Dense Backpropagation Improves Routing for Sparsely-Gated Mixture-of-Experts

Ashwinee Panda*, Vatsal Baherwani*, Benjamin Thérien, Stephen Rawls, Sambit Sahu,
Supriyo Chakraborty, Tom Goldstein

Abstract

Mixture of Experts (MoE) pretraining is more scalable than dense Transformer pretraining, because MoEs learn to route inputs to a sparse set of their feedforward parameters. However, this means that MoEs only receive a sparse backward update, leading to problems such as router load imbalance where some experts receive more tokens than others. We present a lightweight approximation method that gives the MoE a dense gradient while only sparsely activating its parameters. A key insight into the design of our method is that at scale, many tokens not routed to a given expert may nonetheless lie in the span of tokens that were routed to that expert, allowing us to create an approximation for the expert output of that token from existing expert outputs. Our dense backpropagation outperforms standard TopK routing across multiple MoE configurations without increasing runtime.

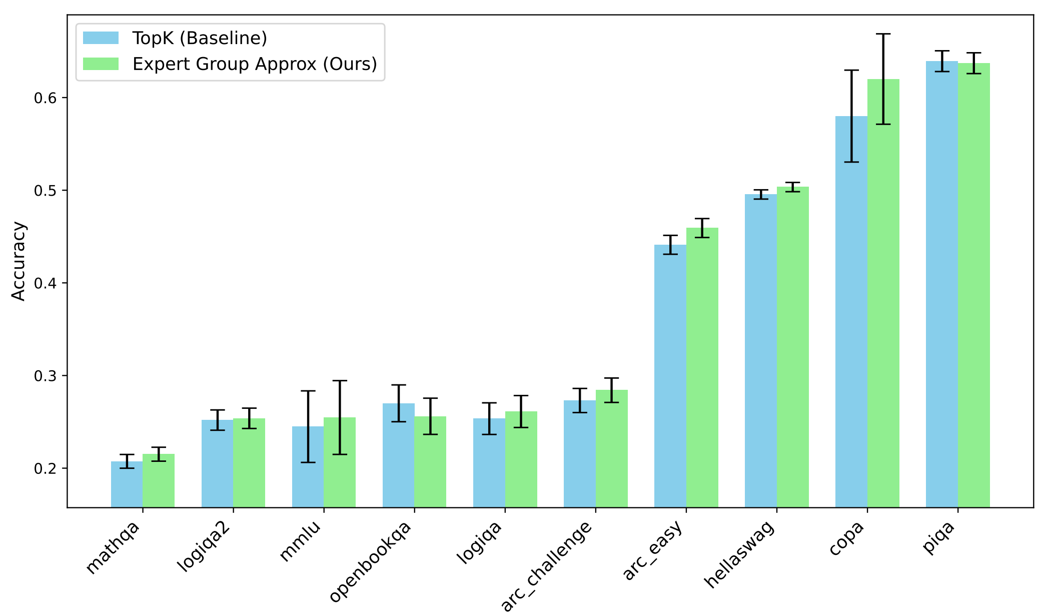


Figure 1: Our dense backpropagation method (green) improves over the baseline (blue) that uses Top-K routing across a range of standard benchmarks. See Section 4 for the experimental setup.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

1 Introduction

Large-scale pretraining hinges on scaling up the number of parameters in a model, because models with more parameters are more sample-efficient and require less training to reach the same performance as smaller models [Kaplan et al., 2020, Hoffmann et al., 2022]. The most common model architecture is a dense Transformer architecture [Vaswani et al., 2023] because its performance scales well with parameters and data. However, a sparsely activated Mixture-of-Experts (MoE) Transformer architecture [Shazeer et al., 2017] has been used by many industry deployments [Team et al., 2024, xAI, 2024, Databricks, 2024, Jiang et al., 2024, Snowflake, 2024, DeepSeek-AI et al., 2024] because MoEs have been shown to scale even better than dense Transformers [Clark et al., 2022, Du et al., 2022, Lepikhin et al., 2020, Fedus et al., 2022]. MoEs learn a *routing* function that selectively activates the TopK subset of their modules, or *experts*, most relevant to a given input. This conditionally sparse activation [Jacobs et al., 1991, Jordan and Jacobs, 1994] allows us to multiplicatively increase the model parameter count without significantly increasing the cost of training or inference.

The sparse router enables MoEs to scale, but it also presents a challenge, because the router does not receive a gradient update from experts that it does not activate, and may not learn to route a token to its appropriate expert. This may cause load imbalance— where a few experts are over-utilized — leading to inefficient training and resource usage [Zoph et al., 2022, Zhou et al., 2022].

In this work we propose a new router that can receive a dense gradient update from a sparse forward pass to address the instability issues arising from sparse routing. Our method adds minimal overhead, but improves on the common Top-K routing in both performance and load balance.

2 Background & Related Work

MoEs. The MoE layer replaces the feedforward networks (FFN) of transformers and consists of two components : **1)** N FFNs (*experts*), $E_0(x), E_1(x), \ldots E_N(x)$ and **2)** a router that assigns tokens to experts. Each input to the MoE layer is processed by K experts where $K < N$, and this is the source of sparsity in MoEs. The K experts are chosen by the router, which is a learnable component that maps each token to a set of weights over the experts. The router performs a linear transformation $\mathbb{R}^{d_{\text{token}}} \rightarrow \mathbb{R}^N$ which produces logits; these are normalized using softmax, resulting in a probability distribution over the experts. With the router's linear transformation parameterized by a matrix W, we can represent the expert weights π in the following way:

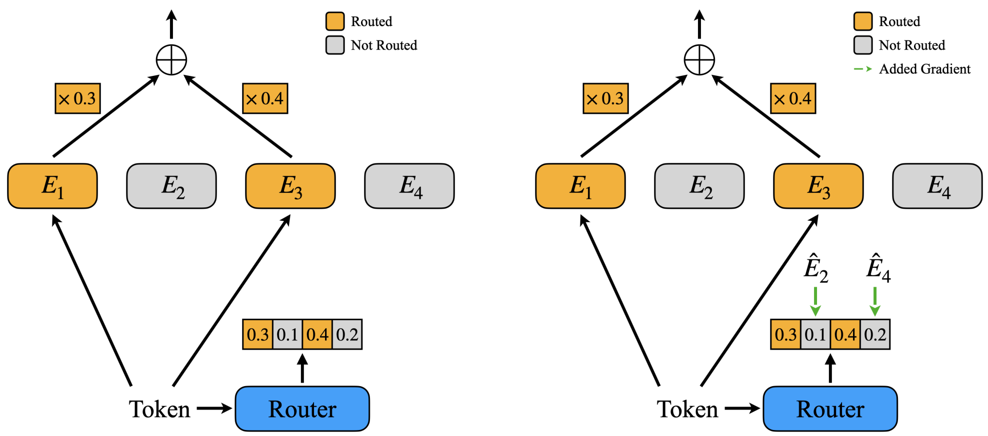
$$\pi \in \mathbb{R}^N = \text{Softmax}(Wx) \tag{1}$$

Once we have these expert weights, we apply a routing function to decide which of K experts to route and process this token through. We consider TopK because it is the most popular.

Top-K routing. A standard method to select K out of N experts given the expert weights is to select the experts corresponding to the K highest weights. Top-K routing [Fedus et al., 2022] passes the token to the K selected experts and averages the expert outputs using these weights to produce the final output. Experts not selected by the Top-K routing function do not process the token, and this introduces sparsity in MoEs. By representing the K chosen experts as the set A, we can express the output of the MoE layer as an average of expert outputs weighted by the router scores:

$$y = \Sigma_{i \in A} \pi_i E_i(x) \tag{2}$$

The expert weights serve two roles. They are used by the routing function to decide which of the K experts to process a token through, and also provide the weights for combining the outputs of the expert. The Top-K routing scheme makes the MoE layer desirable for training large, compute-efficient neural networks. It allows models to be scaled up, by way



(a) Original Router

(b) Dense Approximation Router (ours)

Figure 2: **Overview of Routing with Dense Approximations**. The original mixture of experts router only receives gradients corresponding to experts the token is routed to, because there is no output from other experts. Our approach provides the router with a complete (dense) gradient, by approximating the activations of experts that a token is not routed to. As indicated by the dashed green arrows, the approximated gradients are not actually connected to the token in the computation graph; instead, they are artificially applied in the backward pass.

of increasing the total number of experts, while keeping the compute per token constant (as it is a function of K and not N).

The Router Gradient. Consider the gradient of the MoE layer's output y with respect to the router parameters W. We express y as a function of W by combining Eq. (1) and Eq. (2). With the chain rule, we can backpropagate through this function by considering the gradient at each respective step:

$$\frac{\partial y}{\partial W} = \frac{\partial y}{\partial \pi}\frac{\partial \pi}{\partial W} \tag{3}$$

The first term in Eq. (3), $\frac{\partial y}{\partial \pi}$, is straightforward to compute because the steps in Eq. (1) are easily differentiable, as they consist of linear operations and activations. But Eq. (2) is not differentiable because Top-K expert selection transforms the continuous router weights $\pi \in \mathbb{R}^N$ into a discrete set of selected experts A with $\binom{N}{K}$ possible values. One way to get around backpropagation of nondifferentiable operations is to use the straight-through estimator [Bengio et al., 2013], which treats the operator as the identity function. With straight-through we bypass the Top-K routing function and Eq. (2) becomes the dot product between π and the vector of all $E_i(x)$ with the following gradient:

$$\frac{\partial y}{\partial \pi} = [E_1(x), \quad E_2(x) \quad \cdots \quad E_N(x)] \tag{4}$$

This gradient requires the output of *all* of the experts for that token. Passing a token through all the experts will destroy the sparsity of the MoE layer, ruining the scalability of this architecture. In this work, we develop methods for applying the straight-through estimator while maintaining the sparsity of the MoE layer by *approximating* the output of the experts not selected by Top-K routing.

Related Works. Previous work has tried to address the issue of routing in MoEs. Separate from Top-K is the Sinkhorn routing method [Clark et al., 2022]. Fedus et al. [2022] propose an auxiliary loss that encourages load balancing. Dai et al. [2024] propose multiple additional auxiliary loss terms. Recently, Wang et al. [2024] propose learning biases rather than an

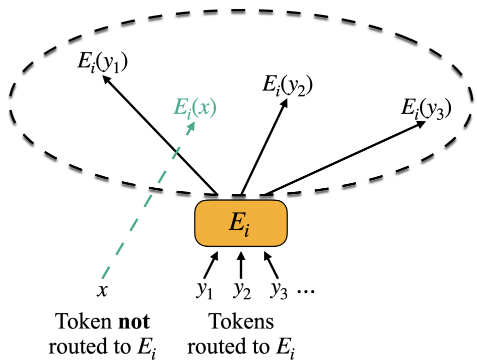


Figure 3: **Motivation for approximating expert outputs.** Consider an expert E_i. In every training batch, some fraction of tokens Y will be routed to E_i. With a large training batch size, we expect more tokens y_j to be routed to E_i. It is likely, then, that the outputs $E_i(y_j)$ will span the output space of this expert. Thus we expect that at the pretraining scale, a linear combination of expert outputs $\sum_j \alpha_j E_i(y_j)$ can approximate $E_i(x)$, for a token x which was not routed to E_i.

auxiliary load balancing loss. Even more recently, Phi-3.5-MoE [Abdin et al., 2024] uses SparseMixer [Liu et al., 2024, 2023], another estimator for $\partial y/\partial \pi$ not involving straight-through. Our approach is to still use straight-through, but *approximate* these additional expert outputs. We now present our method.

3 Designing a New Routing Method

In this section we design a new router that can receive a dense gradient update while being sparsely activated. In a standard MoE, the embedding corresponding to expert i in the routing layer (i.e. the ith row of the routing weight matrix) receives no gradient update from a token x if x is not routed to expert i. This is because $E_i(x)$ is never computed, so it provides no upstream gradient. This corresponds to experts that are not in the top K being omitted in Eq. (2). We apply an approximation $\hat{E}_i(x)$ as a substitute for the upstream gradient, so that the router can receive some non-zero signal corresponding to this expert. Then, the router can factor in outputs from all experts when learning to route each token. We pass this approximation forward during training and update the experts.

3.1 Approximating Expert Activations

To approximate the dense gradient in Eq. (4), we must approximate $E_i(x)$ for every expert i that a token x was not passed to. Although we have no information about what the function E_i looks like for x, when training with large token batch sizes it is very likely that we have outputs of E_i for many other tokens. We hypothesize that although $E_i(x)$ is not computed for a given x, the output lies in the span of other tokens' activations for E_i. In other words, $\hat{E}_i(x) = \sum_j c_j E_i(x_j)$ for some other tokens x_j. Our approximation method relies on finding these relevant x_j and the corresponding weights c_j to develop an approximation for a token. We visualize this general approach in Fig. 3.

We develop two approaches to produce an estimator $\hat{E}_i(x)$, using the expert outputs of other relevant tokens. *Expert group approximation:* We first apply a single approximation to a large group of tokens that were not routed to expert i. This is efficient, but it may not be a good approximation for any specific x. Instead, this is an estimator for the expert output across the entire batch - this is sufficient as we will only need an approximation for the batch gradient to update the router during training. In the expert group approximation, we select relevant x_j and c_j purely based on the router output. *Attention approximation:* Our second approach produces an expert output approximation specifically for each token, where we find relevant x_j using dot product similarities between tokens.

3.2 Notation on Expert Routing

Let $R(x)$ be the set of indices corresponding to the K experts that a token x is routed to. This can be thought of as the *routing decision* for x, based on the selected experts A in Eq. (2). For example, in a top-k sparse mixture of experts block with $N = 8$ experts and $K = 2$, x routed to the first and last experts will have $R(x) = \{1, 8\}$. Note that $R(x)$ can have $\binom{N}{K}$ possible discrete outputs. With this notation, we can partition all tokens X based on their routing decisions and denote X_R as the subset of tokens routed to experts indexed by R. In the preceding example, the token routed to the first and last experts would belong to the set $X_{\{1,8\}}$. Some of our methods involve denoting whether a token was routed to a set of experts instead of its exact routing decision. We denote tokens routed to expert i along with any other experts as $X_{\{i,\cdot\}}$. For example, $X_{\{1,8\}} = X_{\{1,\cdot\}} \bigcap X_{\{8,\cdot\}}$

3.3 Expert Group Approximation

We first consider the case where we approximate the expert output $E_i(x)$ for many tokens at a time. For a token x, we want to approximate outputs of experts that x was not routed to, i.e. $E_i(x)$ where $i \notin R(x)$. We hypothesize that tokens being routed to the same expert is a strong indicator of similarity between the tokens. This is because tokens routed to the same expert E_i both yield high dot products with the embedding W_i corresponding to that expert in the routing layer (see Eq. (1)). We develop an approximation for $E_i(x)$ by aggregating outputs of E_i for tokens that were routed to both expert i, and other experts that x was routed to. Formally, we consider an alternate routing decision $R' = \{i, j, \cdot\}, j \in R(x)$ that consists of one expert j that x is routed to, the expert i we wish to approximate, and any other experts (if $K > 2$). Then, the adjacent token space $X_{R'}$ will consist of tokens that are very similar to x by virtue of having similar routing decisions. Moreover, they will be routed to expert i, and we hypothesize that their outputs $\sum_{x' \in X_{R'}} E_i(x')$ will approximately represent $E_i(x)$. We can aggregate such outputs over all possible routing decisions:

$$\forall x \in X_R : \hat{E}_i(x) = \frac{1}{K} \sum_{j \in R} \frac{1}{|X_{\{i,j,\cdot\}}|} \sum_{x' \in X_{\{i,j,\cdot\}}} E_i(x') \tag{5}$$

We apply a single aggregate approximation for each routing decision to all tokens with that routing decision. In other words, each token belongs to K "groups" corresponding to the K experts it was routed to. Each expert has N groups, and each group j receives an approximation for expert i based on tokens routed to experts i, j. This gives us N^2 total approximations. When we approximate expert i, a token receives one approximation from each group, scaled by $\frac{1}{K}$ to take the average. In Fig. 4 we visualize this method for $K = 2$. We can pick different constants for the number of activated experts and the number of groups, so our method is applicable to $K = 1$, but we keep these constants at $K = 2$ throughout the paper because it is a common choice across prior work.

We also pass the approximations to future layers and update the experts. Although increasing the number of updated experts by N/k times may seem like it will pose high overhead, our method in fact runs at nearly the same speed as standard Top-K routing even with a relatively unoptimized implementation (see Table 4). This is because the gradient $\frac{\partial L}{\partial E}$ for an expert that computed an approximation as $\hat{E}(x) = c_1 \cdot E(Y) + c_2 \cdot E(Z)$ is just $\frac{\partial L}{\partial E} = (1 + c_1)\frac{\partial L}{\partial E(Y)} + (1 + c_2)\frac{\partial L}{\partial E(Z)}$, where the Top-K gradient would have been $\frac{\partial L}{\partial E} = \frac{\partial L}{\partial E(Y)} + \frac{\partial L}{\partial E(Z)}$, so our dense update has no additional cost.

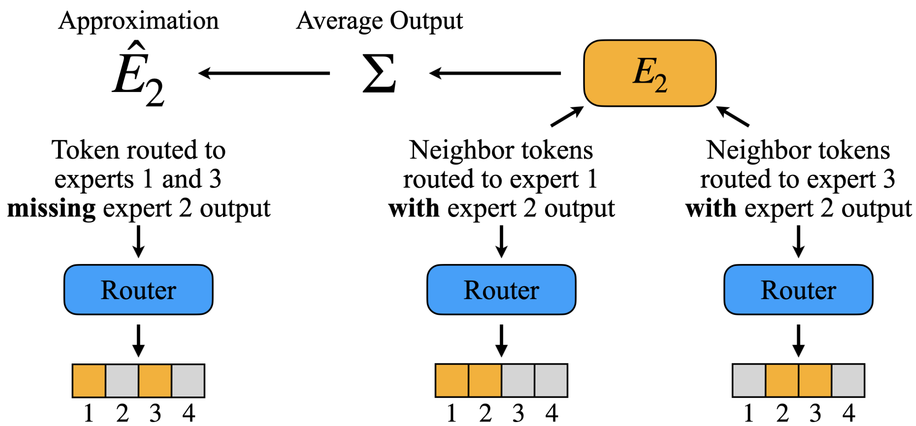


Figure 4: **Architecture of the Expert Group Approximation method.** In this example, we have 4 experts with $K = 2$. Consider all inputs routed to experts 1 and 3, characterized by the routing decision $R = \{1, 3\}$. As described in Figure 2b, we need to approximate these inputs' activations for all other experts. In approximating expert 2, for example, we collect all inputs x' with a routing decision similar to R specifically including expert 2: $R' = \{1, 2\}$ and $R' = \{2, 3\}$. In general there will be K such adjacent groups. The aggregation of these inputs' activations for expert 2 is used to approximate expert 2 for all inputs routed to experts 1 and 3.

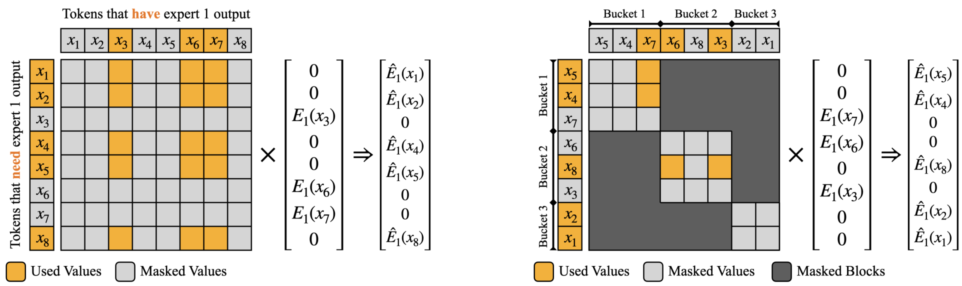
3.4 Token-Specific Approximation with Attention

Our Expert Group Approximation computes an approximation, for each expert, for all tokens routed to it from each other expert, resulting in N^2 total approximations. However, we may want to actually compute an approximation for specific tokens. Consider tokens belonging to the set $x \in X^C_{\{i,\cdot\}}$, i.e. tokens *not* routed to expert i. We want to approximate $E_i(x)$ for such x. At a high level, we want to search for similar tokens to x, select their expert outputs $E_i(x_j)$, and aggregate these outputs as a weighted linear combination. This problem can be solved using attention. We want to query using all tokens *not* routed to expert i, i.e. $X^C_{\{i,\cdot\}}$. The keys will correspond to tokens that *were* routed to expert i, i.e. $X_{\{i,\cdot\}}$. And the values will be the expert outputs of these relevant tokens. Fig. 5a (left) outlines how we compute an approximation using multi-head attention, where each head corresponds to approximating for a single expert.

Sparse Attention with LSH. Computing attention across all tokens on a GPU is computationally expensive, and we do not need attention scores for *all* the tokens to compute the approximation, just for the most similar tokens to x. With a block-sparse attention mask, we can greatly reduce the attention computation, especially when most of the computed scores would be redundant. In Fig. 5b we outline our attention approximation that uses locality-sensitive hashing (LSH) to group tokens into buckets, with a high probability that the nearest neighbors to a token will lie in the same bucket. The attention mask now has an additional condition: the query index q and key index k must correspond to tokens in the same bucket. We sort the QKV into groups based on their assigned buckets to encourage a block-diagonal attention mask, and verify that this sparsity reduces the runtime of our attention approximation. Note that it is possible that some tokens receive no approximation because there are no keys to query in the bucket. In this case, we set the approximation to 0.

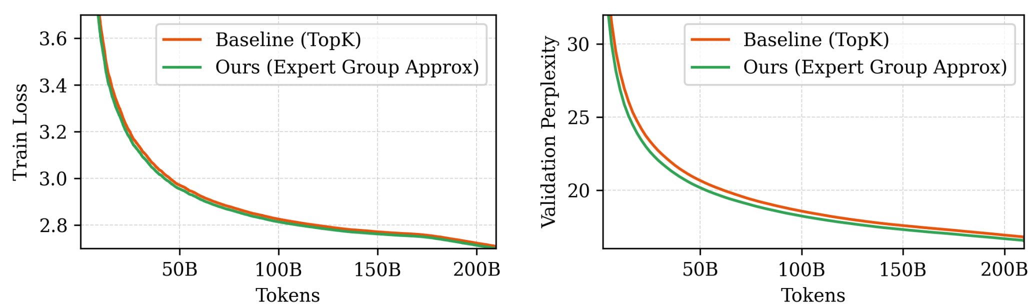
4 Evaluation

Model Architectures. We train two MoEs. Both have 2B total parameters, and use the standard $K = 2$ top-K routing [Zoph et al., 2022]. The first is a Deepseek [DeepSeek-AI et al.,



(a) Direct Attention (b) Attention w/ LSH

Figure 5: **Attention scores of direct and LSH attention methods**. For each expert, we define an attention head that uses queries corresponding to inputs not routed to the expert, and keys corresponding to inputs routed to the expert. Grey entries denote queries and keys that do not meet this criteria, and whose attention scores are masked out. We multiply the attention scores of each head with the values, which are expert outputs of tokens routed to that attention head's expert. This implementation is common to both the direct and LSH attention method. In the latter, we further optimize the attention calculation by sorting inputs into buckets based on cosine similarity. This creates a block-sparse attention map, allowing kernels to skip most of the attention computation.



(a) Train Loss (b) Validation Perplexity

Figure 6: **Training and Validation Results.** We train the Deepseek MoE with 32 experts, using the standard Top-K=2 router and our expert group approximation method, each on 200B tokens of the Fineweb dataset. Without incurring significant overhead, we improve over the baseline.

2024] MoE with 32 *fine-grained* experts (each expert is a bottleneck MLP to maintain 2B total parameters) that has 470M active parameters. The second is a conventional MoE with 780M active parameters.

Dataset. We train on FineWeb [Penedo et al., 2024] with the Llama3 tokenizer [Dubey et al., 2024]. We split it into train, validation, and test splits and report the validation perplexity. We train on 200B tokens for the Deepseek-style MoE and 20B tokens for the standard-architecture MoE.

We defer other implementation details (learning rate schedule, initializations) to Appendix C.

4.1 Main Results

Main Result. Our main result compares the Expert Group Approximation, which performs a dense update of the router weights by approximating the dense gradient, to baseline Top-K routing. In Fig. 6 we plot both training loss and validation set perplexity over the course of training. Our Expert Group Approximation method yields improvements over the baseline Top-K routing method throughout training, and this improvement is still

Table 1: Our expert group approximation obtains the best validation perplexity after 20B tokens on a standard MoE architecture, achieving the same performance as $K = 3$ without activating an additional expert, meaning that we achieve the best performance-FLOPS tradeoff.

Activated Experts	Routing Method	Validation Perplexity
$K = 1$	Baseline	19.61
$K = 2$	Baseline	18.92
$K = 3$	Baseline	18.56
$K = 2$	Expert Group Approx. (Ours)	**18.55**

present after training on 200B tokens. We further evaluate the performance of our Expert Group Approximation on standard benchmark tasks in Fig. 1 and find that our method provides a modest improvement over Top-K across multiple benchmarks. We selected these benchmarks because Penedo et al. [2024] notes these are "high signal" benchmarks for models trained on Fineweb, which is our training dataset.

As we will show, our method improves performance by improving load balance. Load balancing is especially important early on in training – this is where our method shows the largest improvement in Fig. 6. Improving the load balance of experts yields benefits akin to actually activating more experts. In Table 1 we find that our lightweight approximation method improves performance by a similar amount as activating an additional expert (that is, going from $K = 2$ to $K = 3$), without the additional computational overhead during training and inference of actually needing to use the parameters of a third expert. The choice of $K = 2$ in all experiments follows Zoph et al. [2022].

Load Balance. We believe the secret of our method's superior performance lies in how it improves the routing distribution. Without a gradient signal for unactivated experts, top-K routing may not learn a balanced distribution across experts, and route more tokens to some experts than others. On the other hand, our method sends a signal to the router even for experts that a specific token was not routed to. The router now receives more information to make better decisions on how to route inputs. Load imbalancing leads to some experts processing a larger fraction of tokens than others, and thus having to learn a representation for more tokens. although all experts share the same number of parameters. Balancing expert load thus allows for more efficient utilization of MoE parameters.

In Fig. 7 we validate that the baseline top-K ($K = 2$) routing has an "imbalanced load", as measured by the proportion of tokens being routed to different experts relative to the optimal balance (green line) of an even distribution of tokens across experts. Our method improves load balance, which may be one cause for improved performance and is of independent interest on its own because it may lead to more efficient inference (although we do not profile inference workloads in this work).

Ablations. We conduct further ablations on design choices and efficiency in Appendix C.1.

5 Discussion

We propose a training method for MoEs to improve load balancing and language modeling performance. By approximating the signal of a dense mixture-of-experts layer, the MoE router is able to learn a better distribution of routing inputs to different experts. This approximated dense signal unlocks the possibility for more sparse MoEs at training and inference time. Whereas typical Top-K routing would provide too sparse of a signal to learn a stable routing distribution, our method demonstrates significant improvements in load balancing in very sparse configurations. This unlocks better performance on standard language benchmarks, and has little overhead.

Limitations. The scope of our evaluation is limited. We plan to add more benchmark results in a future version of the paper, but had some difficulty setting up tasks in Gao et al. [2024]. We only train models for at most 200B tokens, and the largest MoE we train has

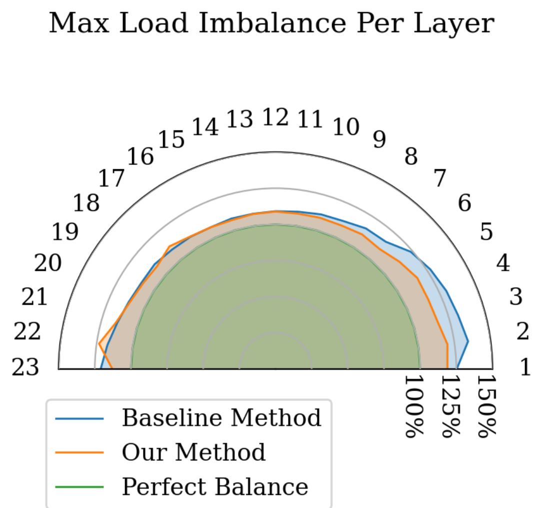


Figure 7: **Load Balancing using Expert Group Approximation**. We define maximum load imbalance at each layer as $\max_i(N \cdot f_i)$ where f_i is the fraction of tokens routed to each of N experts. The green ring indicates perfect balance, where each expert receives $1/N$ fraction of tokens; the outer ring indicates a maximum imbalance of 150%. We record maximum imbalance after training on 3 billion tokens. Our method improves load balance by sending a complete gradient to the router.

fewer than 1B active parameters. We plan to address these limitations in a future version of this work.

References

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020. URL `https://arxiv.org/abs/2001.08361`.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models, 2022. URL `https://arxiv.org/abs/2203.15556`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL `https://arxiv.org/abs/1706.03762`.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017. URL `https://arxiv.org/abs/1701.06538`.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, Soroosh Mariooryad, Yifan Ding, Xinyang Geng, Fred Alcober, Roy Frostig, Mark Omernick, Lexi Walker, Cosmin Paduraru, Christina Sorokin, Andrea Tacchetti, Colin Gaffney, Samira Daruki, Olcan Sercinoglu, Zach Gleicher, Juliette Love, Paul Voigtlaender, Rohan Jain, Gabriela Surita, Kareem Mohamed, Rory Blevins, Junwhan Ahn, Tao Zhu, Kornraphop Kawintiranon, Orhan Firat, Yiming Gu, Yujing Zhang, Matthew Rahtz, Manaal Faruqui, Natalie Clay, Justin Gilmer, JD Co-Reyes, Ivo Penchev, Rui Zhu, Nobuyuki Morioka, Kevin Hui, Krishna Haridasan,

Victor Campos, Mahdis Mahdieh, Mandy Guo, Samer Hassan, Kevin Kilgour, Arpi Vezer, Heng-Tze Cheng, Raoul de Liedekerke, Siddharth Goyal, Paul Barham, DJ Strouse, Seb Noury, Jonas Adler, Mukund Sundararajan, Sharad Vikram, Dmitry Lepikhin, Michela Paganini, Xavier Garcia, Fan Yang, Dasha Valter, Maja Trebacz, Kiran Vodrahalli, Chulayuth Asawaroengchai, Roman Ring, Norbert Kalb, Livio Baldini Soares, Siddhartha Brahma, David Steiner, Tianhe Yu, Fabian Mentzer, Antoine He, Lucas Gonzalez, Bibo Xu, Raphael Lopez Kaufman, Laurent El Shafey, Junhyuk Oh, Tom Hennigan, George van den Driessche, Seth Odoom, Mario Lucic, Becca Roelofs, Sid Lall, Amit Marathe, Betty Chan, Santiago Ontanon, Luheng He, Denis Teplyashin, Jonathan Lai, Phil Crone, Bogdan Damoc, Lewis Ho, Sebastian Riedel, Karel Lenc, Chih-Kuan Yeh, Aakanksha Chowdhery, Yang Xu, Mehran Kazemi, Ehsan Amid, Anastasia Petrushkina, Kevin Swersky, Ali Khodaei, Gowoon Chen, Chris Larkin, Mario Pinto, Geng Yan, Adria Puigdomenech Badia, Piyush Patil, Steven Hansen, Dave Orr, Sebastien M. R. Arnold, Jordan Grimstad, Andrew Dai, Sholto Douglas, Rishika Sinha, Vikas Yadav, Xi Chen, Elena Gribovskaya, Jacob Austin, Jeffrey Zhao, Kaushal Patel, Paul Komarek, Sophia Austin, Sebastian Borgeaud, Linda Friso, Abhimanyu Goyal, Ben Caine, Kris Cao, Da-Woon Chung, Matthew Lamm, Gabe Barth-Maron, Thais Kagohara, Kate Olszewska, Mia Chen, Kaushik Shivakumar, Rishabh Agarwal, Harshal Godhia, Ravi Rajwar, Javier Snaider, Xerxes Dotiwalla, Yuan Liu, Aditya Barua, Victor Ungureanu, Yuan Zhang, Bat-Orgil Batsaikhan, Mateo Wirth, James Qin, Ivo Danihelka, Tulsee Doshi, Martin Chadwick, Jilin Chen, Sanil Jain, Quoc Le, Arjun Kar, Madhu Gurumurthy, Cheng Li, Ruoxin Sang, Fangyu Liu, Lampros Lamprou, Rich Munoz, Nathan Lintz, Harsh Mehta, Heidi Howard, Malcolm Reynolds, Lora Aroyo, Quan Wang, Lorenzo Blanco, Albin Cassirer, Jordan Griffith, Dipanjan Das, Stephan Lee, Jakub Sygnowski, Zach Fisher, James Besley, Richard Powell, Zafarali Ahmed, Dominik Paulus, David Reitter, Zalan Borsos, Rishabh Joshi, Aedan Pope, Steven Hand, Vittorio Selo, Vihan Jain, Nikhil Sethi, Megha Goel, Takaki Makino, Rhys May, Zhen Yang, Johan Schalkwyk, Christina Butterfield, Anja Hauth, Alex Goldin, Will Hawkins, Evan Senter, Sergey Brin, Oliver Woodman, Marvin Ritter, Eric Noland, Minh Giang, Vijay Bolina, Lisa Lee, Tim Blyth, Ian Mackinnon, Machel Reid, Obaid Sarvana, David Silver, Alexander Chen, Lily Wang, Loren Maggiore, Oscar Chang, Nithya Attaluri, Gregory Thornton, Chung-Cheng Chiu, Oskar Bunyan, Nir Levine, Timothy Chung, Evgenii Eltyshev, Xiance Si, Timothy Lillicrap, Demetra Brady, Vaibhav Aggarwal, Boxi Wu, Yuanzhong Xu, Ross McIlroy, Kartikeya Badola, Paramjit Sandhu, Erica Moreira, Wojciech Stokowiec, Ross Hemsley, Dong Li, Alex Tudor, Pranav Shyam, Elahe Rahimtoroghi, Salem Haykal, Pablo Sprechmann, Xiang Zhou, Diana Mincu, Yujia Li, Ravi Addanki, Kalpesh Krishna, Xiao Wu, Alexandre Frechette, Matan Eyal, Allan Dafoe, Dave Lacey, Jay Whang, Thi Avrahami, Ye Zhang, Emanuel Taropa, Hanzhao Lin, Daniel Toyama, Eliza Rutherford, Motoki Sano, HyunJeong Choe, Alex Tomala, Chalence Safranek-Shrader, Nora Kassner, Mantas Pajarskas, Matt Harvey, Sean Sechrist, Meire Fortunato, Christina Lyu, Gamaleldin Elsayed, Chenkai Kuang, James Lottes, Eric Chu, Chao Jia, Chih-Wei Chen, Peter Humphreys, Kate Baumli, Connie Tao, Rajkumar Samuel, Cicero Nogueira dos Santos, Anders Andreassen, Nemanja Rakićević, Dominik Grewe, Aviral Kumar, Stephanie Winkler, Jonathan Caton, Andrew Brock, Sid Dalmia, Hannah Sheahan, Iain Barr, Yingjie Miao, Paul Natsev, Jacob Devlin, Feryal Behbahani, Flavien Prost, Yanhua Sun, Artiom Myaskovsky, Thanumalayan Sankaranarayana Pillai, Dan Hurt, Angeliki Lazaridou, Xi Xiong, Ce Zheng, Fabio Pardo, Xiaowei Li, Dan Horgan, Joe Stanton, Moran Ambar, Fei Xia, Alejandro Lince, Mingqiu Wang, Basil Mustafa, Albert Webson, Hyo Lee, Rohan Anil, Martin Wicke, Timothy Dozat, Abhishek Sinha, Enrique Piqueras, Elahe Dabir, Shyam Upadhyay, Anudhyan Boral, Lisa Anne Hendricks, Corey Fry, Josip Djolonga, Yi Su, Jake Walker, Jane Labanowski, Ronny Huang, Vedant Misra, Jeremy Chen, RJ Skerry-Ryan, Avi Singh, Shruti Rijhwani, Dian Yu, Alex Castro-Ros, Beer Changpinyo, Romina Datta, Sumit Bagri, Arnar Mar Hrafnkelsson, Marcello Maggioni, Daniel Zheng, Yury Sulsky, Shaobo Hou, Tom Le Paine, Antoine Yang, Jason Riesa, Dominika Rogozinska, Dror Marcus, Dalia El Badawy, Qiao Zhang, Luyu Wang, Helen Miller, Jeremy Greer, Lars Lowe Sjos, Azade Nova, Heiga Zen, Rahma Chaabouni, Mihaela Rosca, Jiepu Jiang, Charlie Chen, Ruibo Liu, Tara Sainath, Maxim Krikun, Alex Polozov, Jean-Baptiste Lespiau, Josh Newlan, Zeyncep Cankara, Soo Kwak, Yunhan Xu, Phil Chen, Andy Coenen, Clemens Meyer, Katerina Tsihlas, Ada Ma, Juraj Gottweis, Jinwei Xing, Chenjie Gu, Jin Miao, Christian Frank, Zeynep Cankara, Sanjay Ganapathy, Ishita Dasgupta, Steph Hughes-Fitt, Heng Chen, David Reid, Keran Rong, Hongmin Fan, Joost van Amersfoort, Vincent Zhuang, Aaron Cohen, Shixiang Shane Gu, Anhad Mohananey, Anastasija Ilic, Taylor Tobin, John Wieting, Anna Bortsova, Phoebe Thacker, Emma Wang, Emily Caveness, Justin Chiu, Eren Sezener, Alex Kaskasoli, Steven Baker, Katie Millican, Mohamed Elhawaty, Kostas Aisopos, Carl Lebsack, Nathan Byrd, Hanjun Dai, Wenhao Jia, Matthew Wiethoff, Elnaz Davoodi, Albert Weston, Lakshman Yagati, Arun Ahuja, Isabel Gao, Golan Pundak, Susan Zhang, Michael Azzam, Khe Chai Sim, Sergi Caelles, James Keeling, Abhanshu Sharma, Andy Swing, YaGuang Li, Chenxi Liu, Carrie Grimes Bostock, Yamini Bansal, Zachary Nado, Ankesh Anand, Josh Lipschultz, Abhijit Karmarkar, Lev Proleev, Abe Ittycheriah, Soheil Hassas Yeganeh, George Polovets, Aleksandra Faust, Jiao Sun, Alban Rrustemi, Pen Li, Rakesh Shivanna, Jeremiah Liu, Chris Welty, Federico Lebron, Anirudh Baddepudi, Sebastian Krause, Emilio Parisotto, Radu Soricut, Zheng Xu, Dawn Bloxwich, Melvin Johnson, Behnam Neyshabur, Justin Mao-Jones, Renshen Wang, Vinay Ramasesh, Zaheer Abbas, Arthur Guez, Constant Segal, Duc Dung Nguyen, James Svensson, Le Hou, Sarah York, Kieran Milan, Sophie Bridgers, Wiktor Gworek, Marco Tagliasacchi, James Lee-Thorp, Michael Chang, Alexey Guseynov, Ale Jakse Hartman, Michael Kwong, Ruizhe Zhao, Sheleem Kashem, Elizabeth Cole, Antoine Miech, Richard Tanburn, Mary Phuong, Filip Pavetic, Sebastien Cevey, Ramona Comanescu, Richard Ives, Sherry Yang, Cosmo Du, Bo Li, Zizhao Zhang, Mariko Iinuma, Clara Huiyi Hu, Aurko Roy, Shaan Bijwadia, Zhenkai Zhu, Danilo Martins, Rachel Saputro, Anita Gergely, Steven Zheng, Dawei Jia, Ioannis Antonoglou, Adam Sadovsky, Shane Gu, Yingying Bi, Alek Andreev, Sina Samangooei, Mina Khan, Tomas Kocisky, Angelos Filos, Chintu Kumar, Colton Bishop, Adams Yu, Sarah Hodkinson, Sid Mittal, Premal Shah, Alexandre Moufarek, Yong Cheng, Adam Bloniarz, Jaehoon Lee, Pedram Pejman, Paul Michel, Stephen Spencer, Vladimir Feinberg, Xuehan Xiong, Nikolay Savinov, Charlotte Smith, Siamak Shakeri, Dustin Tran, Mary Chesus, Bernd Bohnet, George Tucker, Tamara von Glehn, Carrie Muir, Yiran Mao, Hideto Kazawa, Ambrose Slone, Kedar Soparkar, Disha Shrivastava, James Cobon-Kerr, Michael Sharman, Jay Pavagadhi, Carlos Araya, Karolis Misiunas, Nimesh Ghelani, Michael Laskin, David Barker, Qiujia Li, Anton Briukhov, Neil Houlsby, Mia Glaese, Balaji Lakshminarayanan, Nathan Schucher, Yunhao Tang, Eli Collins, Hyeontaek Lim, Fangxiaoyu Feng, Adria Recasens, Guangda Lai, Alberto Magni, Nicola De Cao, Aditya Siddhant, Zoe Ashwood, Jordi Orbay, Mostafa Dehghani, Jenny Brennan, Yifan He, Kelvin Xu, Yang Gao, Carl Saroufim, James Molloy, Xinyi Wu, Seb Arnold, Solomon Chang, Julian Schrittwieser, Elena Buchatskaya, Soroush Radpour, Martin Polacek, Skye Giordano, Ankur Bapna, Simon Tokumine, Vincent Hellendoorn, Thibault Sottiaux, Sarah Cogan, Aliaksei Severyn, Mohammad Saleh, Shantanu Thakoor, Laurent Shefey, Siyuan Qiao, Meenu Gaba, Shuo yiin Chang, Craig Swanson, Biao Zhang, Benjamin Lee, Paul Kishan Rubenstein, Gan Song, Tom Kwiatkowski, Anna Koop, Ajay Kannan, David Kao, Parker Schuh, Axel Stjerngren, Golnaz Ghiasi, Gena Gibson, Luke Vilnis, Ye Yuan, Felipe Tiengo Ferreira, Aishwarya Kamath, Ted Klimenko, Ken Franko, Kefan Xiao, Indro Bhattacharya, Miteyan Patel, Rui Wang, Alex Morris, Robin Strudel, Vivek Sharma, Peter Choy, Sayed Hadi Hashemi, Jessica Landon, Mara Finkelstein, Priya Jhakra, Justin Frye, Megan Barnes, Matthew Mauger, Dennis Daun, Khuslen Baatarsukh, Matthew Tung, Wael Farhan, Henryk Michalewski, Fabio Viola, Felix de Chaumont Quitry, Charline Le Lan, Tom Hudson, Qingze Wang, Felix Fischer, Ivy Zheng, Elspeth White, Anca Dragan, Jean baptiste Alayrac, Eric Ni, Alexander Pritzel, Adam Iwanicki, Michael Isard, Anna Bulanova, Lukas Zilka, Ethan Dyer, Devendra Sachan, Srivatsan Srinivasan, Hannah Muckenhirn, Honglong Cai, Amol Mandhane, Mukarram Tariq, Jack W. Rae, Gary Wang, Kareem Ayoub, Nicholas FitzGerald, Yao Zhao, Woohyun Han, Chris Alberti, Dan Garrette, Kashyap Krishnakumar, Mai Gimenez, Anselm Levskaya, Daniel Sohn, Josip Matak, Inaki Iturrate, Michael B. Chang, Jackie Xiang, Yuan Cao, Nishant Ranka, Geoff Brown, Adrian Hutter, Vahab Mirrokni, Nanxin Chen, Kaisheng Yao, Zoltan Egyed, Francois Galilee, Tyler Liechty, Praveen Kallakuri, Evan Palmer, Sanjay Ghemawat, Jasmine Liu, David Tao, Chloe Thornton, Tim Green, Mimi Jasarevic, Sharon Lin, Victor Cotruta, Yi-Xuan Tan, Noah Fiedel, Hongkun Yu, Ed Chi, Alexander Neitz, Jens Heitkaemper, Anu Sinha, Denny Zhou, Yi Sun, Charbel Kaed, Brice Hulse, Swaroop Mishra, Maria Georgaki, Sneha Kudugunta, Clement Farabet, Izhak Shafran, Daniel Vlasic, Anton Tsitsulin, Rajagopal Ananthanarayanan, Alen Carin, Guolong Su, Pei Sun, Shashank V, Gabriel Carvajal, Josef Broder, Iulia Comsa, Alena Repina, William Wong, Warren Weilun Chen, Peter Hawkins, Egor Filonov, Lucia Loher, Christoph Hirnschall, Weiyi Wang, Jingchen Ye, Andrea Burns, Hardie Cate, Diana Gage Wright, Federico Piccinini, Lei Zhang, Chu-Cheng Lin, Ionel Gog, Yana Kulizhskaya, Ashwin Sreevatsa, Shuang Song, Luis C. Cobo, Anand Iyer, Chetan Tekur, Guillermo Garrido, Zhuyun Xiao, Rupert Kemp, Huaixiu Steven Zheng, Hui Li, Ananth Agarwal, Christel Ngani, Kati Goshvadi, Rebeca Santamaria-Fernandez, Wojciech Fica, Xinyun Chen, Chris Gorgolewski, Sean Sun, Roopal Garg, Xinyu Ye, S. M. Ali Eslami, Nan Hua, Jon Simon, Pratik Joshi, Yelin Kim, Ian Tenney, Sahitya Potluri, Lam Nguyen Thiet, Quan Yuan, Florian Luisier, Alexandra Chronopoulou, Salvatore Scellato, Praveen Srinivasan, Minmin Chen, Vinod Koverkathu, Valentin Dalibard, Yaming Xu, Brennan Saeta, Keith Anderson, Thibault Sellam, Nick Fernando, Fantine Huot, Junehyuk Jung, Mani Varadarajan, Michael Quinn, Amit Raul, Maigo Le, Ruslan Habalov, Jon Clark, Komal Jalan, Kalesha Bullard, Achintya Singhal, Thang Luong, Boyu Wang, Sujeevan Rajayogam, Julian Eisenschlos, Johnson Jia, Daniel Finchelstein, Alex Yakubovich, Daniel Balle, Michael Fink, Sameer Agarwal, Jing Li, Dj Dvijotham, Shalini Pal, Kai Kang, Jaclyn Konzelmann, Jennifer Beattie, Olivier Dousse, Diane Wu, Remi Crocker, Chen Elkind, Siddhartha Reddy Jonnalagadda, Jong Lee, Dan Holtmann-Rice, Krystal Kallarackal, Rosanne Liu, Denis Vnukov, Neera Vats, Luca Invernizzi, Mohsen Jafari, Huanjie Zhou, Lilly Taylor, Jennifer Prendki, Marcus Wu, Tom Eccles, Tianqi Liu, Kavya Kopparapu, Francoise Beaufays, Christof Angermueller, Andreea Marzoca, Shourya Sarcar, Hilal Dib, Jeff Stanway, Frank Perbet, Nejc Trdin, Rachel Sterneck, Andrey Khorlin, Dinghua Li, Xihui Wu, Sonam Goenka, David Madras, Sasha Goldshtein, Willi Gierke, Tong Zhou, Yaxin Liu, Yannie Liang, Anais White, Yunjie Li, Shreya Singh, Sanaz Bahargam, Mark Epstein, Sujoy Basu, Li Lao, Adnan Ozturel, Carl Crous, Alex Zhai, Han Lu, Zora Tung, Neeraj Gaur, Alanna Walton, Lucas Dixon, Ming Zhang, Amir Globerson, Grant Uy, Andrew Bolt, Olivia Wiles, Milad Nasr, Ilia Shumailov, Marco Selvi, Francesco Piccinno, Ricardo Aguilar, Sara McCarthy, Misha Khalman, Mrinal Shukla, Vlado Galic, John Carpenter, Kevin Villela, Haibin Zhang, Harry Richardson, James Martens, Matko Bosnjak, Shreyas Rammohan Belle, Jeff Seibert, Mahmoud Alnahlawi, Brian McWilliams, Sankalp Singh, Annie Louis, Wen Ding, Dan Popovici, Lenin Simicich, Laura Knight, Pulkit Mehta, Nishesh Gupta, Chongyang Shi, Saaber Fatehi, Jovana Mitrovic, Alex Grills, Joseph Pagadora, Dessie Petrova, Danielle Eisenbud, Zhishuai Zhang, Damion Yates, Bhavishya Mittal, Nilesh Tripuraneni, Yannis Assael, Thomas Brovelli, Prateek Jain, Mihajlo Velimirovic, Canfer Akbulut, Jiaqi Mu, Wolfgang Macherey, Ravin Kumar, Jun Xu, Haroon Qureshi, Gheorghe Comanici, Jeremy Wiesner, Zhitao Gong, Anton Ruddock, Matthias Bauer, Nick Felt, Anirudh GP, Anurag Arnab, Dustin Zelle, Jonas Rothfuss, Bill Rosgen, Ashish Shenoy, Bryan Seybold, Xinjian Li, Jayaram Mudigonda, Goker Erdogan, Jiawei Xia, Jiri Simsa, Andrea Michi, Yi Yao, Christopher Yew, Steven Kan, Isaac Caswell, Carey Radebaugh, Andre Elisseeff, Pedro Valenzuela, Kay McKinney, Kim Paterson, Albert Cui, Eri Latorre-Chimoto, Solomon Kim, William Zeng, Ken Durden, Priya Ponnapalli, Tiberiu Sosea, Christopher A. Choquette-Choo, James Manyika, Brona Robenek, Harsha Vashisht, Sebastien Pereira, Hoi Lam, Marko Velic, Denese Owusu-Afriyie, Katherine Lee, Tolga Bolukbasi, Alicia Parrish, Shawn Lu, Jane Park, Balaji Venkatraman, Alice Talbert, Lambert Rosique, Yuchung Cheng, Andrei Sozanschi, Adam Paszke, Praveen Kumar, Jessica Austin, Lu Li, Khalid Salama, Wooyeol Kim, Nandita Dukkipati, Anthony Baryshnikov, Christos Kaplanis, XiangHai Sheng, Yuri Chervonyi, Caglar Unlu, Diego de Las Casas, Harry Askham, Kathryn Tunyasuvunakool, Felix Gimeno, Siim Poder, Chester Kwak, Matt Miecnikowski, Vahab Mirrokni, Alek Dimitriev, Aaron Parisi, Dangyi Liu, Tomy Tsai, Toby Shevlane, Christina Kouridi, Drew Garmon, Adrian Goedeckemeyer, Adam R. Brown, Anitha Vijayakumar, Ali Elqursh, Sadegh Jazayeri, Jin Huang, Sara Mc Carthy, Jay Hoover, Lucy Kim, Sandeep Kumar, Wei Chen, Courtney Biles, Garrett Bingham, Evan Rosen, Lisa Wang, Qijun Tan, David Engel, Francesco Pongetti, Dario de Cesare, Dongseong Hwang, Lily Yu, Jennifer Pullman, Srini Narayanan, Kyle Levin, Siddharth Gopal, Megan Li, Asaf Aharoni, Trieu Trinh, Jessica Lo, Norman Casagrande, Roopali Vij, Loic Matthey, Bramandia Ramadhana, Austin Matthews, CJ Carey, Matthew Johnson, Kremena Goranova, Rohin Shah, Shereen Ashraf, Kingshuk Dasgupta, Rasmus Larsen, Yicheng Wang, Manish Reddy Vuyyuru, Chong Jiang, Joana Ijazi, Kazuki Osawa, Celine Smith, Ramya Sree Boppana, Taylan Bilal, Yuma Koizumi, Ying Xu, Yasemin Altun, Nir Shabat, Ben Bariach, Alex Korchemniy, Kiam Choo, Olaf Ronneberger, Chimezie

Iwuanyanwu, Shubin Zhao, David Soergel, Cho-Jui Hsieh, Irene Cai, Shariq Iqbal, Martin Sundermeyer, Zhe Chen, Elie Bursztein, Chaitanya Malaviya, Fadi Biadsy, Prakash Shroff, Inderjit Dhillon, Tejasi Latkar, Chris Dyer, Hannah Forbes, Massimo Nicosia, Vitaly Nikolaev, Somer Greene, Marin Georgiev, Pidong Wang, Nina Martin, Hanie Sedghi, John Zhang, Praseem Banzal, Doug Fritz, Vikram Rao, Xuezhi Wang, Jiageng Zhang, Viorica Patraucean, Dayou Du, Igor Mordatch, Ivan Jurin, Lewis Liu, Ayush Dubey, Abhi Mohan, Janek Nowakowski, Vlad-Doru Ion, Nan Wei, Reiko Tojo, Maria Abi Raad, Drew A. Hudson, Vaishakh Keshava, Shubham Agrawal, Kevin Ramirez, Zhichun Wu, Hoang Nguyen, Ji Liu, Madhavi Sewak, Bryce Petrini, DongHyun Choi, Ivan Philips, Ziyue Wang, Ioana Bica, Ankush Garg, Jarek Wilkiewicz, Priyanka Agrawal, Xiaowei Li, Danhao Guo, Emily Xue, Naseer Shaik, Andrew Leach, Sadh MNM Khan, Julia Wiesinger, Sammy Jerome, Abhishek Chakladar, Alek Wenjiao Wang, Tina Ornduff, Folake Abu, Alireza Ghaffarkhah, Marcus Wainwright, Mario Cortes, Frederick Liu, Joshua Maynez, Andreas Terzis, Pouya Samangouei, Riham Mansour, Tomasz Kepa, François-Xavier Aubet, Anton Algymr, Dan Banica, Agoston Weisz, Andras Orban, Alexandre Senges, Ewa Andrejczuk, Mark Geller, Niccolo Dal Santo, Valentin Anklin, Majd Al Merey, Martin Baeuml, Trevor Strohman, Junwen Bai, Slav Petrov, Yonghui Wu, Demis Hassabis, Koray Kavukcuoglu, Jeffrey Dean, and Oriol Vinyals. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context, 2024. URL https://arxiv.org/abs/2403.05530.

xAI. Grok-1, 2024. URL https://github.com/xai-org/grok-1?tab=readme-ov-file.

Databricks. Dbrx, 2024. URL https://www.databricks.com/blog/introducing-dbrx-new-state-art-open-llm.

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. Mixtral of experts, 2024. URL https://arxiv.org/abs/2401.04088.

Snowflake. Arctic, 2024. URL https://www.snowflake.com/en/blog/arctic-open-efficient-foundation-language-models-snowflake/.

DeepSeek-AI, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Yang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jin Chen, Jingyang Yuan, Junjie Qiu, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruizhe Pan, Runxin Xu, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Size Zheng, T. Wang, Tian Pei, Tian Yuan, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Liu, Xin Xie, Xingkai Yu, Xinnan Song, Xinyi Zhou, Xinyu Yang, Xuan Lu, Xuecheng Su, Y. Wu, Y. K. Li, Y. X. Wei, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Zheng, Yichao Zhang, Yiliang Xiong, Yilong Zhao, Ying He, Ying Tang, Yishi Piao, Yixin Dong, Yixuan Tan, Yiyuan Liu, Yongji Wang, Yongqiang Guo, Yuchen Zhu, Yuduan Wang, Yuheng Zou, Yukun Zha, Yunxian Ma, Yuting Yan, Yuxiang You, Yuxuan Liu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhewen Hao, Zhihong Shao, Zhiniu Wen, Zhipeng Xu, Zhongyu Zhang, Zhuoshu Li, Zihan Wang, Zihui Gu, Zilin Li, and Ziwei Xie. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024. URL https://arxiv.org/abs/2405.04434.

Aidan Clark, Diego de las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, George van den Driessche, Eliza Rutherford, Tom Hennigan, Matthew Johnson, Katie Millican, Albin Cassirer, Chris Jones, Elena Buchatskaya, David Budden, Laurent Sifre, Simon Osindero, Oriol Vinyals, Jack Rae, Erich Elsen, Koray Kavukcuoglu, and Karen Simonyan. Unified scaling laws for routed language models, 2022. URL https://arxiv.org/abs/2202.01169.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pages 5547–5569. PMLR, 2022.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, 2022. URL https://arxiv.org/abs/2101.03961.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive Mixtures of Local Experts. *Neural Computation*, 3(1):79–87, 03 1991. ISSN 0899-7667. doi: 10.1162/neco.1991.3.1.79. URL https://doi.org/10.1162/neco.1991.3.1.79.

M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the em algorithm. In Maria Marinaro and Pietro G. Morasso, editors, *ICANN '94*, pages 479–486, London, 1994. Springer London. ISBN 978-1-4471-2097-1.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. St-moe: Designing stable and transferable sparse expert models, 2022. URL https://arxiv.org/abs/2202.08906.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew Dai, Zhifeng Chen, Quoc Le, and James Laudon. Mixture-of-experts with expert choice routing, 2022. URL https://arxiv.org/abs/2202.09368.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation, 2013. URL https://arxiv.org/abs/1308.3432.

Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models, 2024. URL https://arxiv.org/abs/2401.06066.

Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, and Damai Dai. Auxiliary-loss-free load balancing strategy for mixture-of-experts, 2024. URL https://arxiv.org/abs/2408.15664.

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. Phi-3 technical report: A highly capable language model locally on your phone, 2024. URL `https://arxiv.org/abs/2404.14219`.

Liyuan Liu, Young Jin Kim, Shuohang Wang, Chen Liang, Yelong Shen, Hao Cheng, Xiaodong Liu, Masahiro Tanaka, Xiaoxia Wu, Wenxiang Hu, Vishrav Chaudhary, Zeqi Lin, Chenruidong Zhang, Jilong Xue, Hany Awadalla, Jianfeng Gao, and Weizhu Chen. Grin: Gradient-informed moe, 2024. URL `https://arxiv.org/abs/2409.12136`.

Liyuan Liu, Jianfeng Gao, and Weizhu Chen. Sparse backpropagation for moe training, 2023. URL `https://arxiv.org/abs/2310.00811`.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale, 2024. URL `https://arxiv.org/abs/2406.17557`.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, 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, 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, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, 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, 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, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, 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, 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, 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, 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, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaoqing Ellen Tan, 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, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alex Vaughan, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Franco, 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, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, Danny Wyatt, 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, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Firat Ozgenel, Francesco Caggioni, Francisco Guzmán, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, 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, Karthik Prasad, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kun Huang, Kunal Chawla, Kushal Lakhotia, 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, Maria Tsimpoukelli, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, 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, Nikolay Pavlovich Laptev, Ning Dong, Ning Zhang, 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, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Rohan Maheswari, Russ Howes, Ruty Rinott, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, 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, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Kohler, 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, Vítor Albiero, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaofang Wang, Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, and Zhiwei Zhao. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

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. A framework for few-shot language model evaluation, 07 2024. URL https://zenodo.org/records/12608602.

Noam Shazeer. Glu variants improve transformer, 2020. URL https://arxiv.org/abs/2002.05202.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. URL https://arxiv.org/abs/1607.06450.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023. URL https://arxiv.org/abs/2104.09864.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. Deepnet: Scaling transformers to 1,000 layers, 2022. URL https://arxiv.org/abs/2203.00555.

Toan Q. Nguyen and Julian Salazar. Transformers without tears: Improving the normalization of self-attention. 2019. doi: 10.5281/ZENODO.3525484. URL https://zenodo.org/record/3525484.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

Adam Ibrahim, Benjamin Thérien, Kshitij Gupta, Mats L. Richter, Quentin Anthony, Timothée Lesort, Eugene Belilovsky, and Irina Rish. Simple and scalable strategies to continually pre-train large language models, 2024. URL https://arxiv.org/abs/2403.08763.

Alex Andonian, Quentin Anthony, Stella Biderman, Sid Black, Preetham Gali, Leo Gao, Eric Hallahan, Josh Levy-Kramer, Connor Leahy, Lucas Nestler, Kip Parker, Michael Pieler, Jason Phang, Shivanshu Purohit, Hailey Schoelkopf, Dashiell Stander, Tri Songz, Curt Tigges, Benjamin Thérien, Phil Wang, and Samuel Weinbach. GPT-NeoX: Large Scale Autoregressive Language Modeling in PyTorch, 9 2023. URL https://www.github.com/eleutherai/gpt-neox.

Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. Megablocks: Efficient sparse training with mixture-of-experts, 2022. URL https://arxiv.org/abs/2211.15841.

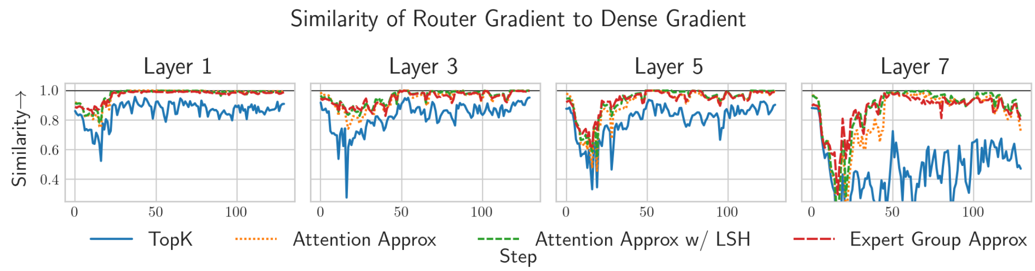


Figure 8: **Accuracy in approximating the dense router gradient for each approach**. We use a model with 8 experts and $K = 2$. We artificially compute the *dense gradient* of the output with respect to the router weights at each step by passing inputs through a dense mixture of experts layer, where all experts are selected. We do this independent of the actual forward pass computation, while using the same set of MoE parameters. The similarity between this dense gradient and the actual gradient propagated to the router indicates how well the router is learning from all experts. We plot this similarity with a standard Top-K router, and with each of our proposed router modifications. Our approaches are much more accurate and stable in approximating the dense router gradient.

A Appendix

In the Appendix we first go over more statistics of the approximations and then then provide more details on our hyperparameters and the implementation.

B Approximation Statistics

Gradient Approximation. We verify that our method is indeed faithfully approximating the dense router gradient i.e. the gradient to the router if all experts were activated. We track the dense gradient by routing to all experts and backpropagating only on the MoE output (independent of the full forward pass). In Fig. 8 we compare this dense gradient to the actual router gradient for each of our approaches and the standard Top-K router. We plot the cosine similarity between the dense gradient and our observed router gradient, where the relatively high similarities our methods achieve indicate we are more accurately approximating the dense gradient. We also observe a major difference between the gradient of the standard Top-K router and our approach in later layers.

The differences in our approaches become clear as we scale the model to become more sparse. We expand to $N = 32$ experts while maintaining $K = 2$ in Fig. 9 and find that it is more difficult to approximate the true dense router gradient. While all of our approaches sufficiently approximate the dense gradient with $N = 8$ experts, the performance gap between them is apparent with $N = 32$. The expert group approximation and LSH attention methods are significantly better than the direct attention method, and this is also consistent with our validation results in Table 2. This is likely due to the heuristics we apply to restrict our approximation to only the most relevant tokens: the expert group approximation requires tokens to have an expert in common, and LSH requires tokens to be similar. Moreover, the gap between our methods and Top-K is wider with 32 experts. We believe that in larger models with even more experts, our method will yield increasingly significant improvements over Top-K routing. We provide further analysis on our approximations in Appendix B.

B.1 Further Gradient Analysis

In Fig. 10 we provide an additional analysis of the gradient norm of our approximation compared to the dense gradient. This logging is also done with $N = 8$ and $K = 2$. The Top-K gradient has significantly lower norm than the dense gradient. Our methods closely

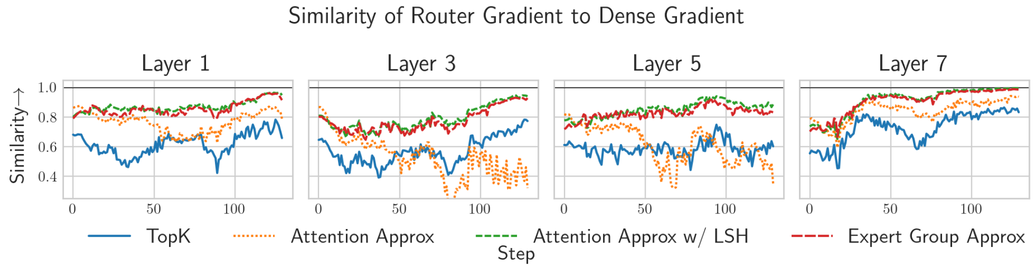


Figure 9: **Dense router gradient approximation accuracy with fine-grained experts**. We implement fine-grained experts as in DeepSeekMoE [DeepSeek-AI et al., 2024] to observe the behavior of our approximation methods across more experts while keeping parameter count fixed. In this example, the model now has 32 experts with $K = 2$. With more experts, it becomes increasingly hard to approximate the dense gradient, and the difference between our methods and the Top-K router is more apparent. Moreover, we can clearly compare the efficacy of each method and see that the attention approximation with LSH is the best. Note the average number of tokens per expert also decreases by a factor of 4 as well, and we would expect even better performance in our approximations by scaling the local batch size per GPU, because we don't communicate tokens across ranks.

approximate the dense gradient norm consistently; replicating both the direction and magnitude suggests that we are sufficiently approximating the dense gradient entirely.

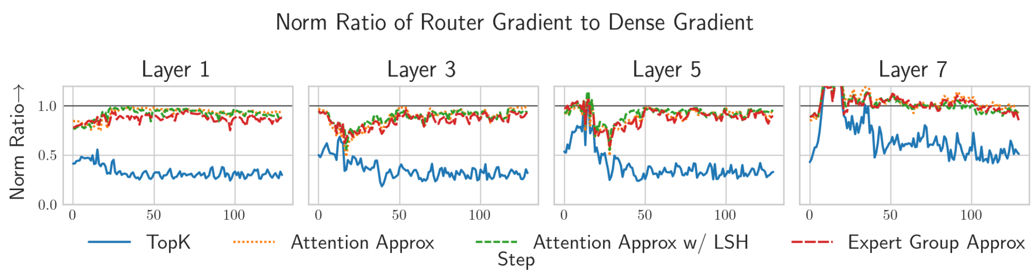


Figure 10: **Comparison of gradient norms relative to the dense gradient**. When computing the dense gradient, we also record its L_2 norm and log the ratio of this to the L_2 norms of the actual router gradients during training. Our methods produce router gradients with approximately the same magnitude. Along with the results showing strong cosine similarity, this suggests that we are almost perfectly approximating the dense gradient.

C Hyperparameters and Implementation Details

Model Configuration. Both MoEs have 24 blocks and a hidden dimension of 1024. The Deepseek MoE has fine-grained experts. Each expert is a bottleneck MLP of shape 1024, 704). The conventional MoE has an expansion factor such that the intermediate size of the MLP is 2816. We use SwiGLU [Shazeer, 2020] MLPs following Llama [Touvron et al., 2023], 16 attention heads with dimension of 64, LayerNorm [Ba et al., 2016] and RoPE [Su et al., 2023]. **Hyperparameters.** We use the initialization from Wang et al. [2022] for residual branch merge layers and the initialization from Nguyen and Salazar [2019] for all other layers. We use the AdamW optimizer [Loshchilov and Hutter, 2019]. We use the modified cosine learning rate schedule from Ibrahim et al. [2024]. We use a sequence length of 2048 and a global batch size of 1024, resulting in a global token batch size of 2^{21}. We set the auxiliary loss [Fedus et al., 2022] to 0.01.

Implementation. We train with the gpt-neox library [Andonian et al., 2023] integrated with Megablocks [Gale et al., 2022]. The TFLOPS vary depending on the method and the number of experts chosen; for simplicity, we do not account for the router or the number of experts

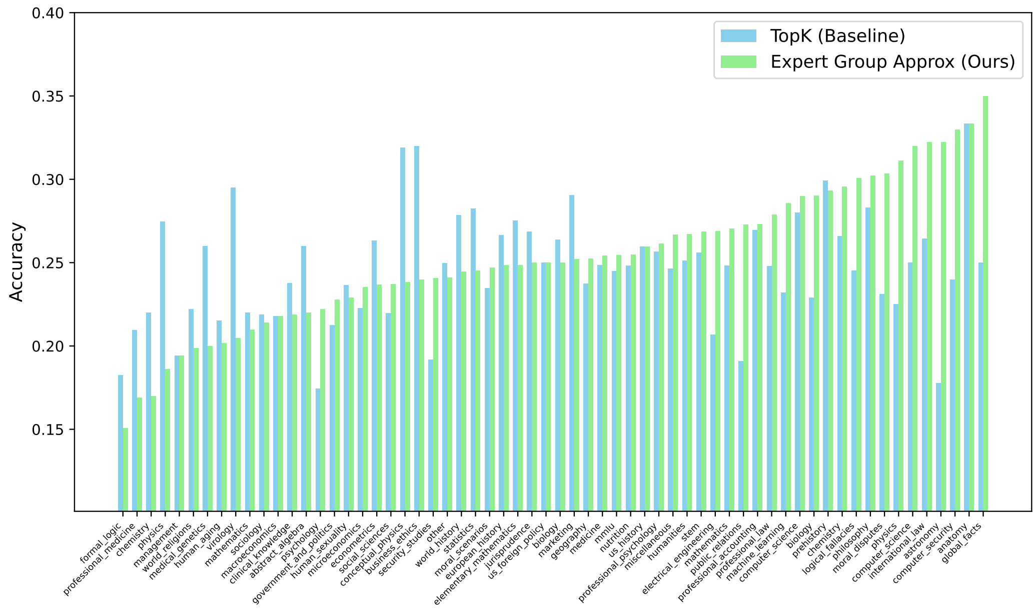


Figure 11: MMLU Scores of our method and the TopK Baseline.

activated when reporting the TFLOPS, so that the number of flops we count in a forward and backward pass is the same as a dense model.

C.1 Ablating Design Choices

We ablate our main design choices and show that our main method does not compromise throughput.

Comparing Different Approximation Methods. We use the Expert Group Approximation method for our main results because it is lightweight, easy to implement, and provides good performance. However, the other two methods we consider also outperform the top-K ($K = 2$) baseline. Indeed, as we showed in Fig. 9, the Attention+LSH method seems to obtain a better approximation of the true dense gradient. The primary reason why we report our main results with Expert Grouping is because the Expert Group Approximation method requires no additional memory overhead. This allows us to use larger microbatches, and therefore there are more tokens on each GPU that we can use for the approximation. In Table 2 we find that even with a microbatchsize $4\times$ smaller than that of the Expert Group method, the Attention+LSH method is competitive.

Routing Method	Microbatchsize	Validation Perplexity
Attention	4	18.72
Attention+LSH	4	18.64
Expert Group	16	18.55
Baseline	16	18.92

Table 2: Comparison of routing methods after training on 20B tokens.

Expert Group Approximation. We consider two variations on the expert group approximation method. As a reminder, in this method we construct a mask of shape $experts, experts$ for each token. The row is the expert that token was routed to, and the column is the expert we want an approximation for. When we take the product of this mask and the router scores, we can weight each row by the probability corresponding to the expert we want an approximation for, or weight each approximation by the probability for the expert we're using the approximation for. The former should give us more "accurate" approximations, because it will prioritize tokens that are more likely to be routed to the expert we want an approximation for. The latter should give us more "viable" approximations, because it weights by closeness to the space we're using the approximation for. We find in Table 3 that neither method improves over the baseline, but we may investigate further.

Table 3: Expert Approx. ablations at 12B tokens.

Routing Method	Validation Perplexity
Expert Group Approx.	20.81
"Accurate"	20.97
"Viable"	21.14

Table 4: Comparing the throughput of the baseline and our method.

Routing Method	TFLOPS
Top-K (K=2)	73.4
Expert Group Approx.	71.7

Method Overhead. We have already outlined the implementation of the Expert Group Approximation method, which only requires materializing two additional tensors of size $experts, experts$ and $experts, micro\_batch\_size \times sequence\_length$. In Table 4 we compare the throughput of our method to the baseline, and find that even with an unoptimized method, we achieve 97.7% of the throughput of the baseline. We anticipate that we can further close this gap by directly modifying the gradient in the backward pass, rather than performing the approximation in the forward pass as we currently do and letting PyTorch's autograd compute the gradient.

Approximation Gradients. Our method is implemented to allow for complete forward and backpropagation of the expert approximations. One other possibility is only sending a gradient signal in the backward pass – that is, the output of the MoE layer does not change but the gradient does incorporate the approximated expert outputs. However, we find that this leads to poor performance and instability. An additional possible restriction is preventing additional gradients to the experts and only updating the router with the dense approximation. This can be done by treating the existing expert outputs as constants in the approximation. However, this too leads to instabilities because the router is learning to incorporate all experts, but the experts themselves are not being updated accordingly. Therefore, it is important to use the dense MoE approximation in the forward pass, and update both the router and experts in tandem.