Quality, Memory, and Latency.

Quality. Unlike prior work (e.g., Jiang et al. (2024)), we establish a fair comparison between MoEs and dense models by aligning on active parameters (FLOP-aligned, *FA*) and total parameters (parameter-aligned, *PA*). We further reduce confounding factors by normalizing training data, recipes, and architectures wherever possible. Our evaluation shows that MoE architectures improve average language modeling performance by over 2.3% across model sizes. Building on these results, we propose a novel MoE extension following the intuition of "expert specialization," yielding up to an additional 1.1% improvement.

Memory and Latency. As shown in Figure 1, models trained in server environments face additional constraints when deployed on edge devices. Despite these restrictions being largely architecture-independent, the high total parameter count of modern MoE models severely affect their ability to be deployed on the edge. While the sparsity property of MoE architectures can partially offset the high parameter count through expert offloading, this incurs a large (4-20×) inference latency increase (Xue et al., 2024). To relax this trade-off, we propose a "block-wise expert selection" loss that reduces expert offloads by 6× and improves inference latency by 50%.

2 CoSMoEs Models

2.1 Sparse Mixture-of-Experts

At the core of this work is the sparse Mixture-of-Expert (MoE) architecture[1], popularized by GShard (Lepikhin et al., 2020) and Switch Transformers (Fedus et al., 2022). While MoEs can be applied to different parts of the architecture, the most common approach replaces the dense feed-forward layer with a router component and multiple experts. By selecting a discrete subset of experts

[1]When referring to MoE throughout this paper, we assume sparsity.

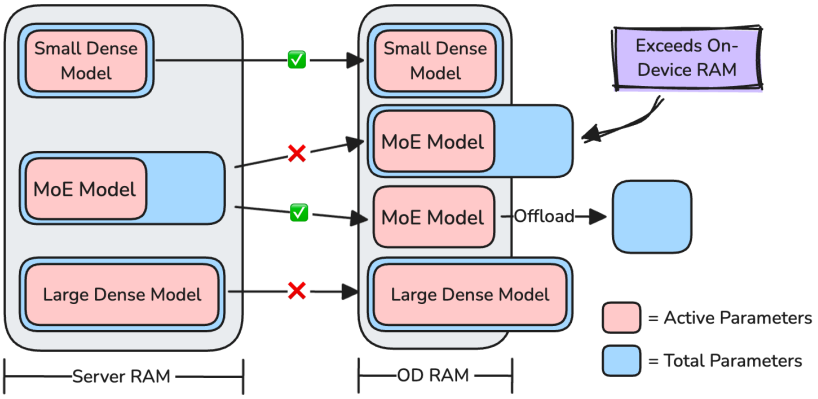

Figure 1: Server-side training environment (left) compared to the memory-constrained inference environment (right), showing deployment restrictions for parameter-heavy MoEs and large dense models on edge devices.

at each step, sparse MoE models are characterized by two quantities: their *active parameters* (FLOPs) and their *total parameters* (model size in memory). For expert selection, different routing paradigms have been proposed, either selecting experts per token (token choice) (Shazeer et al., 2017) or per expert (expert choice) (Zhou et al., 2022). We use token-choice routing, following the findings for text-only models in Muennighoff et al. (2024).

2.2 Weight-Decomposed Experts

Standard MoE models have a naturally large total parameter count because each expert maintains a full copy of the feed-forward layer weights. To reduce this overhead, we propose a lightweight expert formulation using matrix weight decomposition ("WD"), inspired by Low-Rank ("LoRA") adapters (Hu et al., 2021). The key intuition is that each expert is intended to "specialize" on a subset of input tokens, so each expert need not have full representational capacity. We therefore replace the full expert matrices of shape $n \times m$ with their decomposition into two smaller matrices of shape $n \times r$ and $r \times m$:

$$M_{n\times m} \approx L_{n\times r} \times R_{r\times m} \tag{1}$$

with $r \ll n$ and $r \ll m$ (see also Figure 2). In preliminary experiments, we find that setting r to half the hidden dimension yields the best trade-off between parameter reduction and model performance. To ensure a parameter-aligned comparison, we adjust the number of heads and layers for weight-decomposed models (prefixed with *WD*), as detailed in Section 3.1.

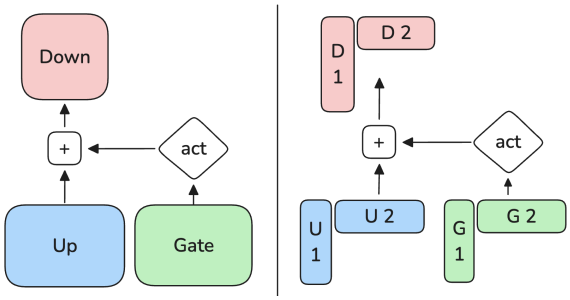

Figure 2: Feed Forward Layer: Standard (left) and Weight-Decomposed (right).

2.3 Block-wise Expert Selection

A central challenge for on-device MoE inference is expert offloading: when only active experts are kept in accelerator memory, each change in expert assignments requires transferring parameters between storage and compute, incurring substantial latency. Multiple lines of research have explored inference-time solutions such as predictive offloading and bitwidth adaptations (Yi et al., 2023; Eliseev and Mazur, 2023; Aminabadi et al., 2022). We approach this problem from a different angle: rather than optimizing the offloading mechanism, we reduce the number of offloading events by encouraging *temporally coherent* expert assignments during training.

To this end, we propose a "Block-wise Expert Selection" (BlES) loss term that penalizes frequent expert switches between consecutive tokens, closely related to the expert load balancing loss in Fedus et al. (2022):

Let R be a router logits tensor with shape (B, T, E), where B is the batch dimension, T is the sequence length, and E is the expert dimension. We compute routing weights W by applying

the softmax function to R, scaled by a temperature parameter τ:

$$W = \text{softmax}(\tau R) \tag{2}$$

In a non-differentiable step, we select the top-k experts K for each token based on W. Let S be the selected experts tensor with shape (B, T, K):

$$S = top\_k(W, K) \tag{3}$$

We then count the hard expert replacements H by comparing consecutive tokens' expert assignments:

$$\begin{aligned} H_e &= \sum_{b=1}^{B} \sum_{t=1}^{T-1} |(S_{[b,t+1]} = e) - (S_{[b,t]} = e)| \\ H &= \sum_{e=1}^{E} H_e \end{aligned} \tag{4}$$

where e is the expert index and $S_{[b,t]} = e$ equals 1 if expert e is among the top-k candidates for token t. We normalize H by the batch size, top-k, and number of tokens:

$$H_{norm} = \frac{\lfloor \frac{H}{2} \rfloor}{B \cdot K \cdot (T-1)} \tag{5}$$

Since H is non-differentiable, we introduce a continuous surrogate L that sums the per-expert probability differences between consecutive tokens:

$$\begin{aligned} L &= \sum_{b=1}^{B} \sum_{t=1}^{T-1} \sum_{e=1}^{E} |W_{b,t+1,e} - W_{b,t,e}| \\ L_{norm} &= \frac{L}{B \cdot T} \end{aligned} \tag{6}$$

The final loss is the product of the hard and soft expert selection terms:

$$loss = H_{norm} \cdot L_{norm} \tag{7}$$

To match the sequence-level computation of the BlES loss, we replace the standard (model-level) load balancing loss (Fedus et al., 2022) with **layer-wise load balancing** (following Lin et al. (2024)).

Layer-wise load balancing. An important design choice is to compute the load balancing loss *per layer* rather than aggregating across layers. When the loss is computed at the model level, it can be trivially minimized by consistently selecting a single expert per layer—for example, with 3 experts and 3 layers, always selecting expert 0 in layer 0, expert 1 in layer 1 and expert 2 in layer 2 achieves a perfect balance while also minimizing the BlES term. Figure 3 illustrates this failure mode with 3 layers and 3 experts.

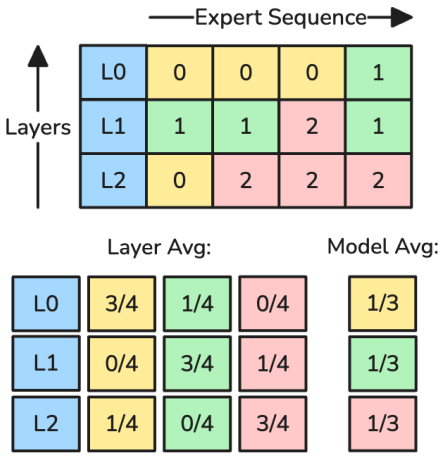

Figure 3: Example expert selection (for simplicity, k=1) for individual layers and the complete model.

3 Experimental Setup

3.1 Model Configuration

We compare two on-device size categories: "Phone-sized" (1–3B parameters) and "Wearable-sized" (100–300M parameters), across three architectures: Dense, MoE, and WD MoE, as described in Table 1[2]. All models are based on the Llama3 architecture, with an MoE component consisting of eight total experts, two of which are active per token. We follow the standard expert implementation from the Hugging Face codebase (Wolf et al., 2020). Model hyperparameters are kept consistent while aligning on active and total parameters. When trade-offs are necessary, we favor depth over breadth, following Liu et al. (2024).

3.2 Training Details

We pre-train all models on the FineWeb Education dataset (FW-edu, Penedo et al. (2024)), a 1.4 trillion token text corpus provided by Hugging Face (Wolf et al., 2020). FW-edu represents a high-quality, general-purpose language dataset.

3.3 Evaluation Metrics

We evaluate along three axes. For **language modeling performance**, we use the EleutherAI

[2]The BlES extension uses the MoE architecture and is therefore not listed separately.

Model	Params	L	H	Hid	Seq	Steps	Bsz
Phone-sized models, 1B-3B Parameters							
Dense	1.50B	16	32	2048	2048	310k	2048
MoE	1.37B (3.75B)	24	18	1440	2048	310k	2048
+ WD	1.42B (3.65B)	26	20	1600	2048	310k	2048
Dense	3.61B	28	24	3072	2048	310k	2048
Wearable-sized models, 100-300M Parameters							
Dense	189M	19	8	512	2048	310k	2048
MoE	188M (377M)	19	8	432	2048	310k	2048
+ WD	188M (377M)	32	10	400	2048	310k	2048
Dense	380M	29	12	768	2048	310k	2048

Table 1: On-device model candidates. Params = #Active (#Total) Parameters, L = Layers, H = Self-Attention Heads, Hid = Hidden size, Seq = Sequence length, Bsz = effective batch size.

LM eval harness with nine benchmarks (Gao et al., 2024), following the evaluation protocols of Llama3 (Grattafiori et al., 2024) and MobileLLM (Liu et al., 2024). For **offloading efficiency**, we report the Expert Replacement Ratio (ExRep), which measures the percentage of realized expert replacements, and the optimal expert balance, which computes the average per-layer deviation from a uniform distribution. For **memory and latency**, we report per-token generation speed and peak memory usage.

4 Results

4.1 Language Modeling Performance

Table 2 presents our language modeling results. A random baseline is shown at the top, followed by our MoE results at phone and wearable scale, with public baselines at the bottom for context.

Phone-sized models. All MoE candidates outperform the random baseline by a large margin and consistently improve over the FA dense model by at least 2%. For MMLU and AGI-English, all models show only minor gains, indicating room for further improvement. On the remaining benchmarks, clear improvements are observed. Among MoE variants, the weight-decomposed model performs best overall. We observe a minor performance regression when using the BlES loss. Compared to the PA dense model, MoE candidates perform better on 3 out of 10 metrics, falling only half a percentage point short on average. In the context of previously published models, our MoE candidates outperform the FA Llama 3.2 1B and OLMoE models but do not reach the PA Llama 3.2 3B performance.

Wearable-sized models. The wearable-sized evaluation shows similar trends. The weight-decomposed model again achieves the best MoE performance, here even surpassing the PA dense model. At wearable scale, at least one MoE model outperforms the PA dense model on 6 of 10 tasks. Compared to the published MobileLLM models, we observe improvements at both the 125M and 350M parameter scales. The BlES model again shows a slight performance drop relative to the standard MoE, consistent with the phone-scale findings.

4.2 Offload Efficiency

As illustrated in Figure 1, running MoE models on-device often requires offloading experts to stay within memory constraints, at the cost of significant latency increases. Since expert offloading frequency is data-dependent, we use a 100-sample subset of the C4 dataset (Raffel et al., 2020) as a proxy for general text data. Table 3 presents results along three dimensions: the expert replacement percentage (ExRep), the realized inference speed[3], and expert balancing quality (deviation from uniform expert balance, ΔUniform). The BlES extension achieves over 6$\times$ fewer expert switches than the standard MoE model. This directly translates to a 1.5$\times$ improvement in generation speed, with only a minor regression of less than 1% relative in expert balancing[4].

Qualitative analysis. Figure 4 visualizes expert assignments across 35 tokens for a single layer, comparing the BlES model (top) and the standard MoE model (bottom). The BlES model reduces expert replacements from 21 to 11 while preserving expert diversity—both models actively use 6 out of 8 experts. This illustrates how the BlES loss encourages temporally coherent expert assignments without collapsing to a single expert.

Layer-wise expert balance. To understand the per-layer impact of the BlES loss, Figure 5 plots the layer-wise expert balance for both models. With BlES, greater expert divergence is observed in lower layers, whereas the standard MoE model exhibits higher divergence in upper layers. Higher expert diversity in later layers appears preferable, given the general intuition that lower layers encode more local, syntactic information while upper layers capture more global, semantic structures.

[3]Full on-device benchmarks are presented in Section 4.3.

[4]Inference latency improvements are batch-size dependent.

Model	Params	MMLU	AGI-E	Arc-C	Arc-E	BoolQ	PIQA	SIQA	HellaS	OBQA	WinoG	Avg
Random Baseline												
Random	–	24.53	16.07	21.08	25.25	51.07	51.74	33.11	26.31	29.40	50.83	32.94
Phone-sized models, ∼1B-3B Parameters												
Dense	1.50B	24.78	17.99	36.95	74.03	59.08	74.54	41.76	59.88	41.20	57.54	48.78
MoE	1.37B (3.75B)	25.96	17.65	42.58	76.77	60.89	75.52	42.12	65.07	42.40	62.35	51.13
+ BlES	1.37B (3.75B)	25.40	17.50	41.55	77.02	62.81	76.06	41.91	63.14	42.60	59.04	50.70
+ WD	1.42B (3.65B)	23.90	**18.20**	43.69	76.81	**66.76**	76.39	**45.14**	66.51	42.80	62.04	52.22
Dense	3.61B	**26.41**	16.82	**44.54**	**77.9**	65.87	**77.48**	43.3	**67.18**	**45.00**	**63.46**	**52.80**
Wearable-sized models, ∼100-200M Parameters												
Dense	189M	22.9	16.82	23.29	56.82	57.09	64.15	37.82	36.36	32.8	50.99	39.90
MoE	188M (377M)	**25.27**	17.37	27.9	63.09	58.39	69.04	39.61	44.09	**34.4**	53.03	43.22
+ BlES	188M (377M)	24.27	17.58	24.83	58.84	**59.82**	66.49	38.64	39.70	33.40	49.96	41.35
+ WD	188M (377M)	23.64	17.16	28.58	62.58	57.13	**69.31**	**40.28**	46.15	33.20	**54.38**	**43.24**
Dense	380M	24.79	**17.86**	**28.92**	**64.35**	52.02	69.21	39.97	**46.53**	33.80	51.62	42.91
Public Baselines across Model Sizes												
MobLLM (2024)	135M	23.02	17.45	19.97	46.38	60.34	64.96	38.08	38.17	28.40	52.57	38.93
MobLLM (2024)	350M	26.33	17.47	23.89	56.4	61.96	68.88	39.87	49.57	31.00	57.38	43.28
Llama3.2 (2024)	1.4B	36.92	18.80	31.31	65.40	63.61	74.54	42.84	47.74	26.20	60.06	46.70
Llama3.2 (2024)	3.6B	54.01	22.53	42.32	74.41	72.81	76.71	47.13	55.32	31.20	69.30	54.50
OLMoE (2024)	1.68B (6.92B)	25.74	17.19	40.87	74.20	60.52	74.70	44.37	60.38	38.40	58.72	49.50

Table 2: Model comparison on zero-shot LM evaluations. Params = #Active (#Total) Parameters, BlES = Block-wise Expert Selection, WD = Weight-Decomposed, MobLLM = MobileLLM. Public baselines are evaluated using the EleutherAI LM eval harness (2024).

Model	ExRep (↓)	Tok/s Gen (↑)	ΔUni (↓)
MoE	43.82	15.02	**9.60**
+ BlES	**6.55**	**23.10**	9.67

Table 3: Impact of the BlES loss on expert replacement ratio (in percent), generation speed (tokens/second), and deviation from the uniform expert distribution (in percent). ↓ = lower is better, ↑ = higher is better.

4.3 On-Device Benchmarks

Since on-device models often execute in CPU-based environments or on proprietary accelerators, we compare model latency on both CPU and GPU[5]. As this paper targets training-time improvements, we use standard inference code from the Hugging Face Transformers library (Wolf et al., 2020) and the gpt-fast codebase (PyTorch Labs, 2023) without inference-specific optimizations such as EdgeMoE (Yi et al., 2023). Table 4 presents results along four dimensions: (1) language modeling performance (from Table 2), (2) inference speed across 128 tokens on CPU and GPU, (3) peak memory after 128 token generations, and (4) suitability for on-device deployment. In addition to the previously described model candidates, we evaluate standard MoE offloading as "Offl," where only active experts are kept on the accelerator.

Latency. On CPU, the FA dense model achieves the highest generation rate; MoE candidates are slightly slower, and the PA dense model is 2× slower. On GPU, MoE models generally produce fewer tokens per second than dense models, primarily due to their deeper architecture (Table 1). We also observe the 1.5× speed-up between standard offloaded MoE models and BlES offloaded models, consistent with Table 3. For context, inference-based offloading strategies such as those in Eliseev and Mazur (2023) and Aminabadi et al. (2022) achieve 2–3× and 5.5× latency reductions, respectively, at comparable model sizes. However, unlike our training-time approach, inference-time offloading methods often require additional modeling components to predict future expert usage, which can be impractical for on-device scenarios.

Peak Memory. Without expert offloading, the peak generation memory of MoE candidates is comparable to the PA dense model. With offloading, peak memory drops to the level of the FA dense model, since only active parameters are kept in memory. This makes offloaded MoE models viable on-device candidates (see ☑).

4.4 Ablation Studies

Going beyond the standard MoE setup with two active and eight total experts, we ablate these dimensions and explore their impact on quality, latency, and memory. We evaluate eight model configurations spanning a range of active and total expert counts. Figure 6 summarizes findings along the active expert (left) and total expert (right) dimensions. For active expert ablations, we fix the total number of experts to 8; for total expert ablations, we fix the number of active experts to 2.

[5]Evaluations are performed in a server environment; actual on-device accelerator numbers may vary.

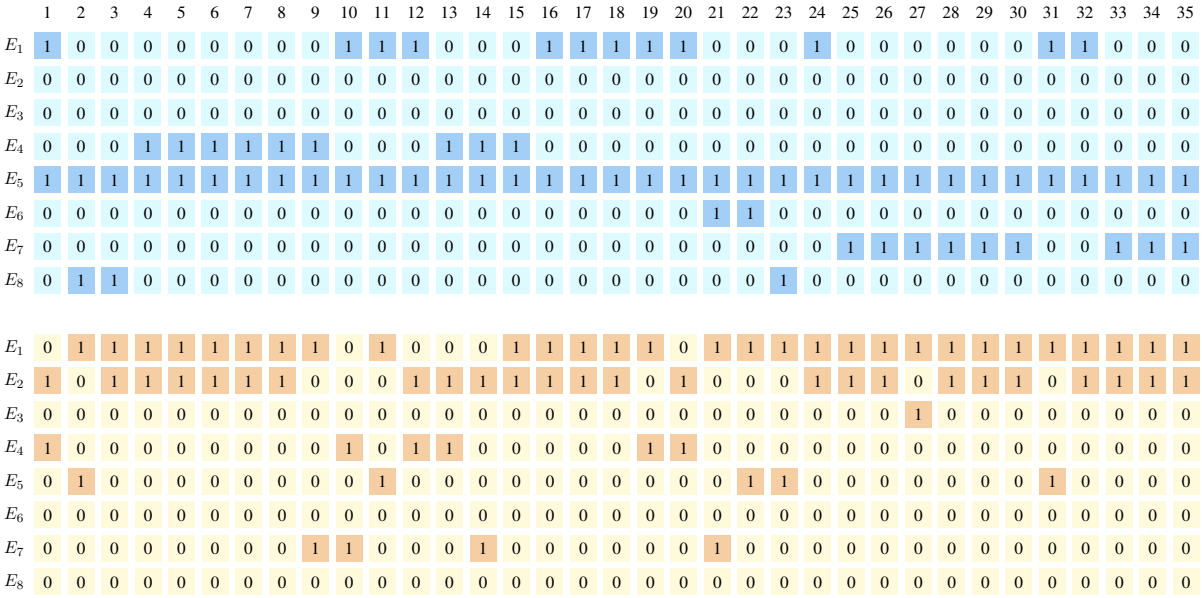

Figure 4: Expert assignments across 35 tokens for a single layer. 1 = Active, 0 = Inactive. Top: BlES (11 replacements), Bottom: Standard MoE (21 replacements).

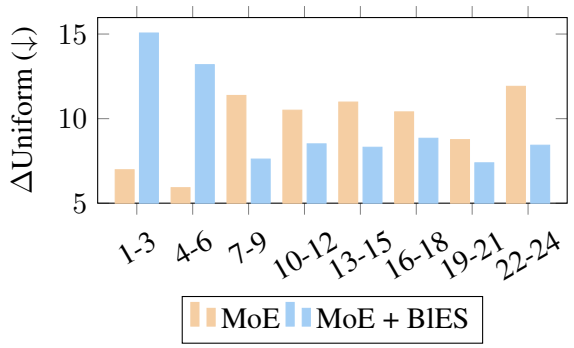

Figure 5: Per-layer divergence of expert routing from the uniform distribution. Large values indicate expert collapse toward a pseudo-dense layer.

Model	LM Eval	Latency		Mem	
Setup	Avg	Gen (tok/sec)		Gen	📱
Metric	%	CPU	GPU	GB	✅ / ❌
Dense	48.78	4.47	73.10	5.8	✅
MoE	51.13	4.30	40.60	14.7	❌
+ WD	52.22	3.85	33.50	14.2	❌
+ Offl	51.13	4.30	15.02	5.4	✅
+ BlES	50.70	4.30	23.10	5.4	✅
Dense	52.80	1.77	42.60	14.0	❌

Table 4: On-device benchmarks. Gen = Generation of 128 tokens (1 token prefill), Offl = Offloaded, BlES = Block-wise Expert Selection. Mem = Peak GPU memory. 📱 = Phone-sized, assuming <6GB of RAM use (e.g., iPhone 12 Pro).

Active Expert Ablation. Increasing the number of active experts improves model quality, though returns diminish as the count approaches 8. Generation speed decreases linearly with the number of active experts, while peak memory remains constant[6].

Total Expert Ablation. Model quality increases roughly linearly with the number of total experts, though the improvement is less pronounced than for active experts. Generation speed is unaffected since FLOPs are held constant. However, total expert count significantly impacts peak memory[7].

In summary, increasing either active or total experts improves model quality but requires a trade-off in either latency or memory.

4.5 Training Efficiency

Figure 7 compares the training dynamics of MoE and dense model candidates, aligned by dataset, training steps, and hyperparameters. We track average language modeling performance at checkpoints from 10k to the full 310k steps.

Comparing the FA and PA dense models with our best-performing MoE model, we corroborate the findings of Lin et al. (2024), observing a 5–10× training efficiency gain for MoE models over their FA dense counterparts. Our MoE candidate reaches the best performance of the 1.4B dense model at around 35k steps, while the larger 3.6B dense model achieves generally higher scores throughout training.

[6]Peak memory would increase if experts were actively offloaded.

[7]Peak memory would remain constant with active offloading, at the cost of reduced generation speed.

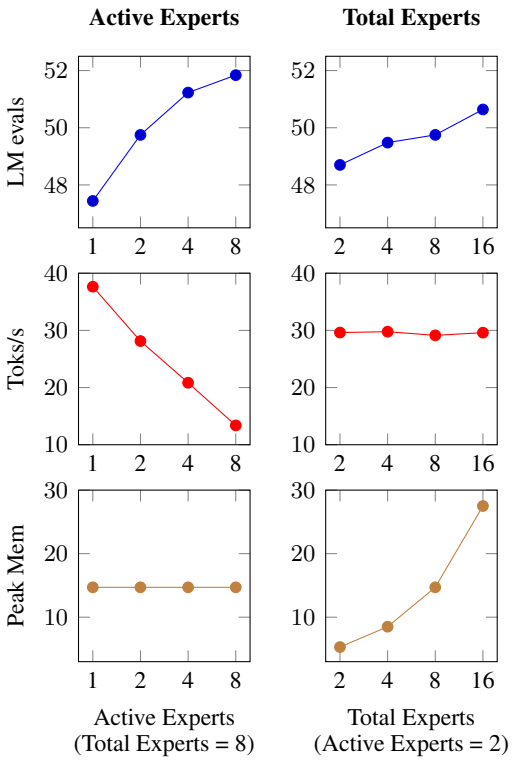

Figure 6: Active (left) and total (right) expert ablations of the 1.4B MoE model at 50k steps (∼210B tokens).

5 Related Work

Small-Scale Language Models. As foundation models become increasingly expensive to train and deploy, two research directions have emerged. The first focuses on improving small-scale architectures, including MobileLLM (Liu et al., 2024), MobiLlama (Thawakar et al., 2024), and the BabyLlama series (Timiryasov and Tastet, 2023; Tastet and Timiryasov, 2024). The second targets training data quality through cleaner, more curated pipelines, as demonstrated by the Microsoft Phi series (Abdin et al., 2024) and Hugging Face efforts (Ben Allal et al., 2024; Lozhkov et al., 2024). For a comprehensive survey, see Nguyen et al. (2024).

Sparse Mixture of Experts. Sparse MoEs span a wide range of model sizes: Qwen (2023; 2024) and OLMoE (2024) at 1–3B active parameters, Mixtral (2024) and DeepSeek (2024) at around 7B, and DBRX (Databricks, 2023) and Grok-1 (x.ai, 2023) at 36B and 86B active parameters, respectively. The OLMoE paper (Muennighoff et al., 2024) is particularly relevant, presenting training insights and design decisions for MoE models at smaller scales; we follow many of its findings. Comparing the inner workings of large MoE models, Lo et al. (2024) analyze Mixtral, Grok, and DeepSeek, finding initial similarities despite different training paradigms. We pursue similar comparisons but prioritize fairness by controlling confounding factors. For a detailed MoE survey, see Cai et al. (2024).

Weight Decomposition for Mixture of Experts. Along similar lines to our weight-decomposed experts, Dou et al. (2024) proposed a LoRA-style extension that converts dense networks into MoE models during supervised fine-tuning (SFT). By freezing the dense backbone and introducing a router at the SFT stage, their approach aims to reduce catastrophic forgetting of pre-training knowledge. In contrast, we apply weight decomposition directly during pre-training, training more parameter-efficient experts from the start.

Inference Efficiency. Our BlES loss is complementary to prior work on inference-time offloading optimization. Xue et al. (2024) improve expert prefetching and caching to reduce parameter transfers. EdgeMoE (Yi et al., 2023) enhances offloading through predictive strategies and bitwidth adaptations. Other frameworks include Mixtral Fast Inference (Eliseev and Mazur, 2023) and DeepSpeed Efficient Inference (Aminabadi et al., 2022). Unlike these approaches, our method reduces the number of offloading events during training rather than optimizing the offloading mechanism at inference time, making the two approaches orthogonal and potentially complementary.

6 Conclusion

We demonstrate how to enable sparse MoE architectures for on-device inference along three key dimensions: Quality, Memory, and Latency. On the quality front, a fair comparison shows that MoE models outperform their dense counterparts on language modeling tasks by over 2.35%. Our weight-decomposed experts yield further gains of up to 1.1% over standard MoE models. To make MoE models practical for on-device deployment, we address the offloading bottleneck by reducing expert switches during training. Our block-wise expert selection loss improves offloading efficiency by 6× and increases generation speed by 50% compared to standard offloaded MoE models. These results pave the way for deploying MoE architectures in on-device scenarios, supporting high-quality, privacy-preserving foundation models for edge devices.

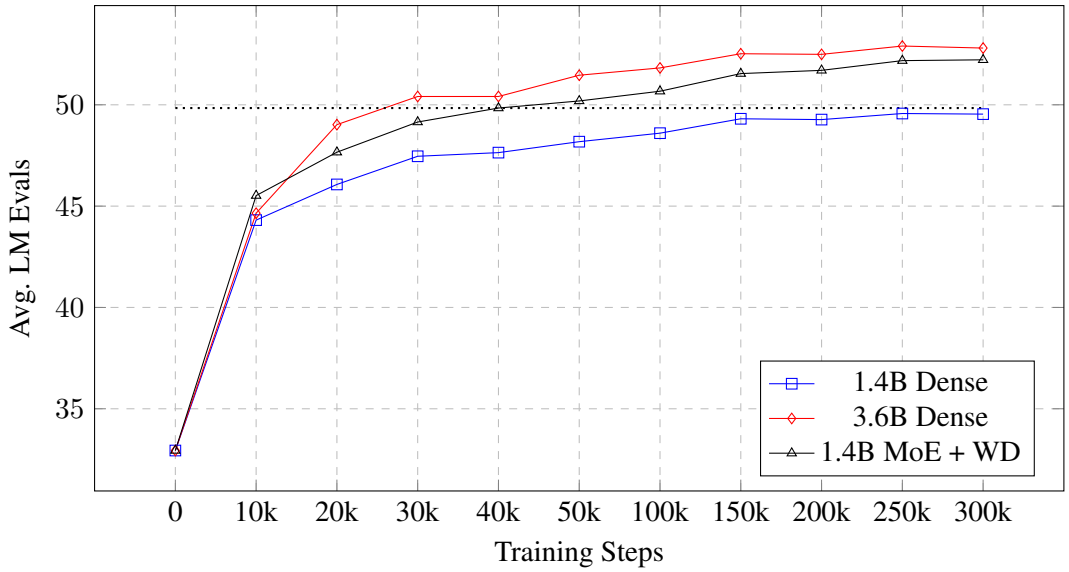

Figure 7: Training dynamics across model candidates. The dotted line marks the best 1.4B Dense checkpoint performance, reached by the MoE + WD model at ∼35k steps.

Limitations

We note two limitations of this work.

First, while we aim for a fair comparison between MoE and dense models, a perfectly balanced comparison remains elusive. Aligning active and total parameters necessarily introduces variation in model depth and width, which affects certain metrics (e.g., latency depends heavily on layer count). Additionally, keeping hyperparameters aligned may inadvertently favor one architecture. Thorough hyperparameter sweeps would help address this but would introduce their own confounding factors.

Second, our evaluations and benchmarks are performed on server hardware, which introduces a level of abstraction from actual edge devices. Although we aim to present a complete picture of MoE models for on-device use, this gap between server and edge environments is a limitation of our current evaluation.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Qin Cai, Vishrav Chaudhary, Dong Chen, Dongdong Chen, Weizhu Chen, Yen-Chun Chen, Yi-Ling Chen, Hao Cheng, Parul Chopra, Xiyang Dai, Matthew Dixon, Ronen Eldan, Victor Fragoso, Jianfeng Gao, Mei Gao, Min Gao, Amit Garg, Allie Del Giorno, Abhishek Goswami, Suriya Gunasekar, Emman Haider, Junheng Hao, Russell J. Hewett, Wenxiang Hu, Jamie Huynh, Dan Iter, Sam Ade Jacobs, Mojan Javaheripi, Xin Jin, Nikos Karampatziakis, Piero Kauffmann, Mahoud Khademi, Dongwoo Kim, Young Jin Kim, Lev Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi Li, Yunsheng Li, Chen Liang, Lars Liden, Xihui Lin, Zeqi Lin, Ce Liu, Liyuan Liu, Mengchen Liu, Weishung Liu, Xiaodong Liu, Chong Luo, Piyush Madan, Ali Mahmoudzadeh, David Majercak, Matt Mazzola, Caio César Teodoro Mendes, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin, Marko Radmilac, Liliang Ren, Gustavo de Rosa, Corby Rosset, Sambudha Roy, Olatunji Ruwase, Olli Saarikivi, Amin Saied, Adil Salim, Michael Santacroce, Shital Shah, Ning Shang, Hiteshi Sharma, Yelong Shen, Swadheen Shukla, Xia Song, Masahiro Tanaka, Andrea Tupini, Praneetha Vaddamanu, Chunyu Wang, Guanhua Wang, Lijuan Wang, Shuohang Wang, Xin Wang, Yu Wang, Rachel Ward, Wen Wen, Philipp Witte, Haiping Wu, Xiaoxia Wu, Michael Wyatt, Bin Xiao, Can Xu, Jiahang Xu, Weijian Xu, Jilong Xue, Sonali Yadav, Fan Yang, Jianwei Yang, Yifan Yang, Ziyi Yang, Donghan Yu, Lu Yuan, Chenruidong Zhang, Cyril Zhang, Jianwen Zhang, Li Lyna Zhang, Yi Zhang, Yue Zhang, Yunan Zhang, and Xiren Zhou. 2024. Phi-3 technical report: A highly capable language model locally on your phone.

Reza Yazdani Aminabadi, Samyam Rajbhandari, Minjia Zhang, Ammar Ahmad Awan, Cheng Li, Du Li, Elton Zheng, Jeff Rasley, Shaden Smith, Olatunji Ruwase, and Yuxiong He. 2022. Deepspeed inference: Enabling efficient inference of transformer models at unprecedented scale.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei

Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report.

Loubna Ben Allal, Anton Lozhkov, Guilherme Penedo, Thomas Wolf, and Leandro von Werra. 2024. Smollm-corpus.

Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. 2024. A survey on mixture of experts.

Databricks. 2023. Introducing dbrx: A new state-of-the-art open llm. Blog post.

DeepSeek-AI. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Jun Zhao, Wei Shen, Yuhao Zhou, Zhiheng Xi, Xiao Wang, Xiaoran Fan, Shiliang Pu, Jiang Zhu, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. Loramoe: Alleviate world knowledge forgetting in large language models via moe-style plugin.

Artyom Eliseev and Denis Mazur. 2023. Fast inference of mixture-of-experts language models with offloading.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2024. A framework for few-shot language model evaluation.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand,

Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. 2024. The llama 3 herd of models.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. 1991. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87.

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, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts.

M.I. Jordan and R.A. Jacobs. 1993. Hierarchical mixtures of experts and the em algorithm. In *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)*, volume 2, pages 1339–1344 vol.2.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020.

Gshard: Scaling giant models with conditional computation and automatic sharding.

Xi Victoria Lin, Akshat Shrivastava, Liang Luo, Srinivasan Iyer, Mike Lewis, Gargi Ghosh, Luke Zettlemoyer, and Armen Aghajanyan. 2024. Moma: Efficient early-fusion pre-training with mixture of modality-aware experts.

Zechun Liu, Changsheng Zhao, Forrest Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, Liangzhen Lai, and Vikas Chandra. 2024. Mobilellm: Optimizing sub-billion parameter language models for on-device use cases.

Ka Man Lo, Zeyu Huang, Zihan Qiu, Zili Wang, and Jie Fu. 2024. A closer look into mixture-of-experts in large language models.

Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. 2024. Fineweb-edu: the finest collection of educational content.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, Akshita Bhagia, Dustin Schwenk, David Wadden, Alexander Wettig, Binyuan Hui, Tim Dettmers, Douwe Kiela, Ali Farhadi, Noah A. Smith, Pang Wei Koh, Amanpreet Singh, and Hannaneh Hajishirzi. 2024. Olmoe: Open mixture-of-experts language models.

Chien Van Nguyen, Xuan Shen, Ryan Aponte, Yu Xia, Samyadeep Basu, Zhengmian Hu, Jian Chen, Mihir Parmar, Sasidhar Kunapuli, Joe Barrow, Junda Wu, Ashish Singh, Yu Wang, Jiuxiang Gu, Franck Dernoncourt, Nesreen K. Ahmed, Nedim Lipka, Ruiyi Zhang, Xiang Chen, Tong Yu, Sungchul Kim, Hanieh Deilamsalehy, Namyong Park, Mike Rimer, Zhehao Zhang, Huanrui Yang, Ryan A. Rossi, and Thien Huu Nguyen. 2024. A survey of small language models.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. 2024. The fineweb datasets: Decanting the web for the finest text data at scale.

PyTorch Labs. 2023. Gpt-fast. GitHub repository.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer.

Jean-Loup Tastet and Inar Timiryasov. 2024. Babyllama-2: Ensemble-distilled models consistently outperform teachers with limited data.

Omkar Thawakar, Ashmal Vayani, Salman Khan, Hisham Cholakal, Rao M. Anwer, Michael Felsberg, Tim Baldwin, Eric P. Xing, and Fahad Shahbaz Khan. 2024. Mobillama: Towards accurate and lightweight fully transparent gpt.

Inar Timiryasov and Jean-Loup Tastet. 2023. Baby llama: knowledge distillation from an ensemble of teachers trained on a small dataset with no performance penalty.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing.

x.ai. 2023. Grok os. Blog post.

Leyang Xue, Yao Fu, Zhan Lu, Luo Mai, and Mahesh Marina. 2024. Moe-infinity: Offloading-efficient moe model serving.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. 2024. Qwen2 technical report.

Rongjie Yi, Liwei Guo, Shiyun Wei, Ao Zhou, Shangguang Wang, and Mengwei Xu. 2023. Edgemoe: Fast on-device inference of moe-based large language models.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew Dai, Zhifeng Chen, Quoc Le, and James Laudon. 2022. Mixture-of-experts with expert choice routing.