University of Warsaw
Faculty of Mathematics, Informatics, and Mechanics
Doctoral School of Exact and Natural Sciences

Jan Ludziejewski

Improving Performance of Mixture of Experts Large Language Models

PhD thesis
in Computer Science

Supervisor:
Marek Cygan
Institute of Informatics
University of Warsaw

Warsaw, September 2025
Version 1.0.1

Supervisor's Statement

I confirm that the presented thesis was prepared under my supervision and that it fulfills the requirements for the doctoral degree in the field of Natural Sciences, in the discipline of Computer Science.

Date Supervisor's Signature

Author's Statement

I declare that the presented thesis was prepared by me and that none of its content was obtained through unlawful means.

The thesis has never before been subject to any procedure for obtaining an academic degree. Furthermore, I declare that the presented version of the thesis is identical to the attached electronic version.

Date Author's Signature

Oświadczenie kierującego pracą

Oświadczam, że niniejsza praca została przygotowana pod moim kierunkiem i stwierdzam, że spełnia ona warunki do przedstawienia jej w postępowaniu o nadanie stopnia doktora w dziedzinie nauk ścisłych i przyrodniczych w dyscyplinie informatyka.

Data Podpis kierującego pracą

Oświadczenie autora pracy

Oświadczam, że niniejsza rozprawa doktorska została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem stopnia doktora w innej jednostce. Niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Data Podpis autora pracy

Abstract

This dissertation advances the efficiency and scalability of Large Language Models (LLMs) through systematic exploration and innovation in Mixture of Experts (MoE) architectures. Across five chapters, the work establishes new theoretical frameworks, introduces practical methods, and proposes novel architectures that together contribute to the development of compute- and memory-efficient models at scale.

Chapter 1 investigates the scaling properties of fine-grained MoE models. A new hyperparameter, granularity, is introduced to control the size of individual experts. Incorporating this dimension into scaling laws shows that expert granularity provides substantial efficiency improvements across a wide range of computational budgets. The analysis demonstrates that MoE models not only outperform dense Transformers but also that their relative advantage grows with scale.

Chapter 2 extends this line of research by jointly analyzing compute and memory constraints. Scaling laws are derived and validated that unify dense and MoE models in a single framework, with active parameters, dataset size, and the number of experts as key variables. Contrary to conventional wisdom, the results show that MoE architectures can be more memory-efficient than dense models, providing actionable insights for model design under real-world hardware limitations.

While the first two chapters focus on architectural scaling, Chapter 3 addresses the optimization of large-scale training dynamics. Relative Learning Rate Schedules (RLRS) are introduced, assigning distinct learning rates across different model components. This method yields training speedups of up to 23% and scales robustly: hyperparameters tuned on smaller models can be effectively reused on models up to 27 times larger. RLRS thus offers a simple, transferable, and computationally efficient solution for accelerating the training of both dense and expert-based models.

Chapter 4 explores alternative formulations of MoE by proposing the Mixture of Tokens (MoT) architecture. Unlike traditional sparse or discontinuous MoE models, MoT provides a continuous mechanism for assigning token mixtures to experts. The design maintains compatibility with autoregressive generation while achieving training speedups over dense Transformers and matching the performance of state-of-the-art MoE approaches.

Finally, Chapter 5 extends the principles of MoE to non-Transformer architectures. Focusing on State Space Models (SSMs), and in particular the Mamba architecture, the MoE-Mamba hybrid model is developed. This architecture combines the advantages of SSMs with the efficiency of MoE, reaching the same performance as Mamba in 2.35× fewer training steps while retaining the favorable inference characteristics of SSMs. The results demonstrate that the benefits of MoE extend beyond Transformer-based systems.

Taken together, the contributions of this dissertation outline a path for the next generation of large-scale models. By developing new scaling laws, optimization strategies, and hybrid architectures, the work shows that MoE-based designs are not only practical but increasingly central to building efficient and powerful LLMs.

Keywords

Deep Learning, Neural Networks, Large Language Models, Transformer, Mixture of Experts, Scaling Laws

Tytuł pracy w języku polskim

Usprawnianie dużych modeli językowych używających mieszaniny ekspertów

Streszczenie w języku polskim

Ta rozprawa doktorska skupia się na zwiększaniu efektywności i skalowalności dużych modeli językowych (LLM) poprzez usprawnianie architektur typu Mixture of Experts (MoE). W pięciu rozdziałach przedstawiono nowe prawa teoretyczne, metody praktyczne oraz opracowano nowe architektury, które łącznie przyczyniają się do rozwoju modeli obliczeniowo i pamięciowo bardziej wydajnych w dużej skali.

Rozdział 1 analizuje właściwości skalowania modeli MoE. Wprowadzony zostaje nowy hiperparametr – ziarnistość (granularity), który pozwala kontrolować rozmiar poszczególnych ekspertów. Uwzględnienie tego wymiaru w prawach skalowania pokazuje, że odpowiednia ziarnistość ekspertów zapewnia znaczną poprawę efektywności w szerokim zakresie budżetów obliczeniowych. Analiza wskazuje, że modele MoE przewyższają nie tylko gęste Transformery, lecz także że ich przewaga rośnie wraz ze skalą.

Rozdział 2 rozwija podobną linię badań, analizując jednocześnie ograniczenia obliczeniowe i pamięciowe. Zawiera wyprowadzone i zweryfikowane prawa skalowania, które łączą modele gęste i MoE we wspólnych ramach, z uwzględnieniem liczby aktywnych parametrów, rozmiaru zbioru danych oraz liczby ekspertów. Wbrew powszechnym opiniom, wyniki pokazują, że architektury MoE mogą być bardziej pamięciooszczędne niż modele gęste, co dostarcza praktycznych wskazówek przy projektowaniu modeli w realnych warunkach sprzętowych.

Podczas gdy pierwsze dwa rozdziały koncentrują się na skalowaniu architektury, Rozdział 3 zajmuje się optymalizacją dynamiki uczenia na dużą skalę. Wprowadzono tu względne harmonogramy uczenia (Relative Learning Rate Schedules, RLRS), które przypisują różne tempo uczenia poszczególnym komponentom modelu. Metoda ta przyspiesza trening nawet o 23% i wykazuje dużą skalowalność: hiperparametry dostrojone na mniejszych modelach mogą być skutecznie przenoszone na modele do 27 razy większe. RLRS stanowi więc prostą, przenośną i efektywną obliczeniowo metodę przyspieszania treningu zarówno modeli gęstych, jak i eksperckich.

Rozdział 4 bada alternatywne podejścia do MoE, proponując architekturę Mixture of Tokens (MoT). W odróżnieniu od tradycyjnych rzadkich czy nieciągłych modeli MoE, MoT zapewnia ciągły mechanizm przypisywania mieszanin tokenów do ekspertów. Architektura ta pozostaje zgodna z autoregresyjnym uczeniem i generowaniem, jednocześnie oferując przyspieszenie treningu względem gęstych Transformerów i osiągając wyniki porównywalne z najlepszymi współczesnymi modelami MoE.

Finalnie, Rozdział 5 rozszerza zasady MoE na architektury inne niż Transformer. Skupiając się na modelach typu State Space Models (SSM), a w szczególności na architekturze Mamba, opracowanany zostaje hybrydowy model MoE-Mamba. Architektura ta łączy zalety SSM z efektywnością MoE, osiągając wydajność porównywalną z Mamba w 2,35 razy mniejszej liczbie kroków treningowych, przy jednoczesnym zachowaniu korzystnych właściwości inferencyjnych SSM. Wyniki te pokazują, że zalety MoE wykraczają poza systemy oparte na Transformerach.

Podsumowując, wkład tej rozprawy doktorskiej wyznacza kierunek rozwoju kolejnej generacji dużych modeli językowych. Dzięki opracowaniu nowych praw skalowania,

strategii optymalizacji oraz architektur hybrydowych, praca ta wskazuje, że rozwiązania oparte na MoE są nie tylko praktyczne, lecz także stają się kluczowym elementem w budowie wydajnych modeli językowych.

Słowa kluczowe

głębokie uczenie, sieci neuronowe, duże modele językowe, transformer, mieszanina ekspertów, prawa skalowania

Contents

Chapter 1

Introduction

1.1 Large Language Models

Large Language Models (LLMs) have recently emerged as general-purpose models capable of solving a wide range of tasks (Brown et al., 2020; Chowdhery et al., 2022; Touvron et al., 2023a; OpenAI et al., 2023; Anil et al., 2023a). Since its introduction, the Transformer architecture has become the de facto standard for large-scale sequence modeling (Vaswani et al., 2023). The dominant trend in recent years has been to increase model size and dataset scale, with the observation that performance improves consistently as we scale. This has been validated empirically across various domains, including language, code, and multi-modal tasks (Kaplan et al., 2020; Chowdhery et al., 2022; Hoffmann et al., 2022). Models like GPT-series and PaLM were one of the first to demonstrate that scaling up continues to unlock new capabilities (Brown et al., 2020; Chowdhery et al., 2022). However, increasing scale comes with a price of never-ending need for more compute and increasing costs (Hoffmann et al., 2022). While the skeleton of the transformer stays the same, with time and scale, optimal hyperparameters change and various parts of it are getting improved and exchanged for more compute-efficient counterparts (Clark et al., 2022; Zhai et al., 2022). In this work, we focus on improving methods that scale, and on ensuring that all choices remain optimal on the way up.

1.2 Scaling Laws

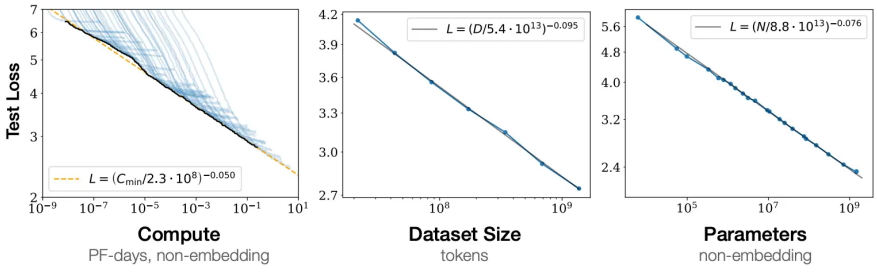


Figure 1.1: (Figure from Kaplan et al. (2020))

A key insight that enabled recent progress in LLMs is the discovery of empirical scaling laws. Introduced by (Hestness et al., 2017), (Kaplan et al., 2020) proposed the

first such formulation for dense Transformers, and (Hoffmann et al., 2022) later refined it into what is now known as Chinchilla scaling laws, which define the optimal balance between model and data size for a given compute budget. These laws were originally meant to allow researchers to design training runs that maximize performance under computational constraints, solving important trade-offs between model size and training length. Scaling formulas related the loss achieved by a model to its number of parameters and the number of training tokens, showing a predictable, power-law decay in loss as we increase compute.

The most popular formula called *Chinchilla scaling law*, is described in (Hoffmann et al., 2022) as a relationship between model size N and dataset D:

$$\mathcal{L}(N, D) = c + \frac{a}{N^{\alpha}} + \frac{b}{D^{\beta}}. \tag{1.1}$$

The power-law formula is composed of three distinct terms that characterize the intrinsic entropy of data, constraints of the model, and limitations in the training data. The term c represents the minimum possible error intrinsic to the data. The remaining two terms are suboptimality terms, which address the limitations in function representation owing to the size of the model and in data signified by the number of tokens. In the limit, with infinite data and model size, the loss is reduced to c.

After constraining this equation by a formula describing utilized compute, we can find an optimal *token-to-param* ratio, but also see a simple and predictable power law trend between cost and final performance. This last property of predictability of gains quickly became as important, providing a foundation and incentive for the scaling race.

1.3 Need For Efficiency

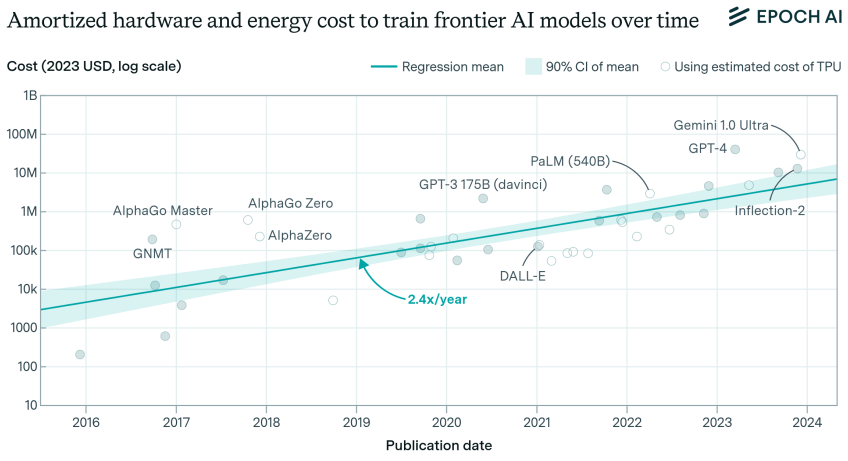


Figure 1.2: (Figure from **?**)

As models continue to grow, so do their computational and environmental costs. Training dense LLMs requires millions of GPU-hours and leads to high energy consumption (Strubell et al., 2019; Faiz et al., 2024). These factors make it necessary to consider

not only how to scale but also how to do so efficiently. There is a growing need for architectures and training strategies that provide better performance per unit of compute.

One other consequence of predictable scaling of a given model class is that in pre-training, work on capabilities can be perceived as equivalent to optimizing efficiency. If we improve any method to always achieve the same performance as our baseline using only half of its compute, we can upscale to get a model that is two times more capable (Hoffmann et al., 2022; Sardana et al., 2024a). This is not only a practical proxy goal, but an important paradigm, influencing the spirit in which this work was written.

This also connects with meta text "The Bitter Lesson" (Sutton, 2019b), which in our view was not only visionary for its time but summarizes the core assumptions under the recent LLM surge. The aforementioned work analyzes how different machine learning methods passed the test of time, arguing that historically specialized approaches relying on expert knowledge and inductive biases have often been forgotten and replaced by more generic algorithms utilizing much higher compute, allowing machines to learn by themselves. This however does not mean that there is no place for novelty—on the contrary, for us this forms a new research paradigm, centered around using available compute efficiently and focusing on methods that scale, which is far from being properly understood.is far from beeing properly understood.

1.4 Mixture of Experts

Mixture of Experts (MoE) models are one such alternative. In MoEs, only a small subset of the total parameters is activated for each token, allowing the model to have a much larger total parameter count while keeping the computational cost mostly unchanged. Prior works such as GShard, Switch Transformer, and Mixtral have demonstrated that MoEs can achieve performance comparable to dense models at greatly reduced computational cost (Lepikhin et al., 2020; Fedus et al., 2022; Jiang et al., 2024). These models replace Feed Forward layers in Transformers with conditionally activated approximations of a much bigger one, relying on a routing mechanism to select which subset of neurons should be used for each token (Shazeer et al., 2017; Fedus et al., 2022).

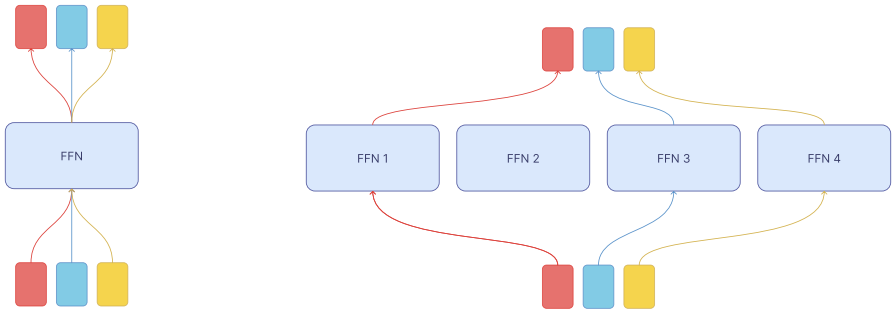


Figure 1.3: **(Left).** Diagram of a standard Feed-Forward layer featured in the Transformer architecture: each token is processed with the same MLP independently of other tokens. **(Right).** Diagram of a Mixture of Experts layer, where each token decides which expert to choose. In this way, different experts process a different number of tokens.

As a result, we only pay FLOPs for parameters we activate. This relies on an observation that normally only a small part of all neurons in Feed Forward are activated,

so if the router would be able to roughly guess which ones do, we can save on computing them (Shazeer, 2020b; Puigcerver et al., 2024). The promise of MoEs lies in the idea that we can build models with much higher capacity without paying the full computational price, aligning with aforementioned paradigms.

1.4.1 Challenges and Skepticism

Despite this promise, the scaling behavior of MoEs remains poorly understood. The only work existing before our analyses, (Clark et al., 2022), suggests that MoEs only outperform dense models up to a certain scale, after which dense models become more efficient. This conclusion is based on the concept of effective parameter count and assumes fixed training data size. However, this perspective contradicts the success of large MoE models deployed in practice. It raises the question of whether the observed limitations are fundamental or rather a result of insufficient understanding of how to scale sparse models.

1.5 Towards Generalized Scaling

While scaling laws have been validated extensively for dense models, it remains unclear whether and how they apply to other architectures. In particular, sparse models like MoEs activate only a subset of parameters per token, potentially breaking the assumptions behind standard scaling laws. Moreover, MoEs introduce new scaling dimensions such as the number of experts, the number of active experts per token, and the internal size of each expert. These factors, including granularity and expansion rate, can affect performance in non-trivial ways (Kaplan et al., 2020; Clark et al., 2022; Zhai et al., 2022; Puigcerver et al., 2024). The compute optimal token-to-parameter ratio, for example, may have different optimal values for MoEs than for dense models and it is not even defined if we should use active or total parameters in this ratio. These dimensions are specific to sparse models and are absent in standard dense architectures, requiring separate treatment and defining new optimality frontiers. Despite MoEs being a core component of many recent large-scale LLMs, the literature lacks comprehensive scaling laws that capture these unique properties.

Additionally, these laws apply not only to model shape but also to optimization hyperparameters. For example, it is widely accepted that with increasing model size, learning rate has to be lowered proportionally (Hoffmann et al., 2022; Kaplan et al., 2020). Other works suggest that because of the instability of MoEs, they also require lower learning rates, however, these interactions remain underexplored in the existing literature (Fedus et al., 2022; Zoph et al., 2022a).

This thesis investigates what is the most efficient Transformer variant, and how to scale it under various constraints, introducing new theoretical frameworks and validating them with large-scale experiments. We aim to provide practical guidelines for designing MoE-based LLMs that maintain efficiency at scale.

1.6 Content

Each chapter describes a different approach to improving the Mixture-of-Experts architecture. **P1**–**P3** introduce new methods for analyzing model hyperparameters: **P1**–**P2** utilize scaling laws to predict model performance and guide architecture design, while **P3**

focuses on tuning decoupled per-module optimization hyperparameters. **P1** introduces new dimension in Mixture of experts scaling called granularity, that greatly increases efficiency of these models, while **P2** is the first scaling law to extend Chinchilla to Mixture of Experts models, enabling analysis of how Transformer scaling properties change with sparsity. Chapters **P4**–**P5** propose architectural modifications to enhance performance and stability: **P4** achieves this by mixing token representations to ensure the continuity of gradients for routed experts, and in **P5** by combining state-space models with sparse computation.

In each chapter we introduce new definitions, which we then assume as known in subsequent sections.

1.7 Context and Timing

As this work unfolded over several years, the surrounding landscape changed significantly. At times, a chapter might assert that certain practices are rare or unexplored, such as routing granularity or MoE scaling heuristics, which by now have become core parts of mainstream models. These shifts are a natural outcome of a fast-moving field, and ideally, they signal that some of the questions raised here were timely and pointed in the right direction.

1.8 List of Manuscripts

My thesis comprises of five manuscripts, from **P1** to **P5**:

P1: Scaling Laws for Fine-Grained Mixture of Experts

The work is detailed in Chapter 2.

Authors: **J. Ludziejewski**, J. Krajewski, K. Adamczewski, M. Pióro, M. Krutul, S. Antoniak, K. Ciebiera, K. Król, T. Odrzygóźdź, P. Sankowski, M. Cygan, S. Jaszczur

Published at **International Conference on Machine Learning (ICML) 2024**, CORE Rank: A*, MNiSW: 200

I estimate my contribution to be at 23%.

P2: Joint MoE Scaling Laws: Mixture of Experts Can Be Memory Efficient

The work is detailed in Chapter 3.

Authors: **J. Ludziejewski**, M. Pióro, J. Krajewski, M. Krutul, M. Stefaniak, M. Cygan, K. Adamczewski, P. Miłoś, S. Jaszczur

Published at **International Conference on Machine Learning (ICML) 2025**, CORE Rank: A*, MNiSW: 200

I estimate my contribution to be at 40%.

P3: Different Rates for Different Weights: Decoupled Relative Learning Rate Schedules

The work is detailed in Chapter 4.

Authors: **J. Ludziejewski**, J. Małaśnicki, M. Pióro, M. Krutul, K. Ciebiera, M. Stefaniak, J. Krajewski, P. Sankowski, M. Cygan, K. Adamczewski, S. Jaszczur

I estimate my contribution to be at 40%.

P4: Mixture of Tokens: Efficient LLMs through Cross-Example Aggregation

The work is detailed in Chapter 5.

Authors: S. Antoniak, M. Krutul, M. Pióro, J. Krajewski, J. Ludziejewski, K. Ciebiera, K. Król, T. Odrzygóźdź, M. Cygan, S. Jaszczur

Accepted at **Conference on Neural Information Processing Systems (NeurIPS) 2024**, CORE Rank: A*, MNiSW: 200

I estimate my contribution to be at 10%.

P5: MoE-Mamba: Efficient Selective State Space Models with Mixture of Experts

The work is detailed in Chapter 6.

Authors: M. Pióro, K. Ciebiera, K. Król, J. Ludziejewski, M. Krutul, J. Krajewski, S. Antoniak, P. Miłoś, M. Cygan, S. Jaszczur

I estimate my contribution to be at 15%.

1.9 Reproducibility

The code for methods described in all Chapters is available in our public repository at `https://github.com/llm-random/llm-random`.

Chapter 2

Scaling Laws for Fine-Grained Mixture of Experts

2.1 Introduction

In the context of the current trend of increasing budgets for training models, a question arises: will MoE models continue to be attractive in the future? This is an important issue, as results from other studies Clark et al. (2022) suggest that the traditional dense models may outperform MoE as the size of models increases.

In this chapter, we argue that previous claims lose their validity when we relax certain implicit assumptions regarding the training process present in previous research Clark et al. (2022). In particular, we refer to the fixed training duration and the constant size of experts in MoE models.

Our results suggest that a compute-optimal MoE model trained with a budget of 10^{20} FLOPs will achieve the same quality as a dense Transformer trained with a $20\times$ greater computing budget, with the compute savings rising steadily, exceeding $40\times$ when budget of 10^{25} FLOPs is surpassed (see Figure 2.1).

Our main contributions are:

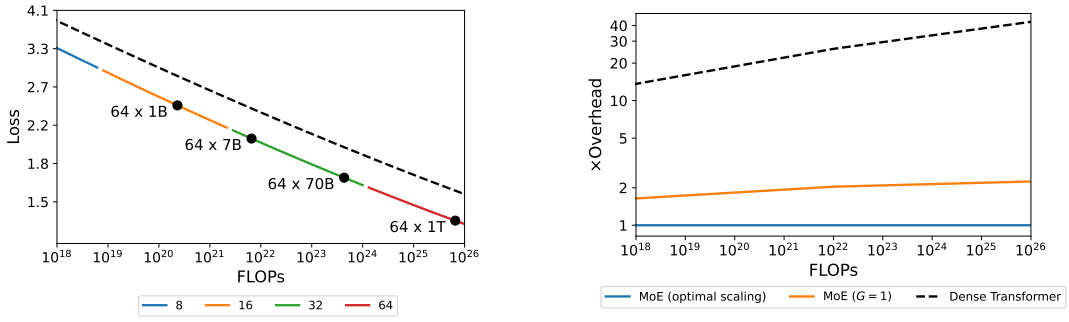
1. Introducing a new hyperparameter - granularity. Adjusting this parameter allows us to determine the optimal size of experts in MoE models, which translates into increased efficiency (see Figure 2.1b and Figure 2.5).

2. Deriving new scaling laws for MoE models that incorporate variable training duration, the number of parameters, and granularity. Such scaling laws allow us to calculate optimal training hyperparameters for MoE models.

3. Demonstrating that, with optimal settings, MoE models can always outperform traditional Transformers at any computing budget. This is a conclusion contrary to the results from Clark et al. (2022), see Section 2.6.3.

2.2 Related Work

Mixture of Experts. In the context of language modeling, MoE was first introduced by Shazeer et al. (2017) as a sparsely gated layer between stacked blocks of LSTM Hochreiter & Schmidhuber (1997). A similar technique was proposed in the context of



(a) **(b)**

Figure 2.1: Mixture-of-Experts can be *always* considered more efficient than dense Transformers, regardless of the model size. **(a)** Compute Optimal scaling curves for granular models and standard Transformers. The dashed line represents a dense Transformer. Colors denote optimal granularity for the given FLOPs training budget. **(b)** Relative number of FLOPs needed to train Transformer and Vanilla MoE (MoE with $G = 1$) to achieve the performance of MoE with compute optimal G.

Transformers by Shazeer et al. (2018) and Lepikhin et al. (2020). Fedus et al. (2022) proposed to route each input to only a single expert and designed a modified initialization scheme to reduce training instability.

Numerous studies have proposed to modify the original routing method. Lewis et al. (2021) used a linear assignment algorithm to postprocess token-expert mappings and ensure even expert selections. Roller et al. (2021) suggested another approach involving deterministic hash functions. Zhou et al. (2022) proposed expert choice routing, eliminating the need for additional load balancing losses. Puigcerver et al. (2023a) designed a fully-differentiable Soft MoE architecture. Concurrently to our work, Dai et al. (2024) proposed to modify the MoE layer by segmenting experts into smaller ones and adding shared experts to the architecture. Independently, Liu et al. (2023) suggested a unified view of sparse feed-forward layers, considering, in particular, varying the size of memory blocks. Both approaches can be interpreted as modifying granularity. However, we offer a comprehensive comparison of the relationship between training hyperparameters and derive principled selection criteria.

Scaling Laws. Scaling laws are empirically derived equations relating the loss of a model with variables such as the number of parameters, training samples, or the computational budget. In the case of dense Transformers, scaling laws were first studied by Kaplan et al. (2020), who observed power law relationships between the final model perplexity and model and dataset size. This work was extended by Hoffmann et al. (2022), by considering variable cosine cycle lengths, and formulating a modified functional form of the scaling equation.

Scaling laws have also been proposed for other architectures and training scenarios. Henighan et al. (2020) studied autoregressive modeling across various modalities, while Ghorbani et al. (2021) considered machine translation. Frantar et al. (2023) explored the impact of pruning on vision and language Transformers, deriving optimal sparsity for a given compute budget. Clark et al. (2022) studied the scaling of MoE when changing

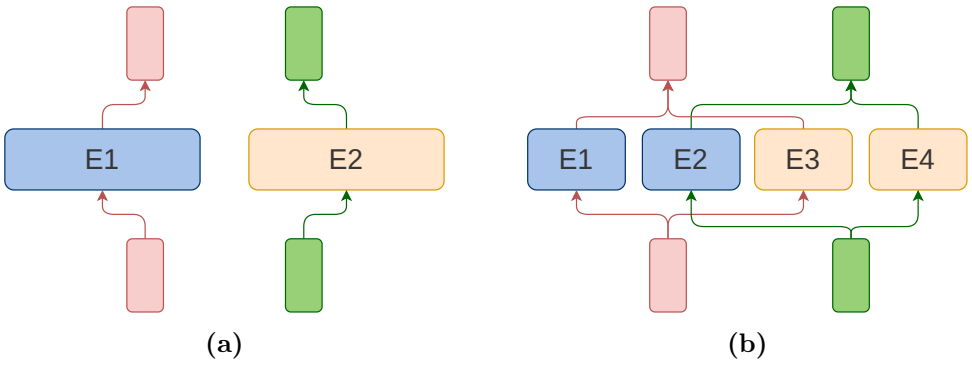


Figure 2.2: **(a)** Standard MoE layer with $G = 1$ **(b)** Corresponding MoE layer with $G = 2$. Each of the original experts is split into two granular ones. The split occurs in the hidden dimension of an expert. Increasing G allows for a more precise mapping between experts and tokens. Since for granularity G, the token is routed to G granular experts, the number of parameters activated per token is the same in both cases.

model size and number of experts on a fixed dataset, concluding that routed models are more efficient only until a certain model size. In this work, we challenge that claim by considering a variable, optimal dataset size for both model families (see Section 2.6.3).

2.3 Background

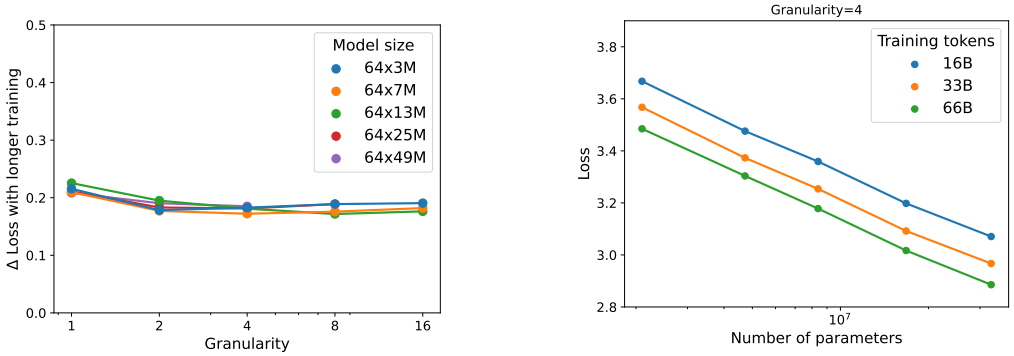
2.3.1 Model Architecture

Transformer. A standard decoder-only Transformer Radford et al. (2018a,b); Kaplan et al. (2020); Brown et al. (2020) consists of an embedding layer, a stack of alternating attention and feed-forward layers, and an unembedding layer. In the model, each input token is converted by the embedding layer into a vector of size d_{model}, the dimension maintained across all the layers in the residual stream.

The feed-forward component consists of two linear transformations and a nonlinearity ϕ in between. It can be described as $\text{FFN}(x) = \phi(xW_1 + b_1)W_2 + b_2$, with W_1 mapping from d_{model} to d_{ff}, and W_2 back to the original d_{model}. It is standard Radford et al. (2018a); Rae et al. (2022); Touvron et al. (2023a); Jiang et al. (2023) to set the hidden dimension as $d_{\text{ff}} = 4 \cdot d_{\text{model}}$.

Feed-forward layers contain the majority of Transformer parameters and require the biggest computational budget counted in terms of FLOPs. Subsequently, they are the main focus of the Mixture of Experts models considered in this work.

Mixture of Experts. The core idea behind MoE in Transformers is to replace the feed-forward layer with a set of *experts*. The size of each expert is typically Fedus et al. (2022); Zhou et al. (2022, 2023); Jiang et al. (2024) set to mirror the original dimensions of the layer, with the hidden expert dimension d_{expert} equal to d_{ff}. Therefore, the total number of parameters in MoE scales linearly with the number of experts. However, the computational cost remains approximately constant as each input is routed and then processed by a subset of experts.



(a) **(b)**

Figure 2.3: **(a)** The difference in the loss between training for 16B and 65B tokens for all model sizes and granularity values. The model size is reported as the expansion rate and the number of active parameters. **(b)** The impact of varying the number of parameters N on the loss for fixed granularity G.

2.3.2 Scaling Law for Mixture of Experts

Large Transformer-based models are known to approximately obey the power-law relationship between final loss $\mathcal{L}$, model size N, and number of training tokens D. For MoE Transformer-based models, Clark et al. (2022) formulated the final loss for a constant dataset size D of 130B tokens, allowing for variations in the number of experts E, as:

$$\mathcal{L}(N, E) = \left(\frac{10^{d/a}}{N}\right)^{a} \left(\frac{1}{E}\right)^{b+c\ln N}. \tag{2.1}$$

However, this result has a notable limitation as it can be applied only to the original dataset size. The scalability and effectiveness are constrained in this scenario because it is crucial to align the number of training samples with the available computational resources for optimal use. As per Kaplan et al. (2020) and Hoffmann et al. (2022), maintaining a constant dataset size while scaling up the neural network size leads to undertraining, resulting in a model that does not perform to its full potential.

2.4 Granularity

As described in Section 2.3, in the standard setting, the inner dimension of each expert network $d_{\text{expert}} = d_{\text{ff}}$, the same size as the feed-forward layer of the base model.

In this work, we suggest an alternative approach where the hidden dimension of the expert is not necessarily set to mirror that of the standard feed-forward layer. Instead, it can be adjusted to a value that is the most effective. This approach allows the configuration of MoE to be articulated in terms of two key hyperparameters: *granularity* (G) and *expansion rate* (R). In the following parts of this work, we will also use the term *active* parameters to refer to the non-embedding parameters used to produce output for a single token, except routing. The number of active parameters is denoted as N_{act}.

Let d_{expert} be the hidden dimension of a single expert. Granularity is defined as

$$G = \frac{d_{\text{ff}}}{d_{\text{expert}}}.$$

In other words, granularity denotes the multiplier factor for the change in the size of an expert from the original standard model, defined as $G = 1$. In this work, we investigate $G > 1$ where experts are smaller than in the standard layer.

Note that increasing granularity does not affect the number of active parameters since, as G increases, the number of experts that process the token grows proportionally to G. That is, for granularity G, a token is routed to G fine-grained experts, keeping the number of active parameters constant. See Fig. 2.2 for visualization.

We then define the *expansion rate*, which describes the increase in the number of parameters from a standard transformer layer to an MoE layer. Given that, N_{MoE} and N_{ff} denote the total number of parameters in an MoE layer excluding routing and the standard feed-forward layer, respectively. The expansion rate R is then defined as

$$R = \frac{N_{\text{MoE}}}{N_{\text{ff}}}.$$

Expansion rate can also be seen as the total number of parameters in an MoE layer compared to active parameters. The relationship between the number of experts (N_{expert}), the expansion rate, and the granularity is described by the following equation:

$$N_{\text{expert}} = G \cdot R. \tag{2.2}$$

For non-granular models, i.e., $G = 1$, the expansion rate is equal to the number of experts.

Intuitively, increasing granularity for a given expansion rate gives the model more flexibility in mapping datapoints to experts, potentially improving performance. We incorporate the notion of granularity into our scaling laws in Section 2.5. The discussion about practical tradeoffs in changing this parameter is given in Section 2.6.

2.5 Scaling Laws

Granularity determines changes in the architecture of MoE. In this section, we answer a central question of this work: whether the granular MoE models follow scaling laws and, if so, how granularity affects them. Thus, we aim to derive a parametric scaling law for predicting the final loss value $\mathcal{L}$ based on granularity G, total number of non-embedding parameters N, and number of training tokens D.

We run over 100 experiments on the decoder-only Transformer architecture, with each feed-forward component replaced by a Mixture of Experts layer. Those experiments involve training models with sizes ranging from 129M to 3.7B parameters across different training durations, from 16B to 130B tokens. We consider logarithmically spaced values of granularity between 1 and 16. To constrain the search space, $R = 64$ is fixed, following the recommendations of Clark et al. (2022). In addition, we also run experiments with dense Transformers to compare their performance with MoE. The details of all architectures, the training procedure, and hyperparameter choices are described in detail in section 2.7.

In the subsequent part of this work, we will use the notation $R \times N_{\text{act}}$ to describe a MoE model with N_{act} active parameters and expansion rate R.

2.5.1 Power Law With Respect to Granularity

We first answer the question of whether granular models follow the scaling laws. In Figure 2.4, we can notice that increasing granularity results in a lower loss. The returns

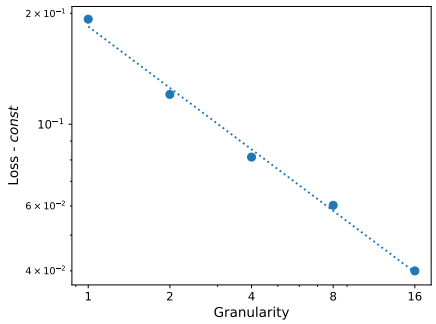


Figure 2.4: We plot the effect of G on $\mathcal{L}_{N,D}(G)$ for constant N and D. Both axes are in the log-scale. The results suggest the linear relationship between $\ln(G)$ and $\ln(\mathcal{L} - c)$. The given values are $N = 64 \times 25M$, $D = 16B$, $const = 3.12$. The plots for additional values of N and D can be found in Figure 2.7.

follow approximately an exponential pattern, converging to a positive constant. The empirical relationship given by Figure 2.4 suggests, for given N and D, the following power-law dependence of loss $\mathcal{L}_{D,G}$ on a varying granularity G, parametrized by $g_{N,D}$, $\gamma_{N,D}$, and $h_{N,D}$,

$$\mathcal{L}_{N,D}(G) = \frac{g_{N,D}}{G^{\gamma_{N,D}}} + h_{N,D}. \tag{2.3}$$

2.5.2 Scaling the Model and Dataset Size

As outlined in Section 2.3.2, the power-law given by Eq. (1.1) consists of three terms that describe inherent data entropy and limitations in function representation and data. This derivation is independent of the architecture. In particular, the Eq. (1.1) also holds for constant granularity. Empirically, we observe a power law relationship in N and D analogous to that in dense models (see also Figure 1 in Kaplan et al. (2020)), as depicted in Figure 2.3 for a fixed value of granularity. Furthermore, the validity of this functional form is verified by fit in Section 2.5.4.

Since we know that separate scaling laws are valid for given granularities, in the general form, the parameters in Eq. (1.1) can be dependent on the model's granularity:

$$\mathcal{L}_G(N, D) = c_G + \frac{a_G}{N^{\alpha_G}} + \frac{b_G}{D^{\beta_G}}. \tag{2.4}$$

2.5.3 The Form of the Joint Scaling Law

Following the above observation that models with constant granularity obey Chinchilla scaling laws given by Eq. (1.1), the key question arises as to how the general notion of granularity G can be incorporated into the joint scaling law. Moreover, the scaling law formula from Eq. (2.4) for constant N and D has to be representable by Eq. (2.3). This is because the former is a more general equation, encompassing shared hyper-parameters across all N, D, and G. It is anticipated to align with the latter, consisting of distinct power laws, each with specific parameters for different N and D values. Consequently, the objective is to identify a function that fulfills these criteria.

$$\mathcal{L}(N, D, G) = \mathcal{L}_{N,D}(G) = \mathcal{L}_G(N, D) \tag{2.5}$$
$$= \frac{g_{N,D}}{G^{\gamma_{N,D}}} + h_{N,D} = c_G + \frac{a_G}{N^{\alpha_G}} + \frac{b_G}{D^{\beta_G}}$$

In the subsequent sections, we aim to determine which of these parameters remain independent of G and identify their functional form. Additionally, we present some rationale for the structure of our formula.

Lower Bound. Consider the limit of Eq. (2.4) for N and D growing to infinity:

$$\lim_{\substack{N\to\infty \\ D\to\infty}} \mathcal{L}(N, D, G) = c_G. \tag{2.6}$$

with the constant term c_G dependent on granularity.

This dependence is contradictory to the fact that the term captures the inherent entropy of the dataset, as also defined in Hoffmann et al. (2022) appendix D.2 *'the minimal loss achievable for next-token prediction on the full distribution P, a.k.a the "entropy of natural text."'*.

The lower bound of the achievable loss for training bigger models on more samples should not depend on the architecture since it is a function of a dataset, not of a model. Therefore, the parameter $c_G = c$ is constant for all granularities.

Granularity and Number of Tokens D. As seen in Figure 2.3(a), the benefit of training a model on a larger dataset is almost the same for each granularity value. This suggests that there is no interaction between D and G. Therefore, we can assume that

$$\frac{b_G}{D^{\beta_G}} = \frac{b}{D^{\beta}}. \tag{2.7}$$

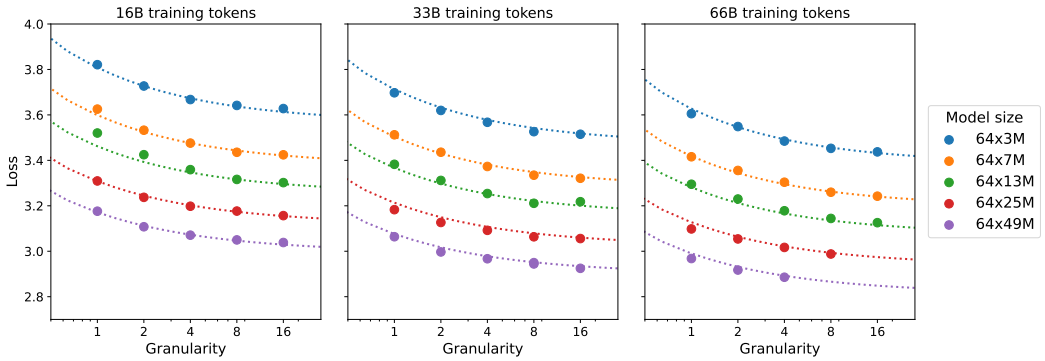


Figure 2.5: We present the fit of the scaling law compared to experimental results.

Granularity and Model Size N. We consider α to be a constant that describes how the function scales with N. In this work, we assume polynomial functional forms that rule out the potential dependency of α on G given the form of Eq. 2.3. Therefore, the only element dependent on G is a_G:

$$\mathcal{L}(N, D, G) = c + \left(\frac{g}{G^{\gamma}} + a\right)\frac{1}{N^{\alpha}} + \frac{b}{D^{\beta}}. \tag{2.8}$$

Finally, one could consider omitting the constant a in the equation above, and it would still reduce to (2.3) for constant N and D. However, this would mean that a model

with infinite granularity and a small number of active parameters can achieve the perfect perplexity of the lower bound. We think that MoE sparse model should not exceed the performance of its dense counterpart matched by a total number of parameters and with all of them activated. This means that constant a can act as a marginal improvement from granularity.

2.5.4 Fitting the Parametric Scaling Law

Subsequently, we fit parameters in (2.8) to describe the scaling of MoE. For comparison, we also perform fitting for dense transformer given by (1.1). Similarly to Hoffmann et al. (2022), we use Huber loss Huber (1964), with $\delta = 0.1$. The optimization is performed using the BFGS algorithm. We include a weight decay of $5e-4$ to enhance generalization. We start with fitting parameters in (2.8) and then find architecture-dependent coefficients α, β, a and b in (1.1). The values are presented in Table 2.1. We depict the fit of the equation in Figure 2.5. We generally observe a good fit, with RMSE = 0.015.

Model	a	α	b	β	g	γ	c
MoE	18.1	0.115	30.8	0.147	2.1	0.58	0.47
Dense	16.3	0.126	26.7	0.127	-	-	0.47

Table 2.1: Values of the fitted coefficients.

We validate the stability of the fit by excluding the top 20% of models with the lowest perplexity and finding the coefficients based on the remaining experiments. We observe that the formula remains almost unchanged in this scenario (see Table 2.6). The validation RMSE is 0.019. Results are depicted in Figure 2.6.

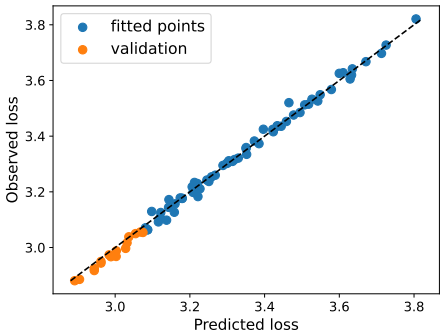


Figure 2.6: Validation of the fit.

2.5.5 MoE Scaling Properties

Comparing the part of the formula that approximates underfitting (that is, dependent on training tokens) in MoE ($30.8D^{-0.147}$) and Transformer ($26.7D^{-0.127}$), we can infer that MoE models need longer training to perform competitively but scale better after reaching that point. Nonetheless, this moment may still precede the compute-optimal for both models. On the other hand, we can see that the exponent on dense models $\alpha = -0.126$ scales better with a total number of parameters than the MoE counterpart

$\alpha = -0.115$. This should not be surprising since dense models use all parameters on each token contrary to MoE, which gains a computational advantage by activating only a subset of them. Therefore, the fair comparison of the performance has to take into account FLOPs used by each model type. In the next section, we find compute-optimal granularity for a given FLOP budget.

2.6 Optimal Allocation of Computational Budget

The goal of this section is to find optimal N, D, G for a given computational budget F. This can be done by solving the following optimization problem,

$$\begin{aligned} \underset{N,D,G}{\text{minimize}} \quad & \mathcal{L}(N, D, G) \\ \text{subject to} \quad & \text{FLOPs}(N, D, G) = F. \end{aligned}$$

2.6.1 Computational Cost of Granularity

It is important to acknowledge that increasing granularity can lead to some challenges in training the model, namely higher computational and communication costs and a larger memory footprint.

The main component responsible for higher costs is the increase in routing operations due to a larger pool of granular experts. This increase is proportional to the value of G. For standard, non-granular MoE models ($G = 1$), the routing overhead still exists, although it has been considered negligible.

Taking into account the routing operation overhead, the number of used FLOPs F is described by the following formula:

$$F = (12{d_{\text{model}}}^2 c_f + d_{\text{model}} RGc_r) \cdot D \cdot n_{\text{blocks}}, \tag{2.9}$$

given expansion rate R, granularity G, and constants that denote FLOPs per active parameter ratio, respectively, within routing (c_r) and within the rest of the network (c_f). The term $12{d_{\text{model}}}^2$ is the number of active parameters within a transformer block, while $d_{\text{model}} RGc_r$ is the number of active parameters within a routing network. We exclude embedding and unembedding from the FLOPs calculations, following Hoffmann et al. (2022).

Observe that, in contrast to scenarios where routing operations are omitted, the FLOPs calculation that incorporates routing overhead relies on both d_{model} and n_{blocks}. Consequently, an additional condition is required to determine the scaling of d_{model} and n_{blocks} in relation to an increase in N, the number of parameters. It is noted that minor variations in the depth-to-width ratio are not significant Kaplan et al. (2020). Following this analysis, we opt to adopt the assumption that $d_{\text{model}} = 64 n_{\text{blocks}}$.

The total number of parameters in the feed-forward layer, excluding the routing matrix, is $2Rd_{\text{ff}}d_{\text{model}} = 8R{d_{\text{model}}}^2$, and $4{d_{\text{model}}}^2$ in attention (key, query, value, and output projection). This results in the following formula for $N = {d_{\text{model}}}^2 \cdot (8R+4) \cdot n_{\text{blocks}}$.

2.6.2 Compute Optimal Formula

Putting all together we need to solve the following optimization problem, given F,

$$
\begin{aligned}
\underset{N,D,G}{\text{minimize}} \quad & \mathcal{L}(N, D, G) \\
\text{subject to} \quad & F = (12{d_{\text{model}}}^2 c_f + d_{\text{model}} R G c_r) \cdot D \cdot n_{\text{blocks}} \\
& N = d_{\text{model}}^2 \cdot (8R + 4) \cdot n_{\text{layers}}, \\
& d_{\text{model}} = 64 \cdot n_{\text{layers}}.
\end{aligned}
$$

All these constraints are reducible to a one-dimensional optimization problem, which is, however, hard to solve analytically. Therefore we approximate the solution using Brent's method Brent (1971). The results of this optimization for varying FLOPs budgets are plotted in Figure 2.1 while the optimal configurations of parameters for selected model sizes are presented in Table 2.2. To validate the uncertainty of these predictions, we follow Hoffmann et al. (2022) and calculate the 10th and 90th percentiles estimated via bootstrapping data (see Section 2.12 for the detailed results).

N	D	G	FLOPs	Loss
64 x 100M	4.37B	8	2.95e+18	3.133
64 x 1B	28.94B	16	1.93e+20	2.491
64 x 3B	72.90B	16	1.41e+21	2.245
64 x 7B	137.60B	32	6.46e+21	2.076
64 x 70B	941.07B	32	4.16e+23	1.694
64 x 300B	2.96T	64	5.69e+24	1.503
64 x 1T	7.94T	64	4.97e+25	1.367

Table 2.2: Compute optimal training hyper-parameters for MoE models. Optimal N and D follow approximately similar relation to these of Hoffmann et al. (2022) for active parameters around the range of $1B$ to $10B$ parameters, requiring comparably longer training for smaller models and shorter for bigger ones. Note that, this also considers optimal granularity and its FLOPs cost.

2.6.3 MoE is Always More Efficient

Contrary to the results from Clark et al. (2022), in Figure 2.1 we can see, that Mixture-of-Experts can be always considered more efficient than dense Transformers, regardless of the model size. According to our previous observations from Section 2.5.5, MoE models scale better with optimal training. However, for short training schedules, they may under-perform dense models. This means that for constant training time and increasing model size, there exists a point where both models will become very under-trained, in which scenario dense models surpass MoE. This shows why in Clark et al. (2022), where varying the number of training tokens has not been considered, MoE was predicted to be under-performing for models bigger than $1T$. However, when all training hyper-parameters N, D, G are properly selected to be compute-optimal for each model, the gap between dense and sparse models only increases as we scale.

2.7 Architecture and Training Setup

All of the models considered in this work are decoder-only Transformers trained on the C4 dataset Raffel et al. (2023). We use GPT2 tokenizer Radford et al. (2018a). Each batch consists of 0.5M tokens packed into 2048 sequences. Our optimizer is AdamW Loshchilov & Hutter (2019), with a weight decay of 0.1. In each training run, we use the maximum learning rate of 2e−4, with linear warmup for 1% steps and cosine decay to 2e−5. To improve stability, we initialize weights using the truncated normal distribution with reduced scale, as advised in Fedus et al. (2022). The models are trained using mixed precision; we always keep the attention mechanism and router in high precision. We assume the *infinite data* regime, as the number of training tokens for any of the runs in less than the number of tokens in the corpus. We follow Hoffmann et al. (2022) and perform our analysis on the smoothed training loss.

In MoE, we use the Expert Choice routing algorithm, as it guarantees a balanced expert load without tuning additional hyperparameters. To maintain compatibility with autoregressive language modeling, we apply the recipe described in Zhou et al. (2022): tokens are grouped by position across different sequences. The group size is always set to 256. We match the number of FLOPs for MoE and dense models with the same d_{model} (meaning we activate an average of $8d_{\text{model}}^2$ parameters per token in each MoE layer). In the router, softmax is performed over the expert dimension, while we choose tokens over the token dimension, as this leads to the best performance (as opposed to performing softmax over the token dimension). We put an additional layer normalization before the output of MoE layer. This gives a small improvement for standard MoE, but is crucial for the performance of models with $G > 1$.

Table 2.3 and Table 2.4 list the considered architecture and training variants for dense and MoE models, respectively.

#parameters (nonemb)	d_{model}	n_{blocks}	n_{heads}	D (in #tokens)	G
64x3M	256	4	4	16B, 33B, 66B	1, 2, 4, 8, 16
64x7M	384	4	6	16B, 33B, 66B	1, 2, 4, 8, 16
64x13M	512	4	8	16B, 33B, 66B	1, 2, 4, 8, 16
64x13M	512	4	8	130B	1, 2, 4
64x25M	512	8	8	16B, 33B,	1, 2, 4, 8, 16
64x25M	512	8	8	66B	1, 2, 4, 8
64x49M	640	10	10	16B, 33B	1, 2, 4, 8, 16
64x49M	640	10	10	66B	1, 2, 4
64x85M	768	12	12	33B	1, 2, 4

Table 2.3: Architecture and training variants (MoE models).

2.8 Discussion

Extreme Granularity. In Section 2.5, we argue that model performance improves with increasing granularity. This postulate largely aligns with the empirical findings of our study. Nonetheless, at exceedingly high granularity levels, such as $G = 64$ in models characterized by $d_{\text{model}} = 256$ and $R = 64$, there is an observable decline in performance.

#parameters (nonemb)	d_{model}	n_{blocks}	n_{heads}	D (in #tokens)
3M	256	4	4	16B, 24B, 33B, 66B
6M	256	8	4	16B, 24B, 33B, 66B
13M	512	4	8	16B, 24B, 33B, 66B
25M	512	8	8	16B, 24B, 33B, 66B
49M	640	10	10	16B, 24B, 33B, 66B
85M	768	12	12	16B, 33B

Table 2.4: Architecture and training variants (dense models).

This phenomenon is particularly evident in scenarios where the number of parameters in the routing mechanism exceeds active parameters in actual experts. Additionally, as described in Section 2.6, the utility of such high granularity is predominantly restricted to models of substantial size. In alignment with the principles outlined in Hoffmann et al. (2022), this research focuses more on findings that can be broadly applied rather than delving into the specific details of these corner-case situations. However, it is hypothesized that the efficiency of models with significantly high granularity could be potentially enhanced through careful expert initialization or modifications to the routing algorithm. These ideas are set aside to be investigated in future studies.

Varying Expansion Rate. In this study, due to computational resources constraint, we focus on $R = 64$, as recommended by Clark et al. (2022). This value of R was also used for the largest models in other works Du et al. (2022); Zhou et al. (2022) and the best-performing configuration in Fedus et al. (2022). Nonetheless, we acknowledge the importance of considering different expansion rates, as different levels of R may be chosen based on factors like the target size of the model in memory. Therefore, in Section 2.13, we present the results of the study for $R = 16$ and show that the main findings of this work are still valid in such cases.

Including R in the Formula. Another possible advancement would be to unify all of the factors N, D, G and R in one formula. While this would open the possibility of studying the relationships between coefficients in more detail, it would also be hard to practically recommend the optimal configuration in such a scenario using only FLOPs. This is because larger values of R typically lead to better performance but also incur additional memory requirements. Therefore, the choice of expansion rate may be heavily dependent on the available hardware configuration. We leave a detailed study of these factors for future work.

Modeling the Cost of Granularity. It is important to note that the exact estimation of the training cost of MoE models is dependent on the training setup, hardware, and implementation. Specifically, increasing G can lead to higher transfer costs, depending on the adopted model of distributed training. Therefore, the precise selection of hyperparameters should be made considering these factors. In this work, we model the cost of operations using FLOPs, which is common in the Scaling Laws literature Kaplan et al. (2020); Hoffmann et al. (2022); Frantar et al. (2023). Additionally, we would like to note that in our setup, we observe significant gains of granular models measured as wall-clock

time needed to achieve given perplexity (see Section 2.15 for an example).

2.9 Conclusions

This study introduces a novel hyperparameter, granularity (G), and underscores the significance of adjusting it for optimizing the efficiency of experts within MoE models. A central finding of this research is that a standard granularity of $G = 1$ is suboptimal across a broad range of FLOPs, leading to the recommendation of using higher granularity values to enhance MoE model performance and efficiency. Simultaneously, this work emphasizes the importance of varying training duration for compute-optimal settings. Consequently, both granularity and variable training length are incorporated into new scaling laws. These laws confidently demonstrate that MoE models consistently outperform dense transformers in terms of efficiency and scaling. This work not only sheds new light on the scaling laws applicable to MoE models but also provides practical guidance for improving computational efficiency in large language models. The insights are critical for the development and optimization of large-scale language models, marking a significant advancement in the field.

2.10 Comparison to Clark et al. (2022) with Effective Parameter Count Curve

One of the main conclusions of our work, that MoE is always more efficient, is contradictory to the thesis from Clark et al. (2022). Their work focuses on a fixed dataset length $D = 130B$, and they define the effective parameter count of any MoE model as the size of a dense model that achieves the same perplexity as a given MoE with a certain number of active parameters. Subsequently, they claim that as the number of parameters grows, this effective parameter count curve crosses with the active parameter curve, which means that using MoE will lead to worse performance in the future. Despite this incompatibility, we do not claim that their experiments and extrapolations are not valid. If we use our fitted scaling laws with a fixed number of training tokens D, the resulting *effective parameter count* curve (which can be properly defined only for a fixed D) would indeed cross at some parameter count in the same way as it did in Clark et al. (2022).

This crossing point of *effective parameter count* curve, that is, a number of parameters where dense Transformers surpass MoE models for a given number of training tokens D, can be calculated by determining where our scaling laws for MoE and for dense models cross, solving the following equation for N, given D: $0.47 + (\frac{2.1}{G^{0.58}} + 18.1)N^{-0.115} + 30.8D^{-0.147} = 0.47 + 16.3N^{-0.126} + 26.7D^{-0.127}$ This solves for:

D	10B	130B	1T
(N) MoE/dense Crossing Point	251B	1.9T	10T

Table 2.5: Crossing point for MoE Effective Parameter Count Curve with Dense

This crossing point will be placed further away for each increase in D.

Importantly, the lines will never cross if we train each of these models in a compute-optimal manner using the same computational budget (Figure 2.1). Our results show that regular dense models perform better than MoE only when both are severely undertrained,

as in the extrapolation from Clark et al. (2022), where a comparison is made between $1T$ parameter models trained on $130B$ steps. Moreover, in industry settings, it is common to overtrain LLMs and extremely rare to undertrain them, and our scaling laws indicate even better performance of MoE models in an overtrained regime.

2.11 Validation of the Scaling Law

In this section, we provide coefficients of the scaling law fitted with 20% of datapoints with the lowest perplexity excluded for the purpose of validation.

Model	a	α	b	β	g	γ	c
MoE	17.6	0.114	26.7	0.140	2.07	0.570	0.472

Table 2.6: Values of the fitted coefficients.

2.12 Reliability of Compute Optimal Formula

In this section, we assess the stability of our predictions presented in Section 2.6.1. Similarly to Hoffmann et al. (2022) we calculate the 10^{th} and 90^{th} percentiles estimated via bootstrapping data (80% of the data is sampled 100 times). See Table 2.7 for the details.

N	D	G
64 x 100M	(2.97B, 5.98B)	(8, 8)
64 x 1B	(21.17B, 40.73B)	(16, 16)
64 x 3B	(50.20B, 105.88B)	(16, 32)
64 x 7B	(101.06B, 205.40B)	(32, 32)
64 x 70B	(638.49B, 1.59T)	(32, 64)
64 x 300B	(1.99T, 5.62T)	(64, 64)
64 x 1T	(5.29T, 16.87T)	(64, 64)

Table 2.7: 10^{th} and 90^{th} percentiles estimated via bootstraping data.

2.13 Varying Expansion Rate

In this section, we provide results for $R = 16$. The training procedure is the same as described in 2.7. The models considered in this part are listed in Table 2.8.

We fit Eq. 2.8 using the same procedure as described in Section 2.5.4. The results are detailed in Table 2.9.

Using the coefficients and FLOPs calculation formulas, we can derive the compute optimal training parameters. The results are presented in Table 2.10.

We can observe that similarly to the case when $R = 64$, larger compute budgets imply larger optimal values of G. Note that the values for 10^{th} and 90^{th} percentiles form larger intervals in this case, as in this part we run a smaller number of experiments and

#parameters (nonemb)	d_{model}	n_{blocks}	n_{heads}	D (in #tokens)	G
64x3M	256	4	4	8B, 16B, 33B	1, 2, 4, 8, 16
64x7M	256	8	4	8B, 16B, 33B	1, 2, 4, 8, 16
64x13M	512	4	8	8B, 16B, 33B	1, 2, 4, 8, 16
64x13M	512	4	8	66B	1, 2, 4
64x25M	512	8	8	8B, 16B, 33B	1, 2, 4, 8, 16
64x49M	640	10	10	8B	1, 2, 4, 8, 16

Table 2.8: Architecture and training variants (MoE models).

Model	a	α	b	β	g	γ	c
MoE ($R = 16$)	19.64	0.124	57.07	0.169	1.18	0.986	0.472

Table 2.9: Values of the fitted coefficients.

N	D	G
16 x 100M	(10.29B, 17.73B)	(8 , 16)
16 x 1B	(53.74B, 103.54B)	(16, 32)
16 x 3B	(106.22B, 261.04B)	(16, 32)
16 x 7B	(177.65B, 511.43B)	(16, 32)
16 x 70B	(721.60B, 3.22T)	(32, 64)
16 x 300B	(1.73T, 10.69T)	(32, 64)
16 x 1T	(3.60T, 28.22T)	(32, 128)

Table 2.10: 10^{th} and 90^{th} percentiles estimated via bootstrapping data for $R = 16$.

keep shorter training durations. However, we believe that this preliminary study forms a valuable addition to the results in the main part.

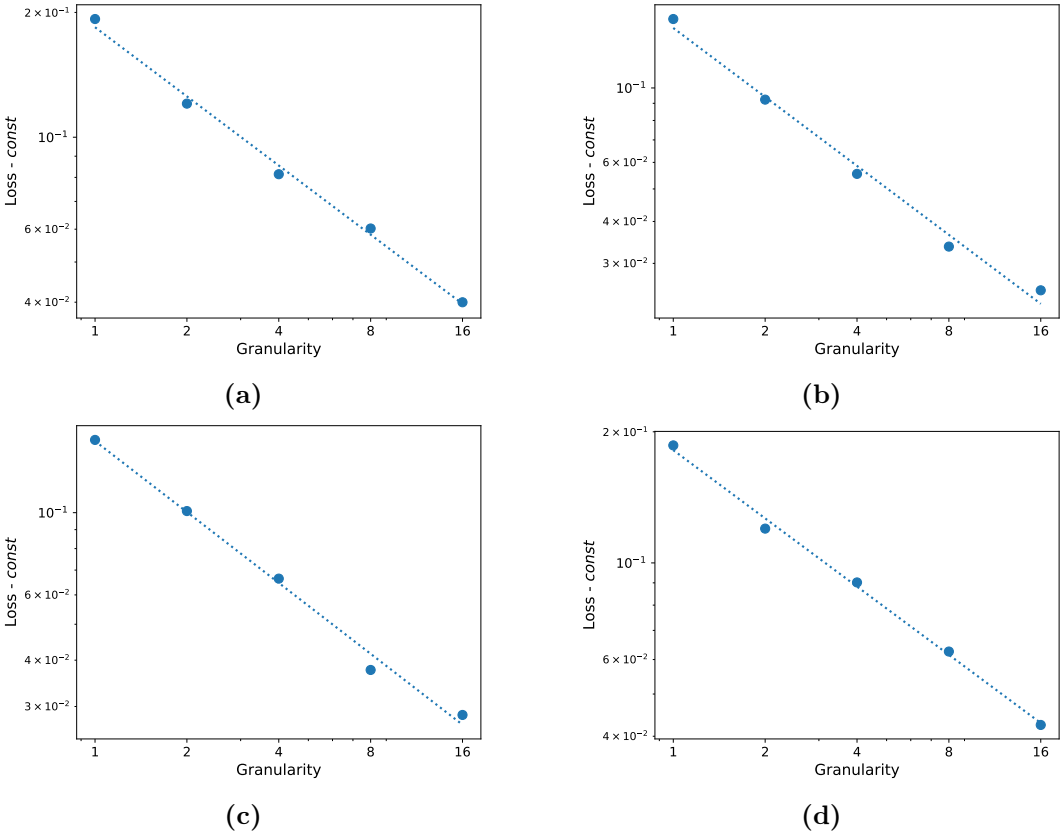


Figure 2.7: Illustration of scaling granularity when N, D are fixed for: **(a)** $N = 64 \times 25M$, $D = 16B$, $const = 3.12$ **(b)** $N = 64 \times 49M$, $D = 16B$, $const = 3.02$ **(c)** $N = 64 \times 25M$, $D = 32B$, $const = 3.03$ **(d)** $N = 64 \times 49M$, $D = 32B$, $const = 2.88$

2.14 Choosing Granularity

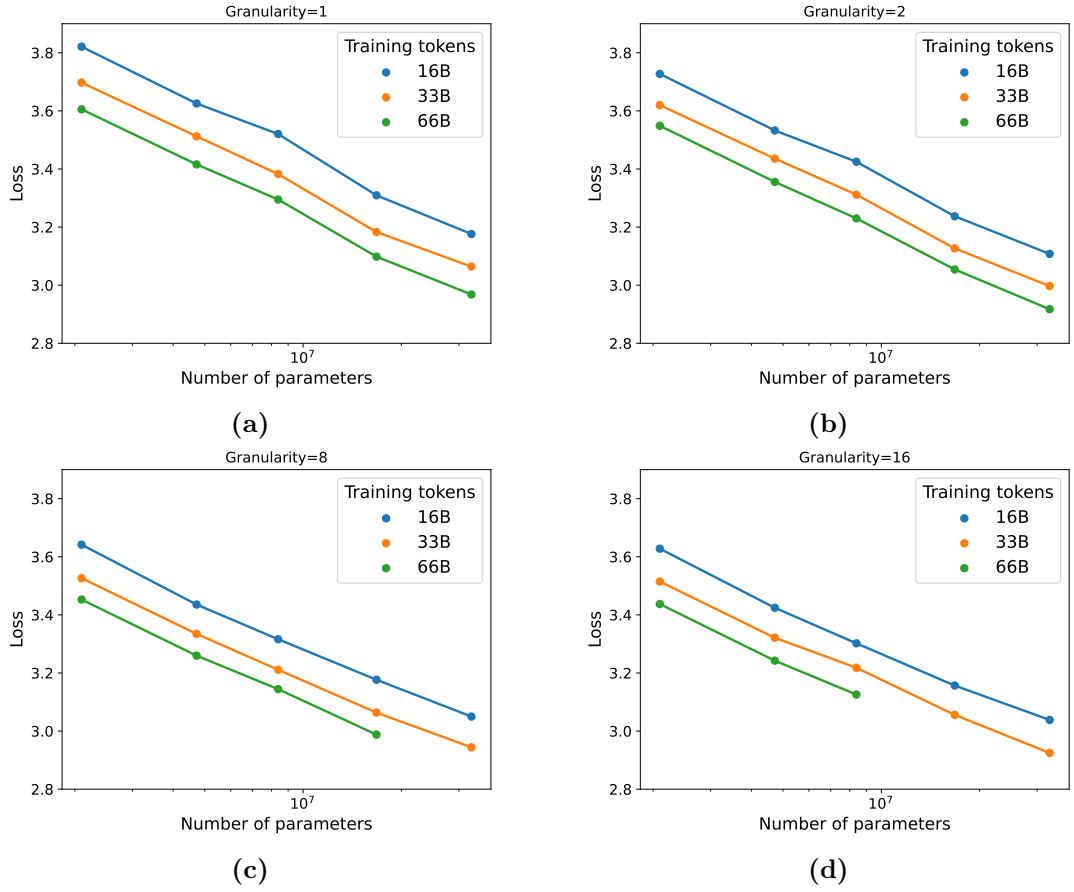


Figure 2.8: Illustration of scaling N and D for constant granularity value of: **(a)** $G = 1$ **(b)** $G = 2$ **(c)** $G = 8$ **(d)** $G = 16$.

Table 2.7 (for expansion rate 64) and Table 2.9 (for expansion rate 16) list the optimal granularity values for various compute budgets. Fig. 1 (a) presents an alternative presentation for $R = 64$. We observe that the optimal granularity values are generally similar between different expansion rates. We can generally provide the following guidelines:

- The standard value of $G = 1$ is almost never optimal.
- For a reasonable default value of G refer to Table 2.7 and Table 2.9 . For constant N or D, one can calculate optimal G and the trade-off between predicted loss and training FLOPs directly from our scaling laws using the coefficients from Table 2.6.
- The exact optimal value of granularity may differ slightly from our setup, but based on other works on scaling laws, we expect only slight differences.

2.15 Measuring Wall-clock Time

In this section, we provide an example of training curves for models with different levels of granularity, measured in terms of wall-clock training time on NVIDIA A100 GPU. We

can see that the model with $G = 8$ achieves the best performance in this case.

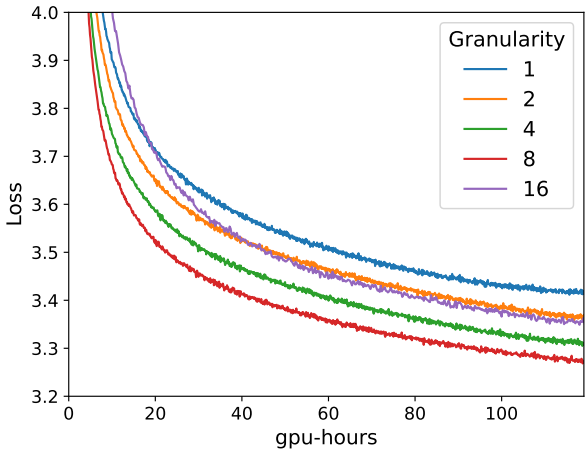


Figure 2.9: Training loss curves for model with $N = 64 \times 7M$, $D = 66B$ tokens.

Chapter 3

Joint MoE Scaling Laws: Mixture of Experts Can Be Memory Efficient

3.1 Introduction

Consider a motivating question: Is an MoE model the optimal choice when constrained by a fixed memory budget, such as a single H100 node? While computational efficiency is important, it does not directly determine the optimal number of experts. Increasing the number of experts has minimal impact on computation but can drastically raise memory requirements, often to a prohibitive level.

To address this question, we derive a *joint* scaling law for both dense and MoE models, accounting for key factors such as the number of active parameters, dataset size, and number of experts. This framework provides a rigorous analysis of model performance under strict memory constraints. Our findings reveal that, contrary to common assumptions, MoE models can be more memory-efficient than dense models.

Our work is the first to provide detailed guidance on selecting the optimal number of experts for MoE models, balancing both computational and memory constraints. Our conclusions are based on extensive, large-scale experiments comprising 270 models, scaled up to 5B parameters.[2]

In summary, the key contributions of this work are:

- We derive a joint scaling law for Mixture of Experts and dense models,

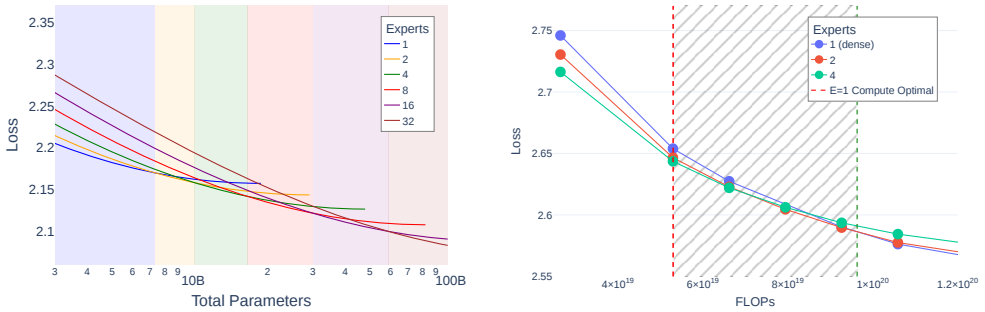
$$\begin{aligned}\mathcal{L}(N_{\text{act}}, D, \hat{E}) = \quad & a\hat{E}^{\delta} N_{\text{act}}^{\alpha+\gamma\ln(\hat{E})} \\ & + b\hat{E}^{\omega} D^{\beta+\zeta\ln(\hat{E})} + c, \end{aligned} \tag{3.1}$$

  where $\mathcal{L}$ is the final training loss, N_{act} is the number of active parameters, D is the dataset size, $\hat{E}$ is the monotonic transformation of the number of experts, and c is the irreducible entropy of the dataset.

- Based on the proposed scaling law, we show that the choice of the optimal number of experts (including dense models with $E = 1$) depends on specific computational and memory constraints, see Figure 3.1. Furthermore, we demonstrate how the optimal token-to-parameter ratio depends on E.

- We show that MoE can often be the preferred alternative to dense models, even if GPU memory is the constraining factor. We validate our theoretical findings by

[2] Checkpoints and inference code are available on Hugging Face.



(a) (b)

Figure 3.1: **(a)** The loss of memory-constrained models predicted using our scaling law under a fixed training budget of 10^{22} FLOPs. Each curve represents a different number of experts. The lines are truncated at compute-optimal points, as undertrained models are both bigger and worse in terms of loss, thus pointless in a memory-constrained scenario. Shaded areas present memory optimal number of experts for the corresponding parameter budgets. **(b)** Experimental validation of the thesis that MoE can be memory optimal. The marked area shows an interval in which a training compute-matched MoE achieves better loss than an overtrained dense model with the same number of total parameters (1.1B). The resulting MoE was trained for longer and had less active parameters, making it more practical.

training a set of 1.1B-parameter models under identical compute and total memory budgets. The MoE models achieve a lower final loss, confirming their superior efficiency in practice. Furthermore, we observe that MoE models only have lower loss, but deliver higher performance during inference.

3.2 Joint MoE Scaling Laws

We now derive the functional form of our joint scaling laws for both dense Transformers and MoE, relating the number of active model parameters N_{act}, training tokens D, and MoE experts E.

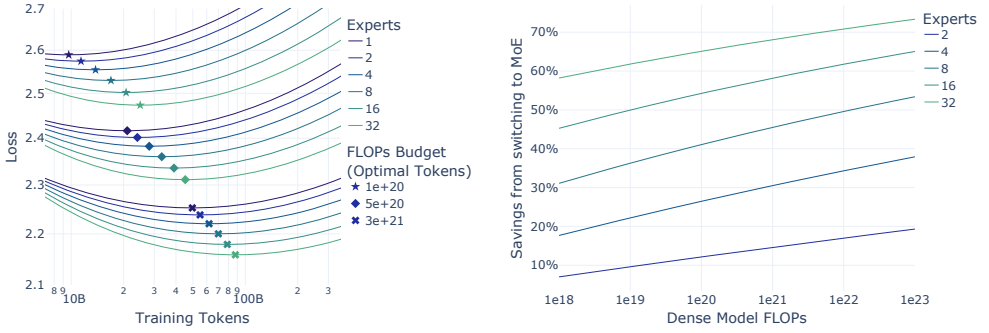
Fixed Number of Experts. Following Hoffmann et al. (2022) and established practice in the literature (Frantar et al., 2023; Kumar et al., 2024a), we postulate the following form of the equation:

$$\mathcal{L}(N_{\text{act}}, D, E) = m(E)N_{\text{act}}^{\mu(E)} + n(E)D^{\nu(E)} + c(E), \tag{3.2}$$

assuming that if we fix the number of experts the model performance can be described using Equation 1.1. In the subsequent part, we will postulate how m, μ, n, ν, c depend on E, deriving the joint equation.

Constant Factor. $c(E)$ represents irreducible loss caused by the inherent entropy of the dataset. Thus, it does not depend on the architecture (E in our case):

$$c(E) := c.$$



(a) (b)

Figure 3.2: **(a)** IsoFLOP profiles for selected training budgets. Compute optimal points are marked for each curve. **(b)** Savings from switching from a compute-optimal dense model to MoE. The advantage of using MoE increases with larger models and expert counts.

Interaction of E with Model and Dataset Size. To quantify the interaction between the number of experts and other training parameters, we gather observations from related work:

1. Scaling in E can be described as a power law (Clark et al., 2022).
2. For a fixed dataset size, as model size increases, the benefit of using an MoE diminishes (Clark et al., 2022).
3. For a fixed model size, as the number of training tokens increases, the benefit of an MoE grows (Chapter 2).

Motivated by Observation 1, we set

$$m(E) = aE^{\delta}, \quad n(E) = bE^{\omega},$$

reflecting the power-law relation between E and the loss.

Additionally, to ensure flexibility in modeling Observations 2 and 3, we introduce an interaction with the exponents over N_{act} and D:

$$\mu(E) = \alpha + \gamma \ln(E),$$
$$\nu(E) = \beta + \zeta \ln(E).$$

Note that if we ignore the second and third terms in Equation 3.2, this yields a functional form identical to Equation 2.1.

Empirically, we observe a good fit for our formula, as described in Section 3.4. This shows that our proposed interactions between E, N_{act}, and D can accurately model the performance of MoE models.

Modeling of E. When the number of experts is small, a certain overhead, caused, for example, by interference from auxiliary losses, can overshadow the benefits of conditional computation. Additionally, using very large numbers of experts brings diminishing returns. To account for these phenomena, we follow Clark et al. (2022) and their transformation of the number of experts $\hat{E}$.

Joint MoE Scaling Law. By combining these observations, we establish the final form of our scaling law:

$$\mathcal{L}(N_{\text{act}}, D, \hat{E}) = a\hat{E}^{\delta} N_{\text{act}}^{\alpha+\gamma\ln(\hat{E})} + b\hat{E}^{\omega} D^{\beta+\zeta\ln(\hat{E})} + c. \tag{3.3}$$

We fit the coefficients in Equation 3.3 based on the results of our experiments; see Table 3.3. In Section 3.3, we present the outcomes and findings derived from the scaling laws. The details of the training runs, as well as the fitting procedure, are described in Section 3.4.

3.3 Compute and Memory Optimality

In this section, we employ our scaling laws to derive recommendations on optimal settings in various training and inference scenarios. See Section 3.7 for details on counting FLOPs, the relations between active and total parameters, and other technical details.

3.3.1 Compute Optimality

A model is considered compute-optimal if, among models trained with the same compute budget F, it achieves the lowest loss. To find such an optimal configuration, we optimize the following:

$$\arg\min_{N_{\text{act}},D,E} \mathcal{L}(N_{\text{act}}, D, E) \tag{3.4}$$

$$\text{s.t. } 6N_{\text{act}}D = F \tag{3.5}$$

Optimal N and D Depend on the Number of Experts. Assuming a given number of experts E, the compute-optimal training configuration can be achieved by selecting the appropriate trade-off between training tokens and model size. IsoFLOP slices comparing the predicted loss with dataset size for selected compute budgets are plotted in Figure 3.2 (a).

For any fixed E our scaling law has the Chinchilla functional form of Equation 1.1. Thus, from Hoffmann et al. (2022), the compute-optimal number of tokens and active parameters for the budget F and the number of experts E are given by

$$N_{\text{act}}^{\text{opt}}(F) = G\left(\frac{F}{6}\right)^{a}, \quad D^{\text{opt}}(F) = G^{-1}\left(\frac{F}{6}\right)^{b}, \tag{3.6}$$

where

$$G = \left(\frac{\mu(E)m(E)}{\nu(E)n(E)}\right)^{\frac{1}{\mu(E)+\nu(E)}},$$

and

$$a = \frac{\nu(E)}{\mu(E)+\nu(E)}, \quad b = \frac{\mu(E)}{\mu(E)+\nu(E)}.$$

The full derivation of $N_{\text{act}}^{\text{opt}}$, D^{opt} can be found in App.3.9. We compare the optimal configurations for several compute budgets in Table 3.1.

Both from comparing the IsoFLOP slices (Figure 3.2) and the values listed in the table, we can see that the compute-optimal configuration for a given compute budget clearly depends on E, with MoE models requiring comparatively larger datasets and correspondingly smaller numbers of active parameters.

Training Budget	Experts	$N_{\text{act}}^{\text{opt}}$	D^{opt}
1×10^{20}	1	1.7B	9.7B
	2	1.5B	11.4B
	4	1.2B	13.9B
	8	990M	17B
	16	810M	20.7B
5×10^{20}	1	4B	21B
	2	3.5B	24B
	4	3B	28B
	8	2.5B	33.2B
	16	2.1B	39B
1×10^{21}	1	5.7B	29.3B
	2	5B	33B
	4	4.4B	38B
	8	3.8B	44.3B
	16	3.3B	51.2B

Table 3.1: Example compute-optimal training configurations for MoE models. For every training budget as the number of experts increases, the optimal D^{opt} also goes up while $N_{\text{act}}^{\text{opt}}$ decreases.

Finding 1.
More experts → higher tokens-to-param ratio.
Assume a fixed compute budget. In this scenario, when increasing the number of experts, it is optimal to decrease the number of active parameters and increase the number of training tokens accordingly (Table 3.1).

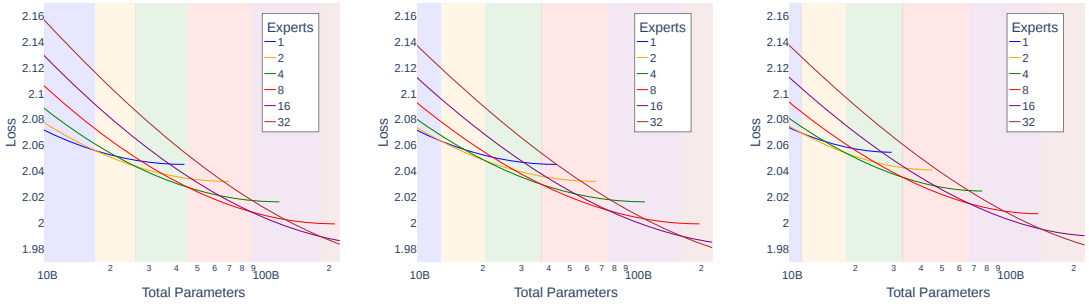
Mixture of Experts is Compute Optimal. Now, we compare the performance across various numbers of experts, with respective values of tokens and active parameters optimized. As illustrated in Figure 3.2, we observe significant compute savings for MoE models compared to dense models, with a larger number of experts providing more pronounced benefits.

Finding 2. More experts → better performance. For a given compute budget, increasing the number of experts always improves performance, provided the size of the model and the number of training tokens are adjusted (Figure 3.2 (a)).

The higher efficiency of MoE in terms of training compute comes at a price of increased memory requirements. However, somewhat surprisingly, we find that MoE models can outperform dense models *of the same size* trained with the same amount of training compute—a result we describe in more detail in the next subsection.

3.3.2 Model Memory Optimality

Often, it is insufficient to consider models solely from the perspective of compute optimality, as a compute-optimal model can be impractically large, preventing its deployment



(a) (b) (c)

Figure 3.3: Loss predicted for various expansion rates at a FLOPs budget $F = 5e22$. The x-axis denotes the size of the corresponding dense model, possibly with KV cache. (a) The model size is simply the number of parameters. (b) The model size includes the KV cache (assuming 8192 tokens). (c) Additionally to KV cache, the training budget is reduced by the inference cost on 100B tokens.

on available hardware. Additionally, a model can be inefficient when run on a GPU with a small batch size (He, 2022). It is thus natural to consider a straightforward extension to the notion of compute optimality, specifically model memory optimality. We say a model is memory optimal if, among models trained with the same compute budget F and having at most M parameters, it achieves the lowest loss:

$$\begin{aligned} &\arg\min_{N_{\text{act}},D,E} \mathcal{L}(N_{\text{act}}, D, E) \\ &\text{s.t. } 6N_{\text{act}}D = F, \quad N_{\text{total}} \leq M \end{aligned}$$

Note that model memory-matched dense and MoE models differ in the number of active parameters—MoE uses just a fraction of them. Intuitively, it should thus have worse performance. At the same time, given some budget, it can be trained on more tokens, lowering the loss. Our scaling laws suggest that MoE models can be model memory efficient. We validate this claim by training a 1.1B dense model and a model size and FLOP matched $E = \{2, 4\}$ counterparts (Figure 3.1). Significantly, the MoE models attains lower loss even if the dense model is overtrained (i.e., after passing its compute-optimal token count).

Finding 3. MoE can also be *memory*-efficient.
A total-parameter-matched MoE model can outperform a dense model trained with the same compute budget (Figure 3.1). Moreover, such an MoE model is more compute- *and* memory-efficient at inference.

3.3.3 Total Memory Optimality

During autoregressive generation, a decoder-only model processes a single token while storing activations (keys and values) for previous tokens in the KV cache. In the case of multi-head attention, its size equals $2T \times N_{\text{blocks}} \times d_{\text{model}}$, where T is the number of tokens in the cache (possibly within multiple sequences in the batch). Including the cache

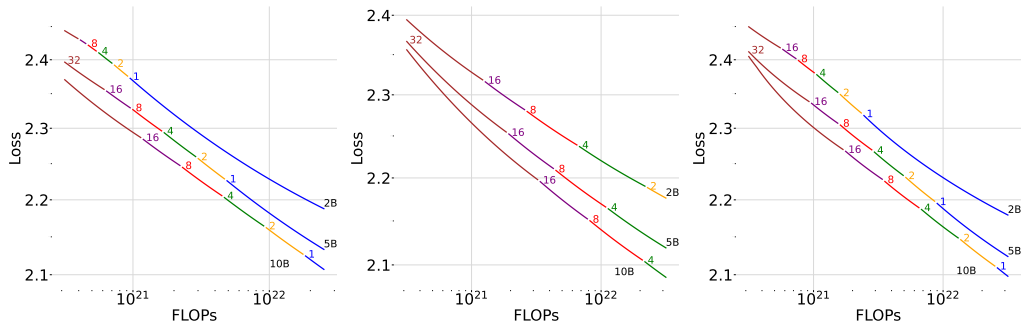


Figure 3.4: Investigation of the optimal number of experts for three different model sizes, 2B, 5B, and 10B; and in three different scenarios, from left to right: simply measuring the size of the model, including the size of a KV-cache with 32k tokens, and including the inference cost of processing 100B tokens.

size yields the optimization criterion:

$$\begin{aligned} &\arg\min_{N_{\text{act}},D,E} \mathcal{L}(N_{\text{act}}, D, E) \\ &\text{s.t. } 6N_{\text{act}}D = F, \quad N_{\text{total}} + 2TN_{\text{blocks}}d_{\text{model}} \leq M \end{aligned}$$

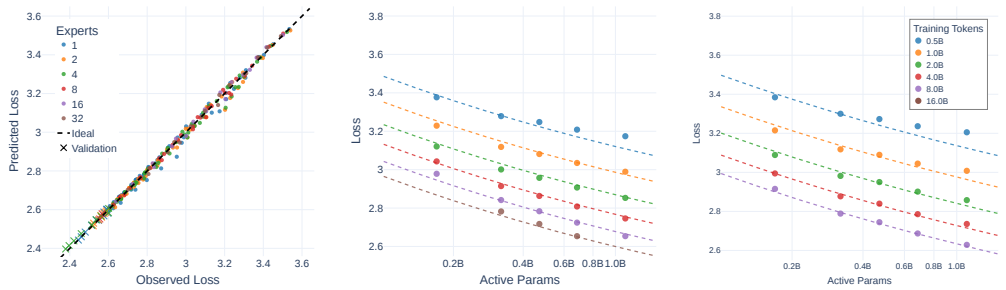
For practical values of T, a fair comparison of memory requirements should include the size of KV cache in addition to the model size. Figure 3.3 (b) presents the optimal models for a given compute and varying memory constraints when the size of the KV cache is included. Importantly, MoE models compare more favorably to dense models in this graph, and as T increases, they outperform dense models at ever smaller model sizes. In Figure 3.1 (b), the $E = \{2, 4\}$ models employ a smaller KV cache. It means that if memory is constrained, the MoE model can store longer contexts or work with a larger batch size than the dense model.

3.3.4 Inference Optimality

Large models, while capable, might also be too costly to run due to their high computational demand. To account for this drawback, we can further assume that a model will process some number of tokens, D_{inf}, throughout its lifetime and find the best model whose demands do not exceed some predefined joint training and inference budget:

$$\begin{aligned} &\arg\min_{N_{\text{act}},D,E} \mathcal{L}(N_{\text{act}}, D, E) \\ &\text{s.t. } 6N_{\text{act}}D + 2N_{\text{act}}D_{\text{inf}} = F. \end{aligned}$$

Figure 3.3 (c) presents the optimal models for a given compute and varying memory constraints if a joint budget needs to accommodate both training and inference demands. We find that in this scenario, MoE models outperform dense at smaller scales than in simple compute-optimality due to decreased inference FLOPs. The $E = 2$ and $E = 4$ models shown in the Figure 3.1 use respectively 36% and 61% less FLOPs per token than their dense counterpart.



(a) **(b)** **(c)**

Figure 3.5: **(a)** Quality of the fit. The maximum absolute error on the held-out extrapolation is 0.018. **(b)** Predicted loss compared with an observed loss for $E = 1$. **(c)** Predicted loss (dashed line) compared with an observed loss for $E = 4$. We can see that on the training dataset, the error increases in an undertrained setting ($D/N < 1$ — more parameters than tokens). However, this scenario is never practical from our perspective.

3.3.5 Summary

The notions of inference optimality and total memory optimality can naturally be combined. Figure 3.3 (c) presents a comparison between different numbers of experts, where the KV cache is included in the model's memory requirements and the compute budget is shared between training and inference. Finally, Figure 3.4 investigates the optimal E for a sample of model sizes while including the KV cache and considering the inference cost.

For practitioners, as a simplification of our analysis, we propose a general rule of thumb:

Rule of Thumb. An MoE model with $E \leq 8$ experts, trained on E-times more tokens than a compute-optimal dense model, outperforms it while maintaining the same total parameter count.

Note that, in this scenario FLOPs matched MoE will generally have less than E-times larger dataset, but we wanted to keep this rule simple and conservative. Detailed comparisons and differences between memory and FLOPs matched models can be found on Figures 3.1 & 3.4.

It is important to remember that such scaling might not always be possible in practice due to limited dataset sizes. This points towards a possible drawback while using MoE and underscores the need for growth in dataset sizes.

3.4 Fitting the Scaling Law

In this section, we present details of experiments and the procedure of fitting the scaling law parameters, see Table 3.3. Those results are based on extensive, large-scale empirical evidence, including 270 models with up to 5B parameters, trained on a variety of compute budgets. For a full list of experiments, see Section 3.14.

	Memory Constraint		
Training Compute	24GB	80GB	640GB
1×10^{21}	16	≥ 32	≥ 32
1×10^{22}	4	16	≥ 32
1×10^{23}	1	8	≥ 32
1×10^{24}	1	1	16

Table 3.2: Optimal E for different training budgets and three typical memory constraints, corresponding to an RTX4090 GPU, an H100 GPU, and an 8xH100 GPU node. We assume 16k tokens in the KV cache and bfloat16 for storing model weights and activations.

3.4.1 Model Hyperparameters

The selection of hyperparameters and training details is crucial for ensuring the robustness of scaling laws (Porian et al., 2024; Pearce & Song, 2024b). In our work, we employ a set of best practices and modern design choices, aiming to provide accurate predictions applicable to real-life practice.

All models used in this study are decoder-only Transformers trained on the highly filtered FineWeb-Edu (Penedo et al., 2024a). It is a subset of FineWeb, whose curation process was guided using popular benchmarks. FineWeb-Edu is selected using a filter for highly educational content. We use a Transformer model with Switch (Fedus et al., 2022) layers, using standard values of router z-loss 0.001 and load balancing loss 0.01. The GPT-2 tokenizer (Radford et al., 2018a) is employed. For better stability, weight initialization follows a truncated normal distribution with a reduced scale of 0.1, as suggested by Fedus et al. (2022). Mixed precision training is used, with the attention mechanism, position embeddings RoPE Su et al. (2024) and router always maintained at high precision. The models use the SwiGLU activation (Shazeer, 2020b) with hidden size equal to $3d_{\text{model}}$ and activate one expert per token (unless the token is dropped due to limited capacity). For evaluation, we increase the capacity factor to ensure dropless processing of the tokens.

Batch Size Ramp-up

Performance of a deep learning optimization procedure can suffer as a result of using an exceedingly large batch size (McCandlish et al., 2018). To mitigate this potential issue, especially early in the training, we employ batch-size ramp-up. Similar strategies are used in contemporary LLM training runs (Rae et al., 2022; Dubey et al., 2024b). We increase the batch size from 64K to 128K after 0.5B training tokens and further to 256K after 1B training tokens. Instead of using noise scale as a critical batch size predictor (McCandlish et al., 2018) we opted for a straightforward grid to directly predict a transition point after which increased batch size does not impair performance.

Learning Rate Scaling

Kaplan et al. (2020) have shown that scaling laws for hyperparameters can be used to adjust them according to the size of the model in the case of dense Transformers. For MoE models, we find the literature inconclusive–while some (Dai et al., 2024) pretrain MoEs with lower LR than corresponding dense models, others (Zoph et al., 2022a) report

better performance when finetuning MoEs with higher learning rates. To fill this gap, we derive a scaling law for the peak learning rate for MoE based on the number of active non-embedding parameters $N_{act\setminus e}$ and the number of experts E:

$$LR(N_{act\setminus e}, E) = \exp(8.39 - 0.81 \ln(N_{act\setminus e}) - 0.25 \ln(E)), \tag{3.7}$$

and use this equation to set the learning rate in our main scaling laws experiments. We fit the coefficients of this equation using the least squares method, minimizing the error between the prediction and the optimal learning rate from the experiment grid. Contrary to Kaplan et al. (2020), we use a linear transformation of the parameter count to predict the logarithm of the learning rate, instead of directly predicting the learning rate. This approach allows us to avoid the breakdown of the formula above 10^{10} parameters mentioned in their work, where the predicted learning rate becomes negative. This phenomenon is independent of the actual fit and is simply a property of the formula used. Besides being well-defined in the extrapolation, we argue that optimal learning rates visibly follow this logarithmic trend, as seen in Figure 3.8.

Finding 4. More experts → lower learning rate.
Increasing the number of experts in MoE model should be accompanied by lowering the learning rate accordingly (Figure 3.8).

The second difference between our formula and the one by Kaplan et al. (2020) is incorporating the number of experts, allowing us to model the optimal behavior of this hyperparameter across dense models and different MoEs. This is an important detail that allows unbiased comparison among different models, ensuring that each one is optimally tuned. Furthermore, it allows us to answer the question of whether MoE should be trained with a lower or higher LR. While our formula accommodates both scenarios, we can clearly see in Figure 3.8 that increasing E requires lower learning rates, resulting in a negative value for the coefficient. Moreover, we verify this thesis by tuning the fit on $E = 1$ and $E = 8$, and validating it on interpolation $E = 4$ and extrapolation $E = 32$. In both cases, the validation predicts the optimal learning rate for the model configuration or a value with practically the same performance.

In Figure 3.9, we perform an ablation of this additional power law on E by repeating our entire fitting procedure without the E component. This shows, especially with the extrapolation on $E = 32$, that dependence on E is crucial, and its omission can impair the performance of MoEs.

Further details about our scaling rule for learning rates can be found in the plots in Section 3.13.

Learning Rate Schedule

Hägele et al. (2024) suggest that a constant learning rate schedule can yield similar performance to other established methods, such as the cosine schedule. At the same time, it offers a valuable advantage when varying training duration, as intermediate checkpoints can be reused when training models for a longer time. With a cosine schedule, intermediate checkpoints can introduce bias into the fit, according to the analysis of Kaplan et al. (2020) by Hoffmann et al. (2022). We employ a constant learning rate schedule with a linear warmup over the initial 130M tokens and with a linear decay from the peak learning rate to 0 over the final 20% of tokens. For each model size, longer runs reuse intermediate checkpoints from the shorter ones.

3.4.2 Optimization of Formula Coefficients

Following Hoffmann et al. (2022), we use the LBFGS algorithm to optimize the coefficients of formula 3.3. See Section 3.8 for details. We observe a good fit with $\texttt{RMSE}_v = 0.0039$ on a held-out set of our 30 runs with the lowest loss, and $\texttt{RMSE}_t = 0.0062$ on the training dataset. To further verify the validity of our formula, we train separate Chinchilla scaling laws 1.1 for different E using the same hyperparameters and the corresponding subset of the initializations grid. This approach serves as a lower bound for loss of our joint formula on the training dataset, as it can emulate its coefficients; however, it is more prone to overfitting because effectively more parameters are utilized. Using this approach, we obtain lower error on the training dataset of $\texttt{RMSE}_t^{\text{sep}} = 0.0059$ and marginally higher on the validation $\texttt{RMSE}_v^{\text{sep}} = 0.0041$. We believe this is strong confirmation that our joint formula is actually describing how variable E influences training.

In Figure 3.5, we visually verify the extrapolation of the joint fit. Prediction errors are categorized by different numbers of experts, highlighting that our joint formula is not biased for any specific E.

3.5 Limitations and Future Work

In our work, we focus on the standard MoE variant, where the size of the expert is the same as the size of the feed-forward layer of a corresponding dense model. Some recent findings (chapter 2 and Dai et al. (2024); Muennighoff et al. (2024); Team (2024b)) indicate that fine-grained MoE models are more efficient and, most probably, would enhance our reported benefits of using MoE. Similarly, adopting a dropless MoE (Gale et al., 2023) approach instead of relying on a capacity factor could lead to further improvements. We leave the integration of those MoE improvements for future work.

Moreover, our Chinchilla-based optimality analysis uses FLOPs, that may not reflect wall-clock training time of models with different architectures. While analyzing total parameter, instead of active parameter matched models partly alleviates this issue because of the same memory-bottleneck, various implementations and distributed training algorithms are not considered in this work.

We assumed, the Chinchilla scaling law (1.1) as the basis of our formulas. While this is well-grounded in literature, this formula is known to have limitations, especially for a wide range of token-to-parameter ratios. We observed this also in some of our experiments, as outliers often are highly under or over-trained.

3.6 Conclusions

In this work, we derived the joint scaling laws for Mixture of Experts, relating the loss of the model to the number of parameters, the number of training tokens, and the number of experts. By considering both compute and memory constraints, as well as the expected inference workload, we demonstrated that MoE models can outperform dense models even when constrained by memory usage or total parameters, contrary to common assumptions and intuitions that MoE models are more memory-intensive than dense models.

Our analysis reveals how the optimal training strategies shift as the number of experts varies. This provides a principled framework for selecting MoE hyperparameters under given constraints, highlighting the trade-offs between memory and compute performance.

a	α	δ	γ	b	β	ω	ζ	E_{start}	E_{max}	c
35.91	-0.1889	-0.2285	0.0098	35.98	-0.1775	0.5529	-0.0259	2.0732	290.4521	1.3637

Table 3.3: Fitted coefficients of our joined formula.

3.7 Technical Details

3.7.1 Counting Parameters

There are many ways the size of a model can be measured. The two most important distinctions are whether total or active parameters are counted and whether the parameters in the embedding and unembedding layers are counted. Various papers assume different notations, notably Kaplan et al. (2020) use nonembedding parameters while Hoffmann et al. (2022) opt for the parameter count including embedding and unembedding. Throughout our work, we try to make it clear which way of counting we are using in each particular instance. When no additional information is given, N_{act} and N_{total} denote respectively active and total parameters, including the embedding and unembedding.

If we let d_{model} be the hidden dimension of a model, and d_{vocab} be the vocabulary size (50,257 in our case), then the following relations hold:

$$N_{\text{total}} = 2d_{\text{model}}d_{\text{vocab}} + (4 + 9E)N_{\text{blocks}}d_{\text{model}}^2 \tag{3.8}$$

$$N_{\text{act}} = 2d_{\text{model}}d_{\text{vocab}} + 13N_{\text{blocks}}d_{\text{model}}^2 \tag{3.9}$$

3.7.2 Counting FLOPs

Basing on Sardana et al. (2024b), we assume the cost of training to be $F_{\text{training}} = 6N_{\text{act}}D_{\text{training}}$, and the cost of inference to be $F_{\text{inference}} = 2N_{\text{act}}D_{\text{inference}}$. Due to the relatively small number (≤ 32) of experts used with implicit expert granularity of 1, we can consider the memory and FLOPs cost of routing to be negligible, following Clark et al. (2022).

3.7.3 Model Configs

The vast majority of our experiments use a simple rule for scaling the config, i.e. $N_{\text{blocks}} = N_{\text{heads}} = \text{d}_{\text{model}}/64$ and assume these relations hold in all calculations. We base this rule on findings by Kaplan et al. (2020).

3.8 Fit Details

Following Hoffmann et al. (2022), we use the LBFGS algorithm with a learning rate of $1e{-}4$ and weight decay of $1e{-}5$ to fit the coefficients of Equation 3.3, optimizing the Huber loss with $\delta = 0.01$ over the set of our training runs described in table in Section 3.14. Instead of removing outliers and underperforming models from the training set, we underweight them proportionally to the loss. Optimization hyperparameters were manually tuned to minimize error over the training dataset. The final fitted coefficients of Equation 3.3 are within the boundaries of the grid of initializations given by: $\alpha \in \{0.05, 0.25, 0.5\}$, $\beta \in \{0.05, 0.25, 0.5\}$, $A \in \{30, 100, 300\}$, $B \in \{30, 100, 300\}$, $C \in \{0.5, 1, 2\}$, $\delta \in \{-0.5, 0, 0.5\}$, $\gamma \in \{-0.5, 0, 0.5\}$, $\omega \in \{-0.5, 0, 0.5\}$, $\zeta \in \{-0.5, 0, 0.5\}$. The selected coefficients were those with the lowest score, defined as the sum of RMSE on

E	m	μ	n	ν	c
1	30.3640	−0.1817	53.9838	−0.1965	1.3637
2	27.7982	−0.1780	66.8401	−0.2065	1.3637
4	24.8462	−0.1731	87.7022	−0.2192	1.3637
8	21.8330	−0.1676	119.9126	−0.2338	1.3637
16	19.0159	−0.1617	167.5073	−0.2494	1.3637
32	16.5424	−0.1557	234.6726	−0.2652	1.3637

Table 3.4: The fitted coefficients of our joint formula, Equation (3.3), reduced to the Chinchilla scaling law, Equation (1.1), for a given number of experts, E. We observe that the dataset exponent, ν, increases significantly. This is one of the reasons why compute-optimal parameter-to-token ratios change with E.

the training and a held-out extrapolation validation set. The formula in Equation 3.3 was calculated in logarithm, without any exponentials, using only linear transformations and the logsumexp operation. It was optimized to predict the logarithm of L, and parameters a, b, and c were optimized in logarithm. All these steps were taken to increase numerical stability and were essential for proper convergence.

3.9 Derivation of $N_{\mathbf{act}}^{\mathbf{opt}}$ and $D^{\mathbf{opt}}$

To derive the optimal N_{act}, D given some compute budget F and E, one needs to solve:

$$\arg\min_{N_{\text{act}},D} L_{\hat{E}}(N_{\text{act}}, D) \quad \text{s.t.} \quad F = 6\, N_{\text{act}}\, D.$$

To solve for N_{act}, substitute:

$$D = \frac{F}{6\, N_{\text{act}}},$$

and set the derivative to 0:

$$\frac{d}{dN_{\text{act}}} L_{\hat{E}}(N_{\text{act}}, D) = \frac{d}{dN_{\text{act}}} \Big[m(\hat{E})\, N_{\text{act}}^{\mu(\hat{E})} \;+\; n(\hat{E}) \Big(\frac{F}{6}\Big)^{\nu(\hat{E})} N_{\text{act}}^{-\nu(\hat{E})} \Big] = 0.$$

After rearranging:

$$N_{\text{act}}^{\text{opt}} = \Big(\frac{m(\hat{E})\,\mu(\hat{E})}{n(\hat{E})\,\nu(\hat{E})}\Big)^{-\frac{1}{\mu(\hat{E})+\nu(\hat{E})}} \Big(\frac{F}{6}\Big)^{\frac{\nu(\hat{E})}{\mu(\hat{E})+\nu(\hat{E})}}.$$

The derivation of D^{opt} is analogous.

3.10 Token Dropping Analysis

In general, we observe that the load balancing loss quickly induces balance between experts. Overall, the percentage of dropped tokens is low and doesn't exceed 10%, therefore doesn't significantly affect the training efficiency. Below we present a plot of per-layer average amount of dropped tokens (excluding the first 10% of training), for 2 selected active parameter counts and number of experts varying from 2 to 32. We observe relatively the largest ratios of dropped tokens in the initial and last layers.

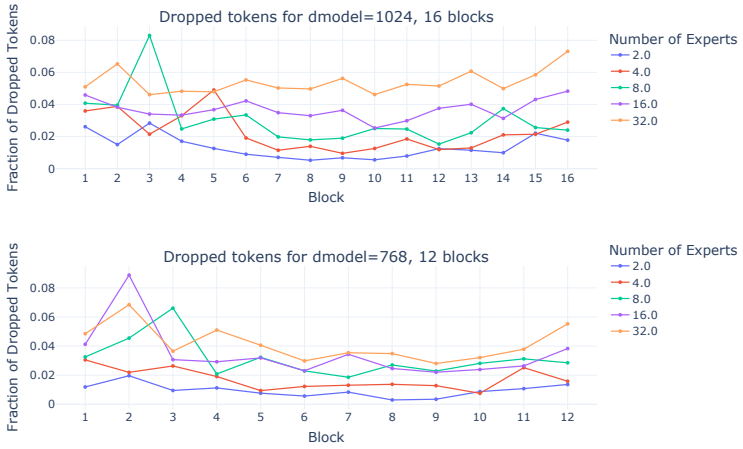


Figure 3.6: Dropped tokens for selected models.

3.11 Downstream Performance

In addition to measuring pretraining loss, we evaluate downstream performance using LM Evaluation Harness Gao et al. (2023). Unlike in training, we employ dropless MoE. We observe a strong correlation between the pretraining perplexity and downstream performance across all E's. The results can be seen in Figure 3.7. On some tasks (HellaSwag, Winogrande, SciQ), dense models seem to outperform MoE models given the same pretraining perplexity. They also seem to be more robust to domain shift in language modeling, as exemplified by the results on the LAMBADA benchmark. On some other tasks (e.g. OpenBookQA), MoE models seem to fare similarly or slightly better than dense models if they have been trained to the same pretraining perplexity.

3.12 Bootstrap Results

To quantify the uncertainty of our derived results, we calculate bootstrapped results for the optimal number of active parameters and training tokens, following (Hoffmann et al., 2022). The results are shown in Table 3.5. We sample 80% of data 100 times and report the 10th and 90th percentiles.

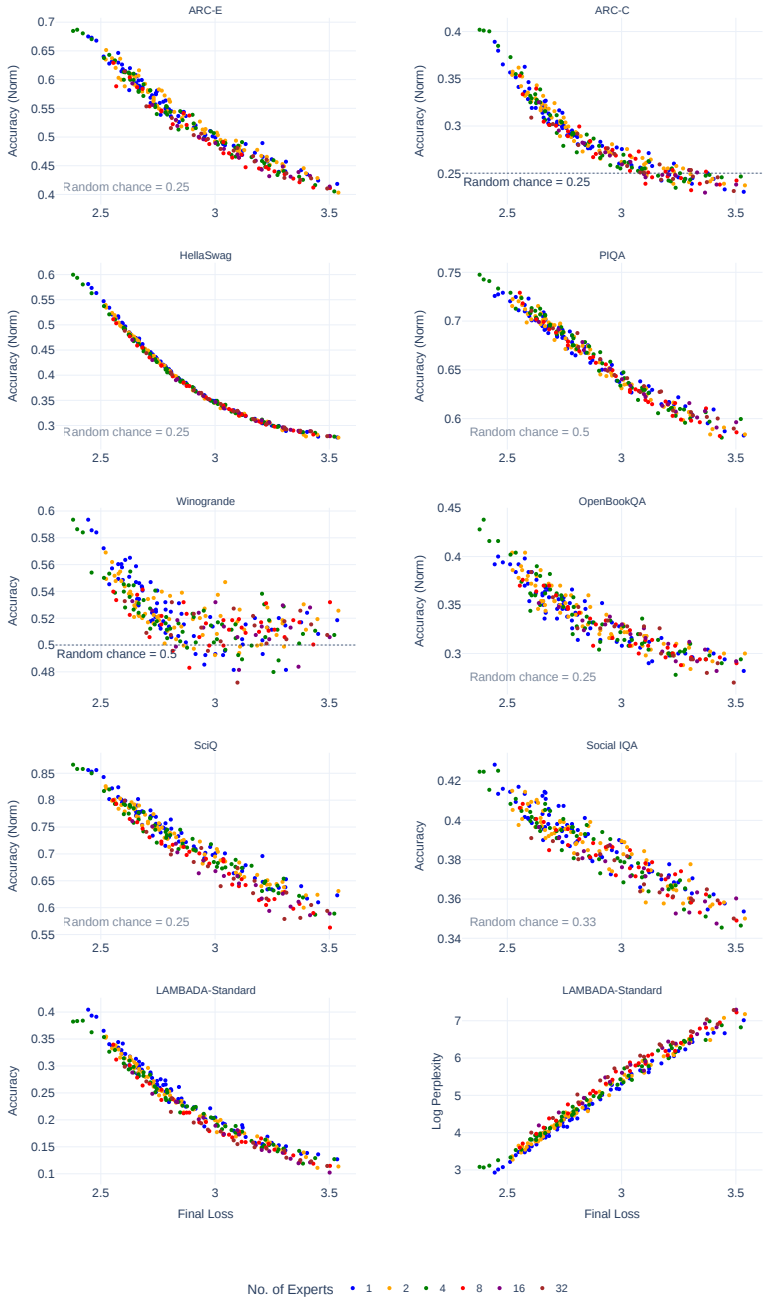


Figure 3.7: Downstream performance of the models.

Training Budget	**Experts**	$N_{\mathbf{act}}^{\mathbf{opt}}$ **(90% interval)**	$D^{\mathbf{opt}}$ **(90% interval)**
1×10^{20}	1	1.5B–2.1B	7.9B–11.1B
	2	1.4B–1.8B	9.1B–12.1B
	4	1.2B–1.6B	10.3B–14.1B
	8	940.0M–1.5B	11.4B–17.7B
	16	732.7M–1.3B	12.6B–22.8B
	32	559.5M–1.2B	13.7B–29.8B
1×10^{21}	1	4.5B–7.8B	21.4B–37.3B
	2	4.1B–7.3B	22.7B–40.9B
	4	3.5B–7.0B	23.7B–47.6B
	8	2.8B–6.8B	24.7B–60.1B
	16	2.0B–6.6B	25.4B–81.8B
	32	1.6B–6.5B	25.7B–104.1B
1×10^{22}	1	13.3B–30.0B	55.5B–125.3B
	2	12.1B–30.0B	55.6B–137.3B
	4	10.3B–30.6B	54.5B–161.4B
	8	8.0B–31.5B	52.9B–207.2B
	16	5.9B–32.9B	50.6B–284.4B
	32	4.4B–34.5B	48.4B–380.6B

Table 3.5: Bootstrap intervals for the optimal $N_{\text{act}}^{\text{opt}}$ and D^{opt} across training budgets and expert counts.

3.13 Learning Rate Scaling Fit

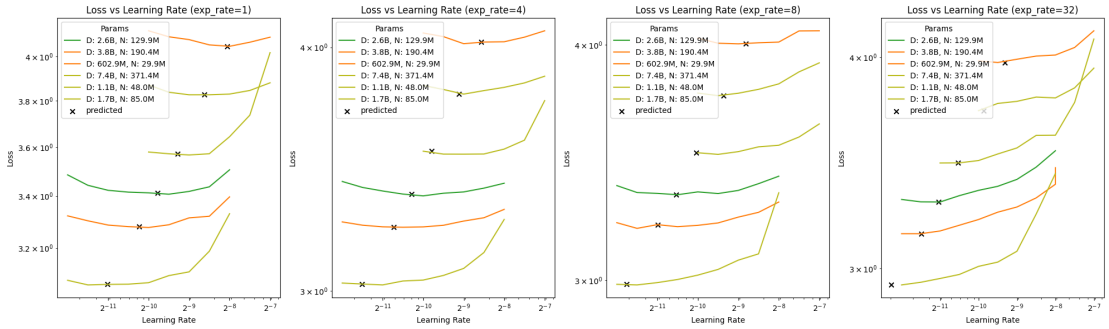


Figure 3.8: Visualization of the fit ($E \in \{1, 8\}$) of our LR scaling rule, interpolation ($E = 4$) and extrapolation ($E = 32$).

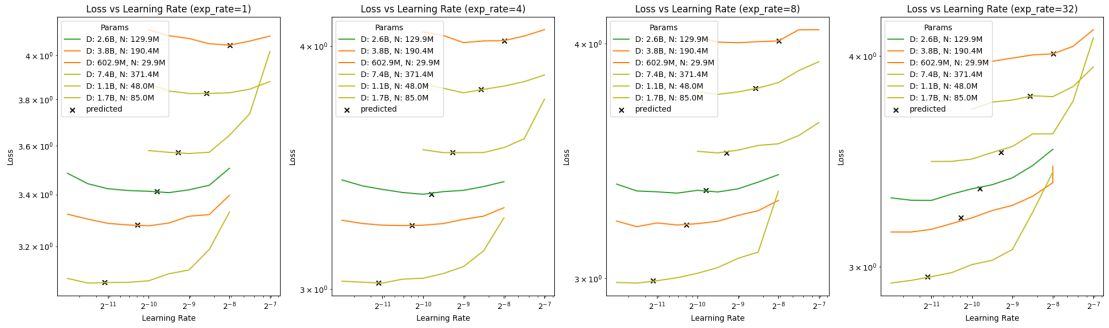


Figure 3.9: Ablation for the LR scaling rule fit without considering the number of experts E. While performance on the training set ($E \in \{1, 8\}$) looks acceptable, the extrapolation on $E = 32$ is clearly suboptimal, validating the need for considering E.

3.14 Experiments Listing

N_{total}	$\text{N}_{\text{attn\_heads}}$	N_{blocks}	d_{model}	N_{act}	E	D
5.0B	16	16	1024	321M	32	16.0B, 8.0B, 4.0B, 2.0B, 1.0B, 500M
3.8B	28	28	1792	1.3B	4	11.1B, 5.6B, 2.8B, 2.0B
3.3B	11	21	1408	683M	8	16.0B, 8.0B, 4.0B, 2.0B, 1.0B, 500M
3.0B	26	26	1664	1.1B	4	80.0B, 64.0B, 48.0B, 32.0B, 16.0B, 8.0B, 4.0B, 2.0B, 1.0B
2.7B	36	36	2304	2.7B	1	9.2B, 5.5B, 2.8B, 2.0B, 1.4B, 980M
2.6B	30	30	1920	1.6B	2	5.4B, 2.7B
2.6B	16	16	1024	321M	16	16.0B, 8.0B, 4.0B, 2.0B, 1.0B, 500M
2.2B	28	28	1792	1.3B	2	18.6B, 11.1B, 5.6B, 4.0B, 2.8B, 2.0B
2.1B	12	12	768	169M	32	8.0B, 4.0B, 2.0B, 1.0B, 500M
2.1B	10	16	1280	469M	8	32.0B, 16.0B, 8.0B, 4.0B, 2.0B, 1.0B
1.9B	22	22	1408	709M	4	35.3B, 12.2B, 10.6B, 7.7B, 5.3B, 3.8B
1.8B	11	21	1408	683M	4	8.0B, 16.0B, 4.0B, 2.0B, 1.0B, 500M
1.8B	26	26	1664	1.1B	2	16.0B, 8.0B, 4.0B, 2.0B, 1.0B, 500M
1.6B	30	30	1920	1.6B	1	5.4B, 2.7B
1.4B	16	16	1024	321M	8	16.0B, 8.0B, 4.0B, 2.0B, 1.0B, 500M
1.3B	28	28	1792	1.3B	1	6.5B, 3.3B, 18.6B, 11.1B, 5.6B, 4.0B, 2.8B, 2.0B
1.3B	10	10	640	118M	32	4.0B, 2.0B, 1.0B, 500M
1.2B	10	16	1280	469M	4	32.0B, 16.0B, 8.0B, 4.0B, 2.0B, 1.0B, 500M
1.1B	12	12	768	169M	16	8.0B, 4.0B, 2.0B, 1.0B, 500M
1.1B	26	26	1664	1.1B	1	14.0B, 12.0B, 10.0B, 80.0B, 64.0B, 48.0B, 32.0B
1.1B	26	26	1664	1.1B	1	16.0B, 8.0B, 4.0B, 2.0B, 1.0B, 500M
1.1B	22	22	1408	709M	2	3.8B, 49.8B, 24.9B, 12.5B, 6.2B, 3.1B, 1.6B, 778M
1.1B	22	22	1408	709M	2	21.8B, 18.7B, 15.6B, 35.3B, 12.2B, 10.6B, 7.7B, 5.3B
1.1B	18	18	1152	426M	4	31.0B, 25.9B, 20.7B, 10.4B, 5.2B, 2.6B, 1.3B
1.1B	11	21	1408	683M	2	32.0B, 16.0B, 8.0B, 4.0B, 2.0B, 1.0B, 500M
890M	24	24	1536	890M	1	9.9B, 5.0B
850M	20	20	1280	555M	2	16.0B, 8.0B
774M	16	16	1024	321M	4	16.0B, 8.0B, 4.0B, 2.0B, 1.0B, 500M
709M	22	22	1408	709M	1	35.3B, 12.2B, 10.6B, 7.7B, 5.3B, 3.8B, 12.5B, 6.2B
705M	10	16	1280	469M	2	32.0B, 16.0B, 8.0B, 4.0B, 2.0B, 1.0B, 500M
683M	11	21	1408	683M	1	32.0B, 16.0B, 8.0B, 4.0B, 2.0B, 1.0B, 500M
671M	10	10	640	118M	16	4.0B, 2.0B, 1.0B, 500M
664M	8	8	512	79M	32	2.0B, 1.0B, 500M
615M	12	12	768	169M	8	8.0B, 4.0B, 2.0B, 1.0B, 500M
555M	20	20	1280	555M	1	16.0B, 8.0B
472M	16	16	1024	321M	2	16.0B, 8.0B, 4.0B, 2.0B, 1.0B, 500M
469M	10	16	1280	469M	1	32.0B, 16.0B, 8.0B, 4.0B, 2.0B, 1.0B, 500M
376M	10	10	640	118M	8	4.0B, 2.0B, 1.0B, 500M
362M	8	8	512	79M	16	2.0B, 1.0B, 500M
360M	12	12	768	169M	4	8.0B, 4.0B, 2.0B, 1.0B, 500M
321M	16	16	1024	321M	1	16.0B, 8.0B, 4.0B, 2.0B, 1.0B, 500M
289M	11	11	704	142M	4	4.5B, 2.3B, 1.1B
285M	9	9	576	97M	8	3.3B, 1.7B
282M	13	13	832	201M	2	6.4B, 3.2B, 1.6B, 800M
233M	12	12	768	169M	2	8.0B, 4.0B, 2.0B, 1.0B, 500M
228M	10	10	640	118M	4	4.0B, 2.0B, 1.0B, 500M
211M	8	8	512	79M	8	2.0B, 1.0B, 500M
169M	12	12	768	169M	1	8.0B, 4.0B, 2.0B, 1.0B, 500M
154M	10	10	640	118M	2	4.0B, 2.0B, 1.0B, 500M
135M	8	8	512	79M	4	2.0B, 1.0B, 500M
118M	10	10	640	118M	1	4.0B, 2.0B, 1.0B, 500M
98M	8	8	512	79M	2	2.0B, 1.0B, 500M
79M	8	8	512	79M	1	2.0B, 1.0B, 500M

Chapter 4

Different Rates for Different Weights: Decoupled Relative Learning Rate Schedules

4.1 Introduction

The learning rate is a crucial hyperparameter in Deep Learning, determining the size of the steps that the optimization algorithm takes when updating model parameters during training. In the context of Transformers, widely used for tasks in Natural Language Processing (NLP) and other areas, the learning rate significantly impacts the model's convergence and overall performance. While higher learning rates, with larger updates to the model, may generally converge faster, they can also introduce instabilities. Therefore, the learning rate must be carefully chosen to balance the speed and stability of the training process.

At the same time, modern Deep Learning architectures are not homogeneous, with different parts having distinct structures, serving varied purposes, and exhibiting unique behaviors. Importantly, they also have individual training dynamics. As seen in Figure 4.1, weight updates for different model parts follow different patterns during training. This variability can result in components behaving differently depending on the training phase, which may be problematic in some cases. For example, in Mixture of Experts (MoE) models, the Router often stabilizes early in training, leading to deterministic routing to the Experts (Xue et al., 2024).

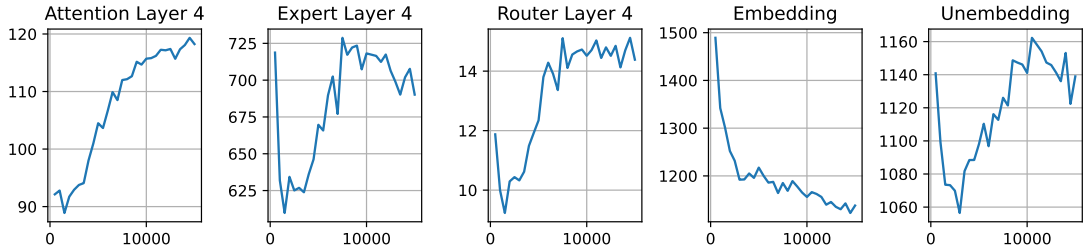


Figure 4.1: Magnitudes of weight updates for different components of the Transformer with MoE (trained without RLRS), right before applying learning rate.

Given the diversity of layers within a model, it is reasonable to expect that their requirements would vary, particularly when balancing training speed and stability. Despite this, a

uniform learning rate is often applied across all modules. A common practice, for example, is to reduce the learning rate for the whole model after the introduction of an MoE layer due to instabilities (Rajbhandari et al., 2022). As a result, hyperparameters are typically tuned for the entire network, even if the instabilities may originate from a single layer. In this work, we challenge the implicit assumption of a global learning rate. Motivated by the heterogeneity of Transformer's modules, we ask ourselves: can we improve the training procedure by tailoring the learning rate schedules to different model's components?

To answer this question, we decouple learning rates in Transformers and tune them individually for various model components—including Embedding, Unembedding, Attention, Feed-Forward, and, in the case of Mixture of Experts architecture, Router and Experts—we enhance the model's overall performance and stability by tailoring the learning rate to meet the specific needs of each component.

Commonly used adaptive optimizers, like AdamW (Loshchilov & Hutter, 2019), show the importance of well-adjusted learning rates during training. However, adaptive optimizers generally treat all layers in the same way and do not differentiate between them. Our proposed RLRS is used on top of an adaptive optimizer (AdamW in this work, but RLRS is independent of the optimizer), and we show that adjusting learning rate schedules for individual model components brings improvements.

Furthermore, we propose a straightforward scheme to adjust relative learning rate values that can be effectively scaled to models larger by orders of magnitude. This approach eliminates the need for extensive hyperparameter searches for larger models, resulting in significant computational savings and enhancing its practical applicability. (This could also be combined with Tensor Programs (Yang et al., 2022), see discussion in Section 4.6.1.)

In essence, we propose the following approach: first, relative learning rates, RLRS, should be tuned on a small model; later, the same RLRS can be reused when training the model's significantly larger counterpart. Our method is easy to implement, with no additional overhead required, apart from the relatively inexpensive hyperparameter search on the small model. While tailored to our specific training setup, the relative LR values configuration shared in this work have proven robust across a range of models sizes and training durations, making them an excellent starting point. Additionally, we provide an analysis showing how these values, obtained using automated methods, align with our intuitive understanding of Transformer training. In summary,

- We propose distinct, relative learning rate schedules (RLRS) tailored for different components of a Transformer model, optimizing each part individually for better overall performance.
- We demonstrate performance improvements of the introduced method in standard Transformers, and achieving even larger speedup in the case of Mixture of Experts (MoE) based models, highlighting the importance of relative learning rates for more complex models.
- We demonstrate that the hyperparameters tuned on small models extrapolate to larger models, showing that our approach generalizes effectively across different architecture sizes.

4.2 Decoupled Relative Learning Rate Schedules

We define a *decoupled learning rate* as a separate learning rate schedule for different layer types (also called parts, modules, or components). Decoupled learning rate schedules

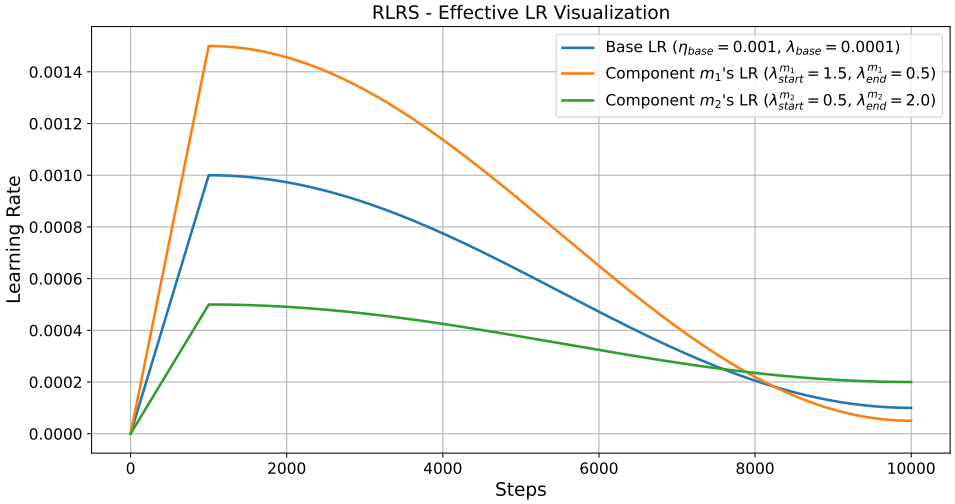


Figure 4.2: Visualization of distinct cosine learning rate schedules for each model component, compared against the base model's learning rate.

enable the learning procedure to focus on different components during various phases of a model's pretraining, facilitating a more targeted and efficient optimization process.

We specify decoupled learning rates following the structure of the cosine learning rate scheduler (Loshchilov & Hutter, 2016), widely used for training Large Language Models (LLMs) (Touvron et al., 2023a; Hoffmann et al., 2022). The cosine scheduler adjusts the learning rate over time according to a cosine function, starting with a high learning rate that gradually decays to a minimum value in a smooth, nonlinear manner.

We define the base learning rates, shared across modules, using parameters:

- *Base LR* (η_{base}) — the reference learning rate for the entire model. In a typical cosine schedule, it is the peak learning rate.
- *Base LR Final Fraction* (α_{end}) — the reference fraction of the base learning rate at the end of the training. In a typical cosine schedule, the final learning rate $\eta_{\text{end}} = \eta_{\text{base}} \times \alpha_{\text{end}}$.

The cosine scheduler adjusts the learning rate following a cosine curve over a specified number of iterations. The learning rate η_t at step t is computed using the cosine function: $\eta_t = \eta_{\text{end}} + \frac{1}{2}(\eta_{\text{start}} - \eta_{\text{end}})\left(1 + \cos\left(\frac{t}{T}\pi\right)\right)$, where t is the current step, and T is the total number of steps.

For each component m of the model, we further define a learning rate scaling factors *relative* to the base learning rates defined above:

- *Relative Start LR* (λ^m_{start}) — the scaling factor of the base learning rate at the beginning of training.
- *Relative End LR* (λ^m_{end}) — the scaling factor of the final learning rate at the end of training.

Thus, the *decoupled learning rates* η^m_{start} and η^m_{end} for a component m are defined as:

$$\eta^m_{\text{start}} = \eta_{\text{base}} \times \lambda^m_{\text{start}} \tag{4.1}$$

$$\eta^m_{\text{end}} = \eta_{\text{base}} \times \alpha_{\text{end}} \times \lambda^m_{\text{end}} \tag{4.2}$$

These values are adjusted for each Transformer component. In this work, we distinguish the following layer modules m: Embedding, Attention, Unembedding, and additionally, for dense models, the Feed-Forward layer, and for Mixture of Experts models, the Expert and Router layers.

Thus, for a given model, one needs to obtain the baseline learning rate η_{base} and the relative learning rates, λ^m_{start}, λ^m_{end}. In Section 4.5, we show how we find them in practice. However, in the next section, we demonstrate that we do not need to tune the relative learning rates for every model separately, as the same set of values remains robust across a range of model sizes.

4.2.1 Preserving Relative Values

Directly tuning relative learning rate values on large models may be impractical due to significant computational costs. To address this, we propose a method that fine-tunes these values on a smaller proxy model and then transfers them to a larger model. This approach significantly reduces the need for costly tuning on large models, offering substantial computational savings.

Our method, described in Algorithm 1, involves conducting a search for optimal values on smaller models under the assumption that these relative values extrapolate effectively to larger models. This search consumes only a fraction of the training time required for large models.

Algorithm 1 Relative Learning Rate Adjustment Procedure

1: Determine η_{base} for a small model.
2: For each module m, find relative values λ^m_{start} and λ^m_{end} on a small model.
3: Determine base learning rate η_{base} for the large model.
4: Apply relative learning rates λ^m_{start} and λ^m_{end} from the small model to the large model.

While we do not claim that λ^m_{start} and λ^m_{end} values are optimal for larger models, they are straightforward to use and yield substantial improvements, as shown in the next section. We leave the investigation of optimal extrapolation as future work.

Algorithm 1 presents a simple way to introduce relative learning rates independently of the base learning rate and to apply them to both small and large models. Moreover, as we see in Table 4.3, the base learning rate η_{base} tuned with relative rates rarely differs from the optimal learning rate for the baseline non-RLRS model with the same configuration. Therefore, if an already tuned learning rate for a large model is available, we can reuse it as our base learning rate η_{base} with relative schedules. Then further applying relative rates would yield substantial improvement without additional tuning on a large scale, which can be useful in practice.

We provide the details of our implementation of the hyperparameter search in Section 4.5 and Algorithm 2. Moreover, additional methods for adjusting the η_{base} on extrapolation are considered in Section 4.6.1.

4.3 Results

We first provide a detailed setup of our experiments. Then, we present details on how to find relative learning rates, and showcase the improvements obtained via the proposed decoupled relative learning rate schedules. Subsequently, we provide some interpretation of the presented relative rates. For reproducibility purposes, we will provide the complete code and configuration files used in our experiments in a public repository soon.

4.3.1 Experimental Setup

All models in this study are decoder-only Transformers trained on the C4 dataset (Raffel et al., 2023). We use the GPT-2 tokenizer (Radford et al., 2018a) and optimize with AdamW (Loshchilov & Hutter, 2019). Training follows a cosine decay schedule with linear warmup for the first 1% of steps. For stability, weights are initialized with a truncated normal distribution at a reduced scale, as suggested by Fedus et al. (2022). Mixed precision training is applied, with Attention and Router components computed at high precision. The models employ SwiGLU activation and Token Choice routing with 8 Experts, 1 of which is activated per token. Two auxiliary losses are used for the Router: a z-loss weighted at 0.001 (Zoph et al., 2022a) and load balancing weighted at 0.01 (Fedus et al., 2022). Compute-optimal training durations align with Hoffmann et al. (2022), calculated for MoE as 20 times the number of active parameters, excluding Embedding and Unembedding, as in chapter 2. We also report results on an overtrained $\text{MoE}_{8\times113\text{M}}$ model with a token-to-active-parameter ratio nearing 130. For extrapolations, we fine-tune base learning rates for both relative learning rates, RLRS, and the baseline on a grid of $1e{-}n$, $2e{-}n$, $5e{-}n$. By tuning learning rates separately, we ensure that performance differences arise from the effects of relative values rather than from base learning rates. For both dense and MoE models, the weight decay value has been optimized to 0.1, the initialization scale to 0.15, and *Base LR Final Fraction* (λ_{base}) to 0.04 for MoE and 0.06 for dense.

Type	Active Params	Total Params	d_{model}	n_{layers}	$n_{experts}$	BS	SL
$\text{Dense}_{34\text{M}}$	33.6M	33.6M	512	8		256	512
$\text{MoE}_{8\times34\text{M}}$	33.6M	210M	512	8	8	256	512
$\text{Dense}_{113\text{M}}$	113M	113M	768	12		256	512
$\text{MoE}_{8\times113\text{M}}$	113M	708M	768	12	8	256	512
$\text{Dense}_{906\text{M}}$	906M	906M	1536	24		384	1024
$\text{MoE}_{8\times906\text{M}}$	906M	5.67B	1536	24	8	384	1024

Table 4.1: Models used in this chapter. BS indicates batch size, and SL indicates sequence length. Active parameters exclude Embedding and Unembedding parameters.

Speed-up metric. In Tables 4.2 and 4.3, we present a speedup metric that measures how much faster a training process becomes when relative rates are applied. It is calculated using $(\frac{T_{\text{base}}}{T_{\text{relative}}} - 1) \times 100\%$, where T_{base} is the number of steps performed in the standard training with a base learning rate, and T_{relative} is the number of steps incurred until the loss of the training with the relative learning rate schedule exceeds the baseline loss. It is important to note that using this metric likely underestimates the improvement of our

method since for relative learning rate training steps, when we compute the speedup, the cosine schedule has not yet reached its end. We perform three runs for each configuration, and for each of them, we measure the loss per 1% of all training steps . The speedup is then calculated over the means of 3 runs. To reduce variance from random data seeds, we use 3 specified data seeds for each model type comparison.

Local search. Our approach involves optimizing a set of relative learning rates (RLRS) using a straightforward and scalable local search algorithm. While grid search could be employed for hyperparameter optimization, it demands precise boundary definitions and entails an exponential number of training runs, making it computationally expensive. Instead, we intentionally adopt a simple local search method to demonstrate that our approach delivers significant improvements even with basic hyperparameter tuning. This approach highlights the robustness of our method and leaves room for future exploration of more sophisticated optimization techniques.

This algorithm iteratively adjusts each hyperparameter by scaling its value with factors from a predefined set. If the adjustment improves performance, the change is retained, and the process repeats until no further improvements are observed. In our experiments, we optimized weight decay and initialization scale along with all RLRS values, ensuring a fair comparison. For the baseline, the same method was used to optimize the learning rate at the start and end of the cosine schedule, in addition to weight decay and initialization scale.

4.3.2 Tuning Small Models

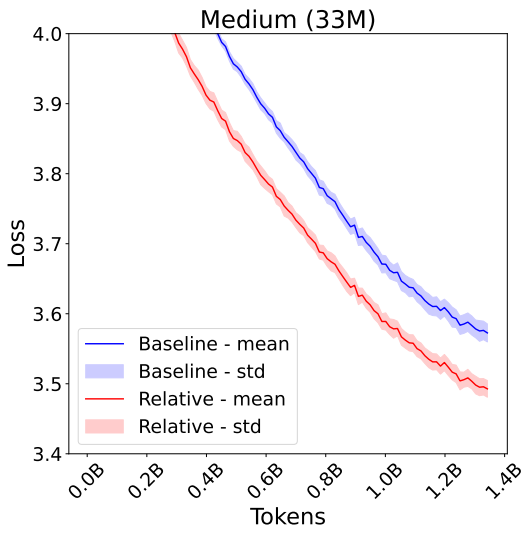


Figure 4.3: Training loss for Baseline vs. RLRS for $\text{MoE}_{8\times 33\text{M}}$ (210M total), averaged among 3 runs.

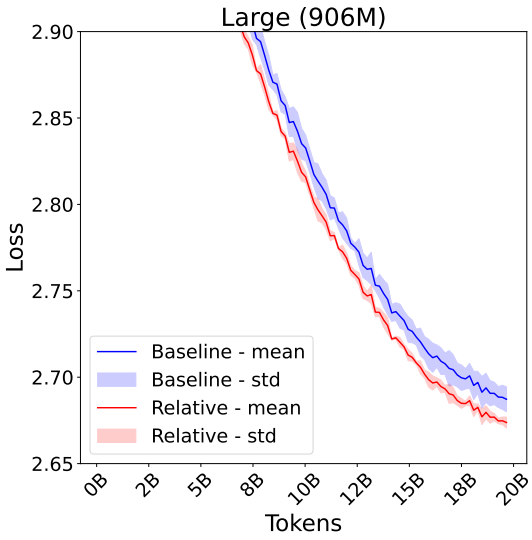


Figure 4.4: Training loss for Baseline vs. RLRS for $\text{MoE}_{8\times 906\text{M}}$ (5.67B total) with hyperparameters from $\text{MoE}_{8\times 33\text{M}}$, averaged among 3 runs.

We begin by presenting results obtained with relative learning rates on a small model. The compact size enables a broader search of hyperparameters; therefore, we propose performing the local search on a small model of choice. The details are given in Sec. 4.5. The relative learning rate values optimized on this smaller model then serve as reference

points for extrapolation to larger models.

Adjusting relative rates according to the proposed schedule results in significantly higher sample efficiency, which reduces the training time by up to 23% for MoE and by up to 17% for a dense Transformer (see Table 4.2). The exact relative learning rate values can be found in Section 4.4.

Type	LR Type	Base LR	Train Tokens	Total Params	Speedup
Dense_{34M}	baseline	2×10^{-3}	1.3B	34M	-
	relative	2×10^{-3}	1.3B	34M	17.2%
$\text{MoE}_{8\times34M}$	baseline	3×10^{-3}	1.3B	210M	-
	relative	3×10^{-3}	1.3B	210M	22.8%

Table 4.2: Using RLRS results in faster model convergence.

4.3.3 Extrapolation

In this section, we show the performance of large models trained with relative learning rate values tuned on small models. We demonstrate that, remarkably, this method improves the training speed of the large models as well.

We use models with 113M and 906M active parameters (in the case of MoE, 708M and 5.67B total parameters, respectively). The models are trained both in a compute-optimal and overtrained setting. As shown in Table 4.3, extrapolating the relative rates results in up to 19% faster training both in the case of MoE and dense model. Furthermore, the improvement is also noticeable in an overtrained MoE model with a token-to-active-parameter ratio that is almost 130. The most far-reaching extrapolations are performed on training runs more than 200× costly in terms of FLOPs as well as on models 27× bigger in terms of parameters than the training runs employed in the small-scale relative learning rates tuning. Even in these scaled-up scenarios, RLRS retains noticeably higher sample efficiency.

In Figure 4.5, we demonstrate that without fine-tuning on a large model, the transferred relative learning rates are still noticeably better than the baseline and generally close to the optimal value.

Type	LR Type	Base LR	Train Tokens	Total Parameters	Speedup
$\text{Dense}_{113\text{M}}$	baseline	1×10^{-3}	2.5B	113M	-
	relative	1×10^{-3}	2.5B	113M	19.0%
$\text{MoE}_{8\times113\text{M}}$	baseline	2×10^{-3}	2.5B	708M	-
	relative	1×10^{-3}	2.5B	708M	19.0%
$\text{MoE}_{8\times113\text{M}}$	baseline	1×10^{-3}	14B	708M	-
(overtrained)	relative	1×10^{-3}	14B	708M	14.6%
$\text{Dense}_{906\text{M}}$	baseline	5×10^{-4}	20B	906M	-
	relative	5×10^{-4}	20B	906M	8.7%
$\text{MoE}_{8\times906\text{M}}$	baseline	2×10^{-4}	20B	5.67B	-
	relative	2×10^{-4}	20B	5.67B	13.6%

Table 4.3: Speedup achieved when extrapolating relative learning rates, RLRS, to larger models.

4.4 Analysis

4.4.1 Interpreting Relative Learning Rates

In this section, we present the numerical results and trends for relative learning rates (RLRS) and analyze them with respect to each layer module. We prioritize MoE models, as RLRS yields more pronounced improvement in this setting. Interpretations provided below are an attempt to convey intuitions related to why RLRS works well that may be practical to other researchers and are not backed by experimental evidence. Although the values have been determined experimentally, they are often interpretable and aligned to counteract the existing issues of each component. While some of these ideas were also the motivation for this work and could be potentially used to guide the search of the optimal relative learning rates, we suspect that utilizing an exhaustive search will be a more reliable method in the long term.

Embedding. The relative learning rate λ_{start} starts high at 5 and decays to 0.6. This aggressive early training helps the Embedding stabilize quickly, as it influences the entire network. Later in the training process, the learning rate is reduced to prevent drastic changes in the Embeddings, ensuring the rest of the model can adjust accordingly. As seen in Section 6.4.5, this is the only layer that prefers adjustment of relative learning rate when increasing model size; that is, while other relative learning rates transfer without change, the Embedding's rate should be increased, but only at the start.

Unembedding. Unembedding handles the conversion of the model output into a probability distribution over the tokens in its vocabulary. We observe that, similarly to the Embedding, the relative learning rate gradually decreases toward the end of training. This behavior also aligns with observations in the literature that weights in the Unembedding may diverge, potentially causing instabilities later in the training process (Chowdhery et al., 2022; Zoph et al., 2022a), which would require reducing gradient values.

Router + Experts. Router (or gating network) plays a crucial role in determining which Expert networks are trained during the learning process. Xue et al. (2024) observed that the model often learns its routing decisions early in the pre-training phase, and these decisions remain largely fixed throughout training. Once a token is assigned to an Expert, it is rarely reassigned, making it difficult for the model to adapt to new or unseen data during later stages of training. Moreover, there have been conflicting guidelines in literature about suiting the learning rate for MoE models in comparison to their dense counterparts i.e. (Zoph et al., 2022a) argues that MoE benefit from higher learning rates. On the other hand, multiple studies analyze the instability of MoEs (Fedus et al., 2022), and the most straightforward approach to alleviate instabilities is to lower the learning rate for the whole model (Wortsman et al., 2023). We suppose that all of these aforementioned analysis are plausible and introduce many contradictory trade-offs that are hard to resolve under the assumption of a fixed learning rate for all modules. The gain from RLRS might come from alleviating these issues. Lowering the learning rate only at the beginning of the training (0.6) for the **Router** may mitigate instabilities and loss spikes caused by MoE, simultaneously delaying the early router stabilization. Increasing the relative learning rate at the end to 1, allow the model to benefit from the higher value while those issues are no longer present. Similarly, the relative learning rate of an **Experts** layer starts from the smallest value of 0.3 to aid stability when the Router is essentially random and prevent early expert specialization, which causes the router to freeze prematurely. It then increases to the highest relative value at the end (1.125), allowing the Experts to fine-tune and benefit from a high learning rate while the Router remains largely fixed.

Attention. In the Attention layers of the MoE model, the relative learning rate remains unchanged, making it unique in not benefiting from relative rates.

We summarize the Decoupled Learning Rate for both dense and MoE models in Table 4.4.

	Embedding	Unembedding	Router	Experts	Attention
start	5	0.6	0.6	0.3	1
end	0.6	0.4	1	1.125	1

Table 4.4: Relative learning rate values (λ) for MoE.

	Embedding	Unembedding	Feed-Forward	Attention
start	5	1	1	1
end	0.6	0.4	0.6	0.2

Table 4.5: Relative learning rate values (λ) for dense models.

4.4.2 Stability

As seen in Figure **??**, the baseline exhibits loss spikes that were absent with the relative schedules. This is also intuitive, as MoE models are considered unstable, thus requiring lower learning rates for optimal learning, which, however, affects the speed of training.

In our method, the learning rates for both the Router and the Experts start off relatively lower, while they are higher for other parts of the model, resulting in both better stability and convergence.

Large Transformer-based models frequently encounter instabilities, even when using hyperparameters that worked well for smaller models. Wortsman et al. (2023) demonstrate that instabilities in small models with a higher than optimal learning rate can be a good proxy measure for instabilities on a larger scale. Following that, we provide Figure **??** comparing the learning rate sensitivity of RLRS and the baseline. We can see that training with relative learning rates outperforms the baseline across various learning rates.

4.5 Finding Decoupled Relative Learning Rates

Our method involves determining a set of relative learning rates. While these hyperparameters could be optimized using a straightforward grid search, such a procedure requires carefully setting the search boundaries and involves an exponential number of training runs. In our experiments, we opt for a more scalable local search algorithm, which is described below.

Algorithm 2 Local Search

1: Iterate over the set of hyperparameters.
2: For a given hyperparameter, multiply its value by a factor from $\{\frac{1}{5}, \frac{2}{3}, \frac{3}{2}, \frac{5}{1}\}$
3: Run experiments, and if there is an improvement, adjust the hyperparameter value.
4: If any change has been made among all hyperparameters, return to Step 1.

We note that Algorithm 2 is a relatively simple optimization method, and we expect that a more complex alternative could either converge faster or find an even better set of hyperparameters. To ensure proper configuration, we optimized the weight decay and initialization scale along with all RLRS values. For the baseline, the same algorithm was used to find the learning rate at the start and at the end of the cosine schedule, along with weight decay and initialization scale.

4.5.1 Ablations

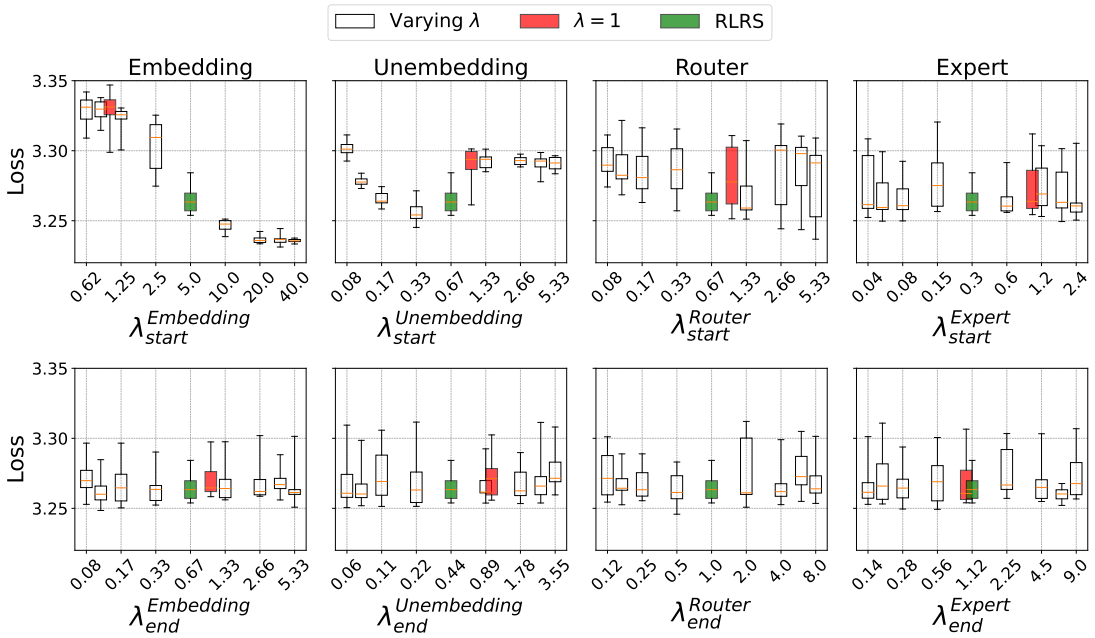


Figure 4.5: The performance of relative learning rates, λ_{small} extrapolated to larger models. We vary relative values for a given component while keeping all others fixed at their optimal values. The top row explores varying starting learning rates, bottom corresponds to end rates. The optimal λ_{small} (green) is compared to larger and smaller relative rates (white) and the baseline learning rate (red) at $\lambda = 1$. The plot shows standard boxplots with 1.5 interquartile range. Each boxplot is based on 10 experiments on $\text{MoE}_{8\times 113\text{M}}$.

We have shown that relative values tuned on a small model improve the training efficiency of their scaled-up counterparts. However, the question remains: how effective is this transfer, and can we obtain better results by further tuning relative values on the large model of choice? In Figure 4.5, we ablate each relative learning rate, λ^m_{start} and λ^m_{end} and show the importance of tuning the relative learning rates for individual modules. We conduct tests by evaluating the relative learning rate, transferred from the small model $\text{MoE}_{8\times 34\text{M}}$ to $\text{MoE}_{8\times 113\text{M}}$, with comparisons to its neighboring values. If the relative rate is not 1, we also include it as a baseline, accounting for cases where the value is not modified by any relative factor. It is important to note that the improvement brought by the method seems to largely come from the interactions between the relative rates for all the components rather than any specific module.

While in most cases the transferred rates λ^m_{start} and λ^m_{end} perform consistently well compared to the surrounding values, an interesting exception is the Embedding layer, which shows a clear preference to increase its relative learning rate when increasing the model size. This aligns with Yang et al. (2022), which studies models with increasing width and finds that Embedding is the only layer type whose learning rate should not be scaled down when increasing the model size. This only applies to the relative learning rate at the start, consistent with the Tensor Programs theory, which analyses the stability of the first iteration of the training process.

4.6 Related and Future Work

The literature on learning rates in Machine Learning, particularly for Transformers, highlights the importance of adaptive learning rate schedules. Stochastic Weight Averaging (SWA) (Izmailov et al., 2018) utilizes a modified learning rate schedule that applies a decaying learning rate during the initial training phase, followed by a constant rate for the remainder. In Sun et al. (2019), the authors introduce the layer-wise decay of the learning rate, which applies higher learning rates to the top layers and lower rates to bottom layers. A related concept, discriminative fine-tuning, is discussed in Howard & Ruder (2018). Furthermore, Everett et al. (2024) explores how various parameterizations and optimizers impact learning rate transfer in large-scale models and proposes a learning rate strategy per layer (depthwise).

4.6.1 Relation to Tensor Programs

Our method explores the transfer of relative learning rates; however, the base learning rate must still be independently tuned for the extrapolated model. Approaches such as Tensor Programs (Yang, 2020; Yang et al., 2022; Everett et al., 2024) propose parameterizations that facilitate the transfer of the base learning rate. By combining these two approaches, it may be possible to achieve a zero-shot transfer of RLRS.

While our methods share similarities with Tensor Programs and draw inspiration from them, our project has a distinct goal. We aim to identify implicit assumptions in the tuning process and decouple parameters to devise a scheme that enables Large Language Models (LLMs) to converge faster. Our extrapolations demonstrate that our optimization scheme depends on the architecture rather than the model size. This scheme is defined relative to the base learning rate, which must be tuned individually for each model size. Our method does not aim to facilitate learning rate transfer between different model sizes and is supported by experimental evidence. We do not mathematically examine the limits of parameter updates in a gradient descent step. A key difference is that our relative values change dynamically during training, and our goal is to enable the model to focus on different modules during pretraining.

4.6.2 Fine-Tuning

Fine-tuning allows users to adapt pre-trained Large Language Models (LLMs) to more specialized tasks. In traditional fine-tuning, certain model components are often "frozen" (effectively setting their relative learning rates to zero) to preserve learned knowledge while adapting other parts. Our proposed method introduces a more flexible approach, serving as a continuous alternative to freezing parameters. This enables fine-grained control over information transfer within specific components of the model. Consequently, our method could be particularly applicable to fine-tuning scenarios and complement existing methods that involve freezing parameters. Parameter-Efficient Fine-Tuning (PEFT) techniques, such as LoRA (Hu et al., 2021), address this by updating only a subset of parameters while freezing the rest. Our work aligns with more advanced methods like LoRA+ (Hayou et al., 2024), which select different learning rates for the adapter matrices, and AdaLoRA (Zhang et al., 2023), which adapts the rank of the LoRA matrices, providing enhanced flexibility in the fine-tuning process.

4.7 Conclusion

We have presented a method for decoupling learning rate schedules across different neural network components, removing the implicit assumption of homogeneity among them. We achieve a higher training speed through increased sample efficiency along with greater stability when using RLRS. Our method applies to any Transformer-based model and significantly enhances performance in Mixture of Experts (MoE) models. By tuning relative learning rates on smaller models, this approach can be used to economically achieve significant improvements in the training of order-of-magnitude larger models.

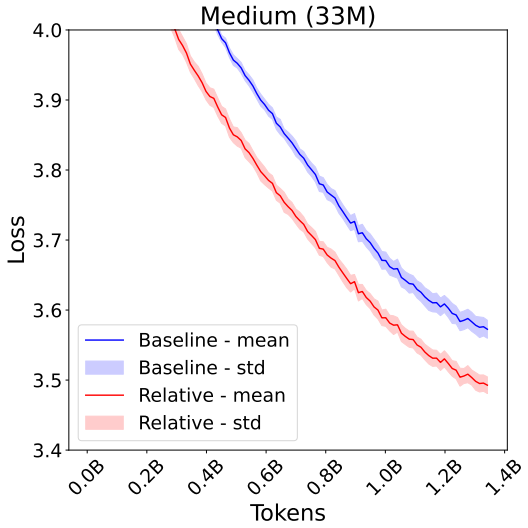


Figure 4.6: Training loss for Baseline vs. RLRS for $\text{MoE}_{8\times 33\text{M}}$ (210M total parameters), averaged among 3 runs.

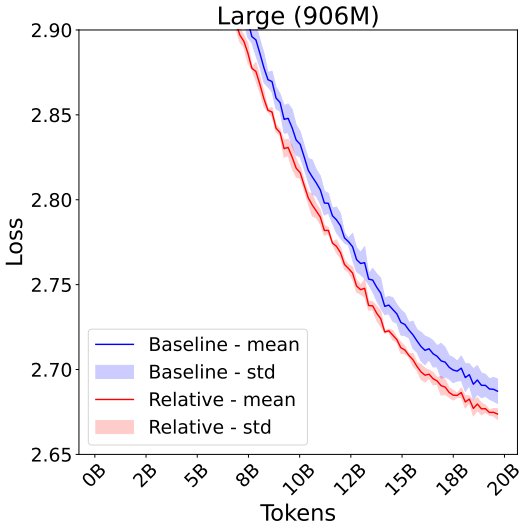


Figure 4.7: Training loss for Baseline vs. RLRS for $\text{MoE}_{8\times 906\text{M}}$ (5.67B total parameters) with hyperparameters extrapolated from $\text{MoE}_{8\times 33\text{M}}$, averaged among 3 runs.

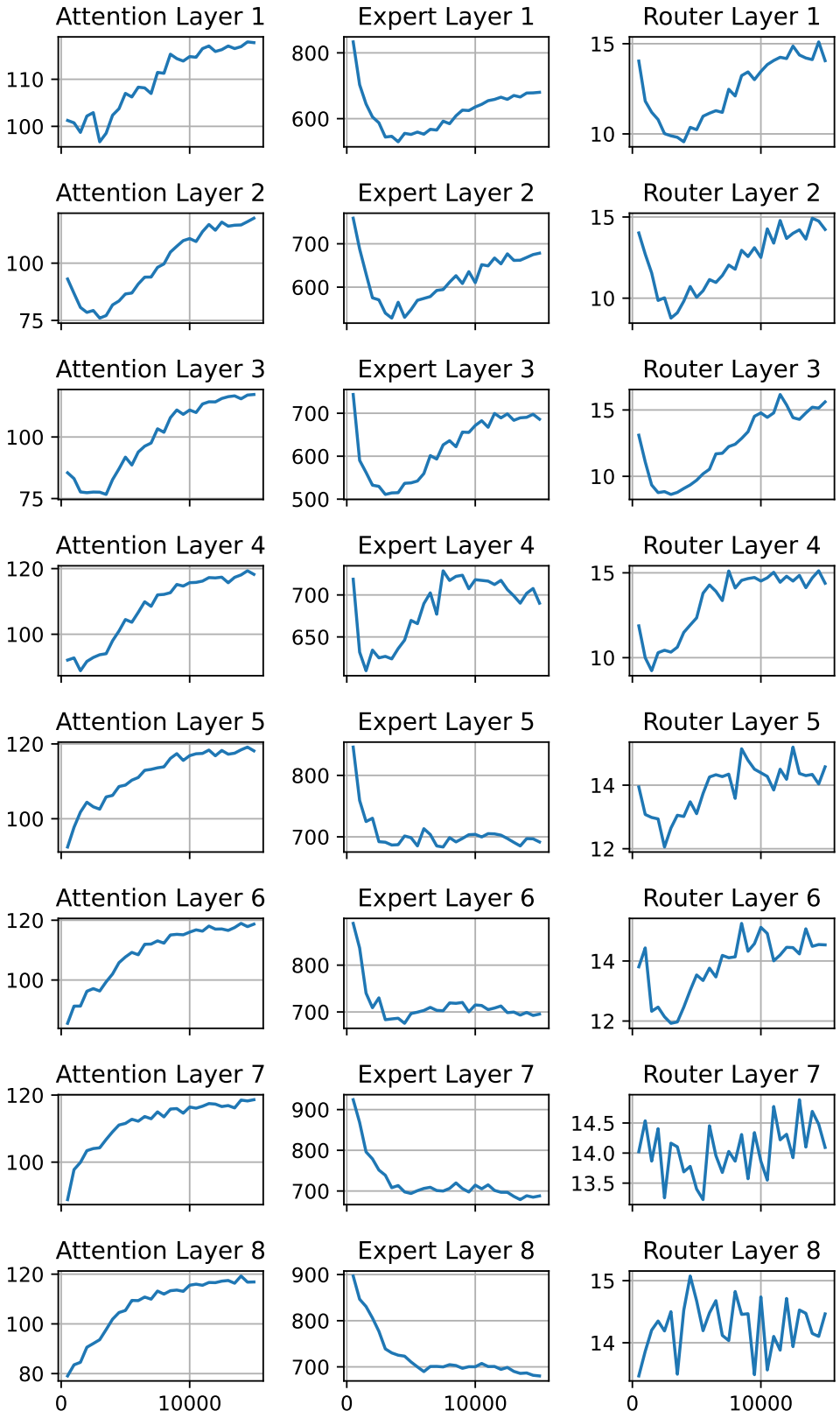


Figure 4.8: Magnitudes of weight updates for different components of all 8 layers of $\text{MoE}_{8\times 33\text{M}}$ trained without RLRS, right before applying learning rate.

Chapter 5

Mixture of Tokens: Continuous MoE through Cross-Example Aggregation

5.1 Introduction

Currently, state-of-the-art models based on MoE leverage sparsity by activating only a fraction of the parameters for each token (Sanseviero et al., 2023a). This allows the networks to increase the number of parameters by an order of magnitude while keeping the FLOPs per token roughly constant.

The aforementioned sparsity is made possible with a *router*, a small network that selects the best experts for each token. This makes the output of an MoE layer discontinuous with respect to its parameters, as only a subset of the experts is chosen for each token (this is typically done with a discrete *top-k* operation). The discontinuity and the resulting fluctuations of the router's decisions have been shown to hurt training efficiency (Dai et al., 2022; Chi et al., 2022) and are hypothesized to be a source of training instability in large MoE models (Mustafa et al., 2022; Puigcerver et al., 2023b). Conversely, existing continuous MoE architectures involve trade-offs, including the inability to scale (Muqeeth et al., 2023; Hazimeh et al., 2021), or incompatibility with autoregressive decoding (Puigcerver et al., 2023b).

This chapter introduces Mixture of Tokens, a novel, continuous Transformer architecture closely related to sparse Mixture of Experts. Similar to MoE, it can support large parameter counts without significant costs in FLOPs. The core idea behind our design is for each expert to process not individual tokens separately, but their combined representation.

This technique results in a continuous model that avoids the top-k operation. It requires no additional techniques commonly required in existing MoE designs (both sparse and continuous), such as load balancing losses, calculating solutions to optimization problems, or non-homogeneous training schedules (Hazimeh et al., 2021; Jaszczur et al., 2021; Dai et al., 2022). It is capable of scaling the parameter counts akin to sparse MoEs and is compatible with autoregressive language modeling and generation. Our analysis demonstrates a $3\times$ speedup over a dense baseline and improved stability over MoE.

In summary, our contributions are the following:

- Introducing the novel Mixture of Tokens (MoT), a continuous Mixture of Experts architecture that mixes tokens from different examples for joint processing.
- An analysis of scaling properties of Mixture of Experts models on multiple scales.

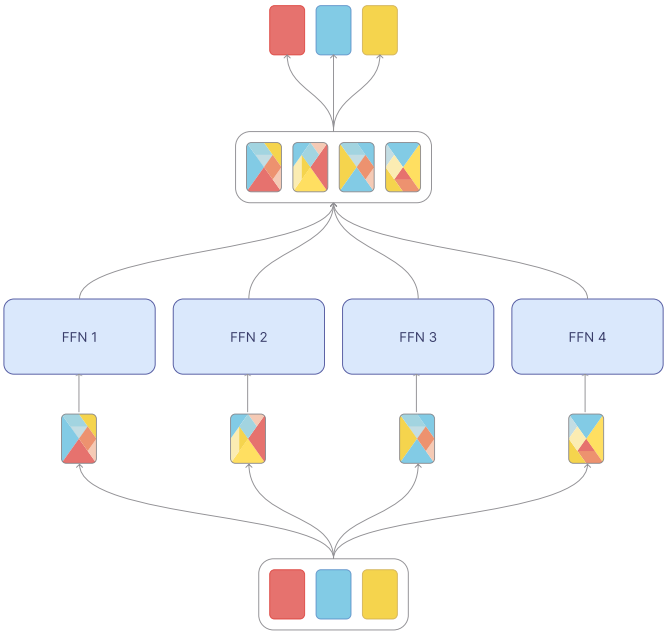


Figure 5.1: Mixture of Tokens: Each expert receives a unique mixture of tokens in the group. Mixing weights are determined by the controller, which is a fully connected layer (omitted for clarity). For a given token, its update is a linear combination of expert outputs, with the coefficients equal to the token's original mixing weights for each expert.

- Introducing transition tuning, allowing a pretrained MoT model to be tuned for sparse MoE inference if desired.

5.2 Related Work

5.2.1 Continuous Mixture of Experts

Continuous architectures serve an important role within the field due to their flexible and efficient nature. Hazimeh et al. (2021) were pioneers in introducing them in MoE by presenting continuous techniques for calculating encodings of the choice of an expert. In another approach, (Muqeeth et al., 2023) proposed a method in which they merge experts based on the weights of the router network. In a recent advancement, (Puigcerver et al., 2023b) proposed a continuous variant of MoE for the Vision Transformer, where patches are mixed only within each image.

5.2.2 From Hard to Soft Methods

From the very beginning of the Deep Learning field, there has been a shift from discrete functions toward continuous ones. The first perceptron McCulloch & Pitts (1943) used "all-or-none" activation, supposedly to align with propositional logic. This was later improved with soft activation functions, enabling gradient descent and multi-layer neural networks. Similarly, soft attention, introduced in Bahdanau et al. (2016), enabled RNNs to look at arbitrary input from the past while maintaining the ability to learn the selection

with standard gradient descent. This is in contrast to hard attention, which requires, e.g., reinforcement learning techniques. While hard attention could perform on par with soft attention Xu et al. (2015); Zohourianshahzadi & Kalita (2021), soft attention, with its simplicity of training, offered better trade-offs and was later used as the basic building block of the Transformer (Vaswani et al., 2017).

Mixture of Experts, introduced into Deep Learning by (Jacobs et al., 1991b,a; Shazeer et al., 2017), appears to be inherently a discrete function, since after all, the expert either processes a given token or it does not. However, similar to the transition from hard to soft attention, an expert in MoE can "attend" to a combination of tokens, taken as a weighted average. This results in a smooth, continuous model and facilitates more stable training.

5.3 Mixture of Tokens

The goal of this work is to develop an efficient, continuous architecture that retains the scalability of Mixture of Experts, while simultaneously omitting the top-k operation, which limits a token's exposure to different experts. An intuitive approach to achieving this is to route all tokens to all experts, but this is computationally infeasible for large-scale pretraining. To overcome this constraint, the method explored in this work considers what happens not to an individual token but to a whole group of tokens instead. The main contribution of this work is the observation that allowing an expert to dynamically produce a continuous representation of the entire group of tokens, which is more lightweight to process than each token individually, yields positive results.

Algorithm 3 Mixture of Tokens layer

for each expert E **do**
  $\text{weights}_E \leftarrow \text{Softmax}\big(\text{Linear}(\text{tokens})\big)$
  $\text{mix} \leftarrow \sum_i \text{token}_i \cdot \text{weights}_{i,E}$
  $\text{output}_E \leftarrow E(\text{mix})$
end for
for each token i **do**
  $\text{update}_i \leftarrow \sum_E \text{output}_E \cdot \text{weights}_{i,E}$
end for

More specifically, in our design, an input batch is divided into groups of tokens and each group is processed independently. Given a group and a single expert, a scalar weight is produced for each token. The weights are then normalized and used to compute a linear combination of the tokens, which is used as the expert's input. The experts' outputs are used for token updates as follows: for each input token, its update is a linear combination of expert outputs, with the token's mixing weights for each expert as coefficients[1]. A diagram of our method is presented in Figure 5.1.

To see why this method is scalable, it is helpful to examine the relationship between the number of tokens in a group and the number of experts. Essentially, if these two quantities are equal, the total computation performed by the experts is the same as in the case of top-1 routing. This allows MoT to benefit from the same parameter scaling as seen in MoE, which we confirm empirically in Section 5.4.2.

[1]The authors note that an MoT layer admits an efficient vectorized implementation, where all meaningful computations are done with batched matrix multiplications.

5.3.1 Intuition Behind Our Method

As we mix tokens from multiple unrelated sequences, we do not expect the model to meaningfully use the information from one sequence to improve prediction in a different sequence. However, we hypothesize that this mixing (1) provides richer feedback (gradients) to train the model, especially the router, and (2) results in a smoother loss landscape, which is resistant to small perturbations in inputs and weights.

Intuitively, for a given expert, from the perspective of each token, the token receives a certain amount of update to its representation (in the residual stream) based on:

- Itself, producing a proper signal is expected to improve the token representation.
- Tokens other than itself, which are essentially random tokens from unrelated sequences. As these sequences are randomly sampled from the dataset, the impact of these tokens will point in random directions and, essentially, just add some amount of noise to the token update.

We generally expect neural networks to be resistant to a certain amount of noise added to them. Moreover, while the signal-to-noise ratio worsens for tokens with low expert weight, the expert weight also modulates the magnitude of the update. Therefore, the amount of noise added to the representation is limited.

We stipulate that MoT experts will learn to focus on a single token or a small number of tokens, thereby minimizing noise and approximating sparse MoE when optimal. However, other tokens will be assigned non-zero weight, allowing some information to flow to the router for each and every token-expert pair, unlike sparse MoE. Additionally, the output of MoT is more continuous, with small perturbations of input/weights corresponding to small changes in the output rather than large discrete jumps, as may occur in the case of sparse MoE.

5.3.2 More Mixtures per Expert

Building on the design described above, we experiment with feeding more than one mixture to each expert. Without further modifications, this approach would result in a linear increase in computational costs for each additional mixture processed. To avoid this added cost, MoT uses more experts, but each expert has a proportionally reduced hidden dimension. This way, each mixture is processed by a smaller expert, and the layer's total number of parameters, as well as the number of FLOPs used by all experts, remains approximately the same. We find that this design consistently improves MoT as the number of processed mixtures increases, just as it does in sparse MoE (Chapter 2) Likewise, the optimal granularity aligns roughly with the number of mixtures in this work.

5.3.3 Token Groups in Mixture of Tokens

The question of how token *groups* are decided within a batch is crucial for compatibility with autoregressive training and inference. The main insight here is that tokens from the same sequence cannot be placed in a single group, as the mixing operation would result in an information leak. Due to this restriction, MoT groups tokens from different examples according to their position in the sequence. Thus, all tokens within a group share the same position in their respective sequences. As mentioned above, to maintain a constant number of FLOPs per token, an increase in the number of experts means an equal increase in group size. An illustration of how grouping is done within a batch of tokens is shown in Figure 5.2.

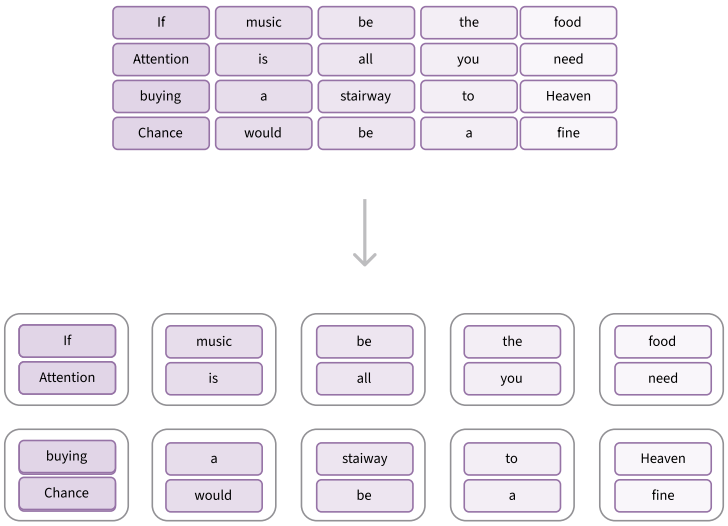


Figure 5.2: Each group consists of tokens with the same position in a sequence. In this example, the group size is 2. Note that the maximum possible group size equals the size of the batch.

5.3.4 Comparison with Other Mixture of Experts Architectures

Scaling The technique featured in Hazimeh et al. (2021) is based on a continuously differentiable sparse top-k router, which is a major advantage compared to the common top-k gating. However, this approach requires that all experts be utilized in a portion of training, rendering it computationally prohibitive for models with a large number of experts. The architecture based on merging experts proposed in Muqeeth et al. (2023) also offers an attractive, continuous alternative to top-k gating, yet the cost of merging all experts again scales linearly with the number of experts. To address this, the technique is applied once per sequence, which limits the expressive power of the final model.

Training stability Lepikhin et al. (2020) reported instabilities during the training of large MoE models, stemming from the inaccuracies when calculating router weights in low precision. To stabilize the training, they resorted to using full precision. Fedus et al. (2022) made progress in using mixed precision when training MoE by using selective high precision for gating. When comparing Lepikhin et al. (2020); Fedus et al. (2022) to MoT, an advantage of our technique emerges - it is more robust to training in lower precision than other methods. We conjecture, that this is due to the merging mechanism being less susceptible to rounding errors than gating in sparse MoEs.

Token dropping Token dropping is a phenomenon where tokens do not receive an update from any expert. This can happen when the expert was selected by too many tokens in a batch Fedus et al. (2022); Lepikhin et al. (2020); Zoph et al. (2022b) or, in the case of routing experts to tokens, when a token is not selected by any expert (Zhou et al., 2022). Existing techniques to combat this phenomenon offer a partial solution, yet the problem persists. In contrast, tokens in MoT are part of every mixture produced within their group; hence, they always receive an update.

Auto-regressive decoding Mixture of Tokens is based on the concept of merging tokens before they are processed by an expert. In the concurrent work, an encoder-only design of a similar nature is featured Puigcerver et al. (2023b). The technique is based on merging patches within an image for vision models, i.e., within a single sample. This

should be contrasted with MoT, which merges tokens from different sequences within a batch. This crucial difference allows MoT to be compatible with autoregressive training and inference.

Time complexity The time complexity of our approach is identical to that of the Token Choice and Expert Choice methods. In all cases, the cost of computing routing logits is of order $O(d_{model} \cdot N_{experts} \cdot N_{tokens})$.

5.4 Experiments

The focus of this work is to investigate the efficiency of Tokens on autoregressive language modeling. To measure model quality, we pretrain models for a fixed number of tokens and compare final perplexity in accordance with existing MoE literature Du et al. (2022); Fedus et al. (2022). In all experiments, the models are trained on the C4 dataset[2] (Raffel et al., 2023) and use the GPT-2 tokenizer. Unless specified otherwise, we use mixed precision, where all heavy computation is done in bfloat16, whereas the optimizer state and weights are kept in full precision. To study the stability of our model, we experiment with training fully in reduced precision.

Our main result is a substantial speed-up of MoT models compared to dense Transformers (Figure 5.6) and results comparable to sparse MoEs (Figure 5.5). What follows is the analysis of the scaling properties of the MoT architecture with respect to the number of parameters (Figure 5.3) and the number of mixtures sent to each expert (Figure 5.4). We investigate the model's performance in low precision in order to simulate training instability and find that MoT is less susceptible to instabilities arising from low-precision training. Lastly, we show the connection between MoT and MoE, by spending an additional fraction of pretraining compute to effectively transform a MoT model into a Token Choice model (Section 5.4.4).

5.4.1 Model Architecture

The base of our experiments is a decoder-only Transformer based on GPT-2 Radford et al. (2019b). We conduct experiments on two model scales: a 77M Medium model and a 162M Base model (refer to Section 5.5 for hyperparameters and training details). To obtain a Mixture of Tokens model, we replace the second half of the Feed-Forward layers in the Transformer with MoT layers. Because, similar to MoE models, the FLOPs and parameter counts in MoT are decoupled, we indicate the model architecture by its dense counterpart in terms of the number of FLOPs and, separately, the number of experts (or equivalently, group size). To this end, a MoT-Medium/32E model is one that uses the same number of FLOPs as a Medium (77M) Transformer model but uses 32 experts in MoT layers.

As outlined in Section 5.3.2, Medium/32E/4 signifies a model employing MoT layers with $32 \cdot 4$ small experts, which add up to the same number of parameters as 32 normal experts.

In addition to using the Transformer as a baseline, we also compare against Token Choice (Fedus et al., 2022) and Expert Choice (Zhou et al., 2022) as sparse MoE baselines. Given that Expert Choice is sensitive to the size of the batch, in order to avoid discrepancy between training and inference, we group tokens prior to routing in training Expert Choice models.

[2]https://huggingface.co/datasets/c4, dataset licensed under ODC-By.

5.4.2 Scaling Results

Mixture of Tokens models demonstrate strong scaling properties with respect to the number of parameters. As seen in Figure 5.3, increasing the number of experts in MoT layers while using the same compute budget yields consistent improvements. All MoT models are a strict improvement over the Transformer. The figure also includes an ablation experiment, where the mixing weights are fixed to $1/n$, where n is the group size. This corresponds to a uniform mixing strategy; the performance of that model clearly suffers, confirming that MoT layers learn non-trivial mixing strategies.

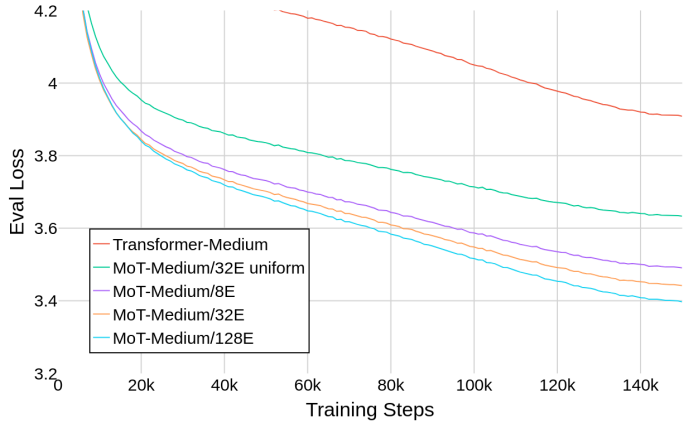


Figure 5.3: Scaling with respect to the number of parameters. Also featured are the Transformer baseline and an MoT model with a non-learnable, uniform routing strategy.

The increased number of token mixtures described in Section 5.3.2 represents another axis of scaling for MoT models, once again exhibiting consistent improvements. We hypothesize that this phenomenon is due to two mechanisms: First, the model becomes more expressive with a larger number of smaller experts. Second, the model can allocate its focus (the mixing weights) more flexibly to more important tokens while reducing the updates for trivial ones.

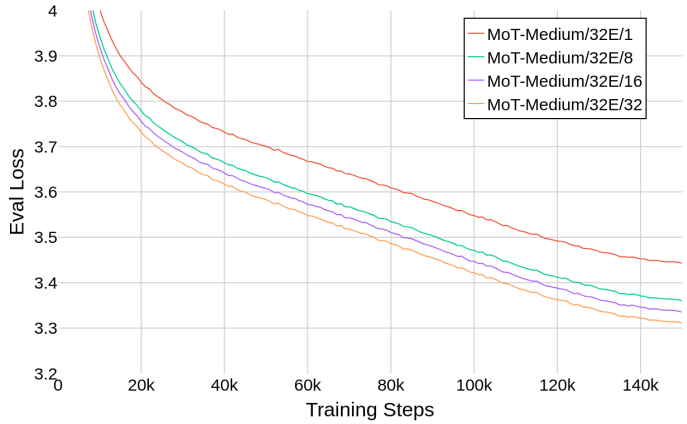


Figure 5.4: Scaling with respect to the number of token mixtures.

5.4.3 Comparison with the Transformer and Sparse MoEs

Crucially, the performance of Mixture of Tokens is comparable to that of the strong Mixture of Experts baselines (Figure 5.5). An increased number of mixtures allows it to compete with both Expert Choice and Token Choice architectures. As the sparse routing is hypothesized to contribute to training instabilities in large sparse models, Mixture of Tokens, being continuous, presents a promising alternative. To investigate training instabilities at the scale we experiment on, we trained models entirely in bfloat16, as opposed to the mixed precision used in all other experiments. The results confirm that MoT is more resistant to lower precision training: as the precision of training decreases, the performance of Expert Choice drops below that of Mixture of Tokens, despite the former attaining better perplexity using mixed precision. We find this to be evidence of the architecture's potential for stable training at higher model scales. See Table 5.1 for details.

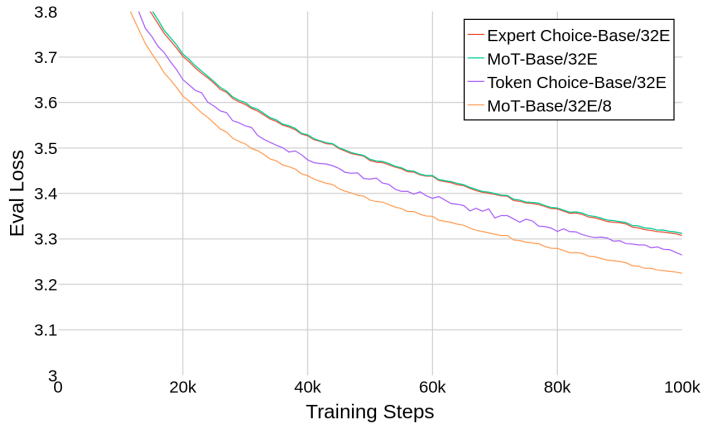


Figure 5.5: Comparison of MoT and sMoE architectures. An increased number of smaller experts allows MoT to match the performance of the best sMoE model. Due to computational constraints, the models were trained for 100K steps.

Finally, we combine our findings on MoT scaling properties to train our most efficient MoT model and compare it to the Transformer baseline (Figure 5.6). The result is a model that achieves the final loss of the baseline in one-third of the training steps. This represents a 3× improvement in terms of the compute budget.

	MoT-Medium/32E	Expert Choice-Medium/32E
Mixed Precision	3.442 (± 0.002)	**3.420 (± 0.002)**
bf16 only	**3.661 (± 0.007)**	3.728 (± 0.044)

Table 5.1: Comparison of training result loss comparison. MoT performs better in the bfloat16-only setting. Learning rates were separately tuned in lower precision for both EC and MoT. Results are averaged over 3 random seeds.

5.4.4 Transition Tuning

Mixture of Tokens suffers from a drawback common to MoEs, namely, it does not support unbatched inference. This is a direct consequence of its design - in the forward pass, it groups several tokens from different examples in the batch. With the growing adoption

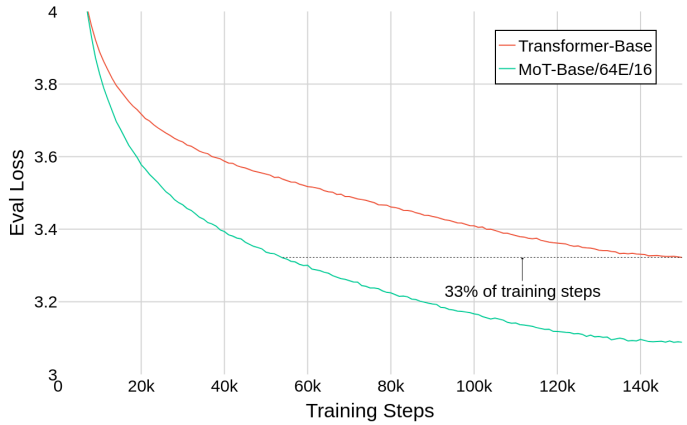


Figure 5.6: Our best MoT model reaches the final loss of the baseline in just 33% of the compute budget.

of Large Language Models on consumer hardware (Touvron et al., 2023a; Cerisara, 2023), this lack of support could hinder the architecture's wider adoption. While a Mixture of Tokens with a group size of one is technically possible, in order to keep FLOPs constant, the layer would need to trivially reduce to a standard Transformer MLP.

To address this issue, we demonstrate that the weights learned by the Mixture of Tokens can be used to directly initialize a Token Choice model of the same specifications (number of experts and expert size). The layer responsible for producing mixing weights is utilized to initialize the sparse router. In order to mitigate the difference in performance that is caused by this change in architecture, we train the entire new model (no weights are frozen) for 10% of the total pretraining steps of the original model in order to recover the original model's performance (measured in eval loss). We call this technique *transition tuning*. This way, it is possible to train with Mixture of Tokens and enjoy unbatched generation at inference time. We hypothesize that this pipeline would be especially attractive in setups where having parts of the model train in higher precision is impossible, e.g., on specialized, low-precision hardware. The results are presented in Figure 5.7.

5.5 Training Hyperparameters

All models were trained using mixed precision unless explicitly stated otherwise. We conducted all experiments with a batch size of 256 and a context length of 256 for 150K training steps (unless explicitly stated), resulting in a total of 10B training tokens. We used the AdamW optimizer with default hyperparameters. When necessary, we adopted a Fully Sharded Data Parallel approach from PyTorch to parallelize training across multiple machines. Learning rates were tuned separately based on model size and architecture. The optimal learning rate for Transformers was 1e-3 for Medium models and 4e-4 for Base models, while for both MoT and MoE, they were 7e-4 for Medium models and 2e-4 for Base models.

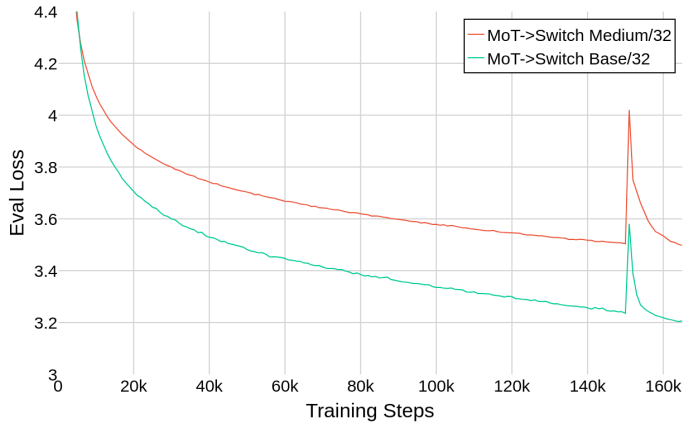


Figure 5.7: Transition tuning: The first 150K steps of the model are completed using the Mixture of Tokens architecture. Then, a new Token Choice model is initialized with weights from the MoT model, and the model trains for an additional 15K steps to recover performance. The spike in loss results from the sudden change of architecture.

Model	Experts	Expert size	Group size	Total params	Blocks	d_{model}	d_{ff}	#att. heads
Transformer-Medium	-	-	-	77M	8	512	2048	8
MoT-Medium/32E	32	2048	32	336M	8	512	-	8
MoT-Medium/32E/8	256	256	32	337M	8	512	-	8
Transformer-Base	-	-	-	162M	12	768	3072	12
MoT-Base/32E	32	3072	32	520M	12	768	-	12
MoT-Base/64E/16	1024	192	64	977M	12	768	-	12

Table 5.2: Training hyperparameters. The table provides example models featured in our experiments. All remaining models can be derived from this table.

5.6 Downstream Evaluation

When trying to predict how specific changes to the model architecture will impact large-scale models, comparing perplexity can provide a reliable indication of model improvements. However, for completeness, we also measured performance on several downstream tasks relevant at this model scale, comparing MoT-Medium to Transformer-Medium, without fine-tuning, in a zero-shot setting. In these evaluations, for MoT, a single evaluation query is included in a batch of 32, with the remainder of the batch comprised of random sequences from the C4 training dataset, ensuring it remains zero-shot. We observe predictable improvements with the Mixture of Tokens on tasks PIQA Bisk et al. (2019), HellaSwag Zellers et al. (2019), and ARC-e Clark et al. (2018), see Table 5.3.

	Transformer-Medium	MoT-Medium/32E/1	MoT-Medium/32E/16
PIQA	60.2	62.4	**65.8**
HellaSwag	27.3	31.1	**33.3**
ARC-e	35.5	37.3	**39.6**

Table 5.3: Performance of a medium-sized model on downstream benchmarks.

5.7 Compute Resources

Model	**GPU RAM**	**Time**	**GPUs**
Transformer-Base	40GB	32h 20m	1
MoT-Base/64E/16	40GB	33h 12m	2
MoT-Medium/128E	40GB	26h 36m	1
MoT-Medium/32E	40GB	23h 9m	1
MoT-Medium/8E	40GB	22h 31m	1
MoT-Medium/32E	40GB	20h 17m	1
Transformer-Medium	40GB	18h 48m	1
MoT->Switch Medium/32	40GB	35h 11m	1
MoT->Switch Base/32	40GB	17h 20m	4
MoT-Medium/32E/1	40GB	23h 9m	1
MoT-Medium/32E/8	40GB	24h 13m	1
MoT-Medium/32E/16	40GB	25h 36m	1
MoT-Medium/32E/32	40GB	28h 20m	1
Expert Choice-Base/32E	80GB	21h 12m	2
MoT-Base/32E	80GB	19h 38m	2
MoT-Base/32E/8	40GB	22h 10m	2
Token Choice-Base/32E	80GB	20h 19m	8

Table 5.4: Compute resources used for each experiment. All models were trained on NVIDIA A100 GPUs, with either 40 or 80 GB of RAM.

5.8 Limitations and Future Work

With the strong performance of MoT on medium-sized models, an obvious next step is to train larger models. This would present an opportunity to validate the stability results on larger models, where training instabilities are more common.

As with most Mixture of Experts models, the memory footprint of MoT layers is substantial. Scaled models require large amounts of RAM on specialized hardware for training, making their adoption expensive. To this end, an attractive future direction would be to investigate model distillation with Mixture of Tokens models.

In this work, we experimented only with text modality in an autoregressive manner. Other modalities, such as vision, heavily overlap with the approach presented in work concurrent to ours (Puigcerver et al., 2023b).

Lastly, both training and inference with MoT involve mixing different examples within a single batch. This mixing of tokens from different sequences and the requirement of

performing batched inference may be undesirable in some use cases. While performing unbatched inference is always inefficient with LLMs, as the memory throughput to access model weights becomes the bottleneck, unbatched inference still finds its uses. Even though transition tuning solves this problem, exploring different inference strategies might bring new insights.

5.9 Conclusions

In this work, we presented the Mixture of Tokens, a novel continuous Mixture of Experts architecture compatible with autoregressive decoding. This architecture scales to model sizes similar to sparse Mixture of Experts models, matches their performance, and is more resistant to training instabilities due to lower precision training. Moreover, we introduced transition tuning, a technique for initializing an MoE model with another pretrained MoE model of a different architecture, and showed that the new model achieves the performance of the original one using a fraction of the compute budget.

Chapter 6

MoE-Mamba: Efficient Selective State Space Models with Mixture of Experts

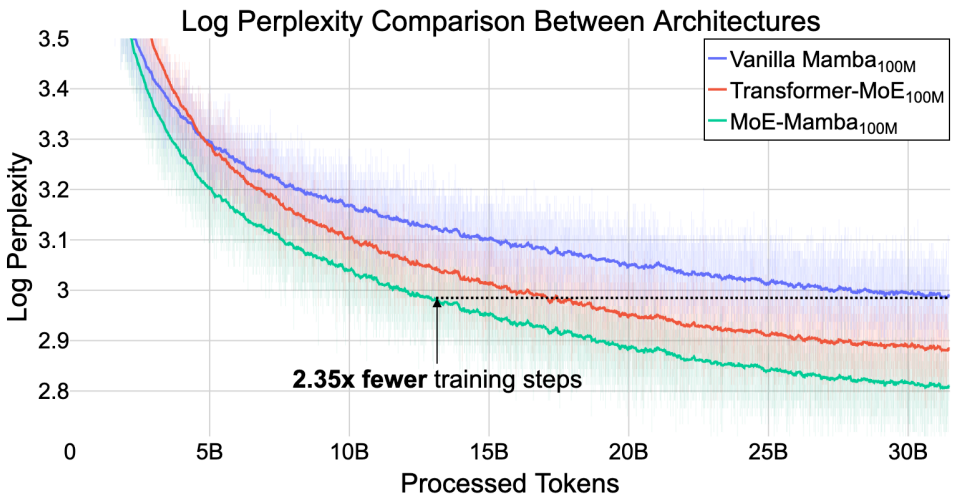


Figure 6.1: Log perplexity throughout the training. From top to bottom: $\text{Mamba}_{100\text{M}}$; $\text{Transformer-MoE}_{100\text{M}}$; $\text{MoE-Mamba}_{100\text{M}}$.

6.1 Introduction

The remarkable effectiveness of Large Language Models is primarily attributed to the Transformer architecture (Vaswani et al., 2017) and training on an internet-wide scale, e.g., (TogetherComputer, 2023). Yet, questions remain: Should Transformers be the only architecture used for LLMs? Can we scale language models even further, and if so, how can this be achieved?

Regarding the first question, State Space Models (SSMs), e.g., Gu et al. (2022b, 2021, 2022a); Gupta et al. (2022); Li et al. (2022); Ma et al. (2022); Orvieto et al. (2023); Smith et al. (2023), have been increasingly gaining attention. This recognition is due to their capability for linear-time inference, highly parallelized training, and strong performance in tasks requiring long-context processing, such as those illustrated by the Long Range Arena

(Tay et al., 2020). Notably, a recent addition to this category, Mamba Gu & Dao (2023), has shown impressive results through its selective mechanism and hardware-aware design, positioning it as a promising contender to the attention-based Transformer architecture.

Scaling is believed to be a critical factor in developing powerful AI systems (Sutton, 2019a). The Mixture of Experts (MoE) approach Jacobs et al. (1991b), a set of techniques that enables an increase in model parameters with minimal impact on computational demands, plays a significant role. Due to their sparse activation, MoEs can be efficiently scaled up to trillions of parameters, as demonstrated by Shazeer et al. (2017); Fedus et al. (2022). MoE variants Fedus et al. (2022); Du et al. (2022) are now routinely used in LLMs, as exemplified in the recent Mixtral model Jiang et al. (2024).

In this chapter, we advocate that to unlock the potential of SSMs for scaling up, they should be combined with Mixture of Experts (MoE). To this end, we introduce **MoE-Mamba**, a model that combines Mamba Gu & Dao (2023) with a Switch layer Fedus et al. (2022). MoE-Mamba enables efficiency gains of both SSMs and MoE, outperforming Mamba and Transformer-MoE, see Figure 6.1. Through comprehensive studies, we confirm that the effect is robust to the design choices and the number of experts. Our results indicate a very promising research direction that may allow scaling SSMs beyond tens of billions of parameters and compete with the largest SoTA language models.

In summary, our contributions are as follows:

- We introduce MoE-Mamba, a model that combines Mamba with a Mixture of Experts layer. MoE-Mamba enables efficiency gains of both SSMs and MoE while reaching the same performance as Mamba in 2.35× fewer training steps.

- Via comprehensive studies, we confirm that the improvement achieved by MoE-Mamba is robust to varying model sizes, design choices, and the number of experts.

- We explore and compare multiple alternative methods of integrating Mixture of Experts within the Mamba block.

6.2 Related Work

State Space Models and Related Attention-Free Architectures State Space Models (SSMs) Gu et al. (2022b, 2021, 2022a); Gupta et al. (2022); Li et al. (2022); Ma et al. (2022); Orvieto et al. (2023); Smith et al. (2023) form a family of architectures used for sequence modeling. Stemming from signal processing, these models can be seen as a combination of RNNs and CNNs Gu & Dao (2023). Although they potentially offer considerable benefits, a number of issues have been identified with SSMs Gu et al. (2022b), preventing SSMs from becoming the leading architecture in the task of language modeling. However, recent breakthroughs Gu et al. (2022b); Fu et al. (2023); Smith et al. (2023); Gu & Dao (2023), have allowed deep SSMs to be increasingly competitive against Transformers (Vaswani et al., 2017). In particular, Mamba Gu & Dao (2023), has shown impressive results through its selective mechanism and hardware-aware design, which allows scaling to billions of parameters while retaining computational efficiency and strong performance. Besides SSMs, numerous other architectures have been proposed that do not rely on the quadratic attention mechanism Zhai et al. (2021); Poli et al. (2023); Sun et al. (2023); Peng et al. (2023).

6.3 MoE-Mamba

In this section, we present the architecture details of our model, MoE-Mamba, see Figure 6.2. We start with a brief overview of the Mamba architecture, followed by a description of the MoE layer. Our main architecture is presented in Section 6.3.2, while sections 6.3.3 and 6.3.4 explore its variants and related approaches.

6.3.1 Preliminaries

Mamba Mamba (Gu & Dao, 2023) is a recently proposed SSM-based model that achieves remarkable, Transformer-like performance. By employing a work-efficient parallel scan, Mamba mitigates the impact of the sequential nature of recurrence, whereas fusing GPU operations removes the requirement to materialize the expanded state. Intermediate states necessary for backpropagation are not saved but recomputed during the backward pass, thus reducing memory requirements. The advantages of Mamba over the attention mechanism are especially prominent during inference, as not only is the computational complexity lowered, but also the memory usage is not dependent on the context length. Figure 6.3 shows the inner structure of a Mamba layer.

MoE Layer In our work, we follow the well-established Zhao et al. (2023a); Sanseviero et al. (2023b) and easy-to-implement Switch Transformer MoE design Fedus et al. (2022) and leave consideration of other MoE designs for future work.

We assume N_{experts} experts $\{E_i\}_{i=1}^{N_{\text{experts}}}$, each being a trainable feed-forward network with the same number of parameters. For each token embedding x, we calculate scores $h(x) = Wx \in \mathbb{R}^{N_{\text{experts}}}$, where W is a trainable linear projection. These are normalized using softmax:

$$p_i(x) = \frac{\exp\left(h(x)_i\right)}{\sum_{i=1}^{N_{\text{experts}}} \exp\left(h(x)_i\right)}.$$

Prior to Switch, top-k routing selecting $k > 1$ most suitable experts for each token was deemed necessary. However, Switch successfully simplifies previous MoE approaches by setting $k = 1$. Namely, the output of the MoE layer for x is given by:

$$y = p_I E_I(x),$$

where $I = \text{argmax}_i\, p_i(x)$.

During batched execution, e.g., in training, each batch contains N tokens. Following the standard procedure, in a case where the assignment of tokens to the experts is not perfect, i.e., some expert E_f is selected by more than N/N_{experts} tokens in the current batch, the excess tokens are dropped and not updated (capacity factor = 1). To further encourage an even distribution of tokens to experts, load balancing loss as described by Fedus et al. (2022) with weight $\alpha = 0.01$ is added to the training objective.

6.3.2 MoE-Mamba Architecture

The vanilla Mamba architecture consists of multiple Mamba blocks stacked one after another, with each layer's output being added to the residual stream; see Figure 6.2. In MoE-Mamba, we interleave Mamba layers with MoE layers (see Figure 6.2). Note that the vanilla Mamba does not use feed-forward layers.

In this way, MoE-Mamba separates unconditional processing of every token by the Mamba layer - which can efficiently integrate the whole sequence context into an internal

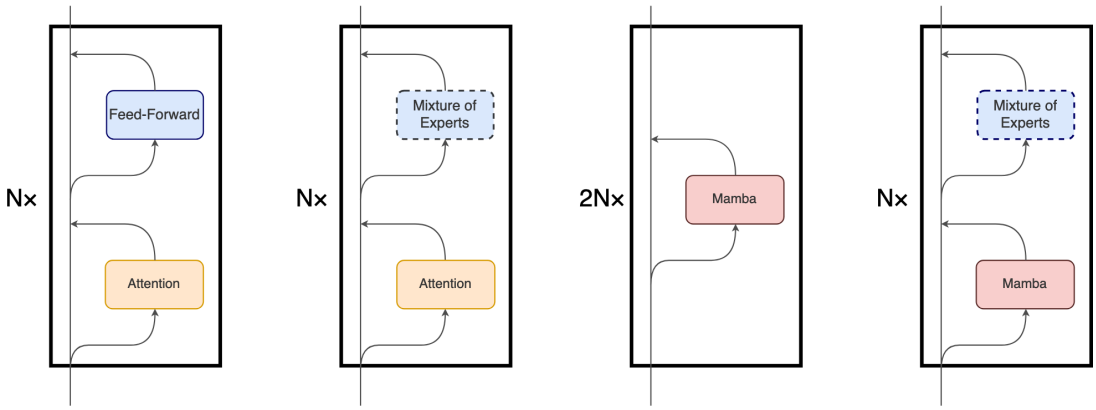


Figure 6.2: Diagrams of the architectures. From the left: vanilla Transformer, Transformer-MoE, Mamba, MoE-Mamba.

representation - and conditional processing by an MoE layer that can apply the most relevant expert (and thus the subset of parameters) for each token. The idea of interleaving conditional and unconditional processing is used in some MoE-based models, typically by alternating vanilla and MoE feed-forward layers Lepikhin et al. (2020); Fedus et al. (2022).

6.3.3 Parallel MoE-Mamba

Apart from interleaving MoE layers with Mamba layers, we explore another design, inspired by Wang (2021) and Chowdhery et al. (2023) in which MoE layer is executed in parallel with Mamba (see Figure 6.3). It achieves positive results, albeit worse than MoE-Mamba.

6.3.4 Modifying Mamba Block

In addition to attaching a separate MoE layer to Mamba, we also conducted other experiments, modifying the original block design by Gu & Dao (2023) to feature conditional MoE computation. Some of the designs show improvements over the baseline architecture and suggest promising future research directions.

6.4 Experiments

In this section we provide empirical validation of our hypothesis that interleaving Mamba with MoE can improve the performance of Mamba. Our main result, see Figure 6.1, shows that MoE-Mamba needs 2.35× fewer training steps to reach the same performance as Mamba. We also provide a detailed analysis of our design choices.

6.4.1 Training Setup

We compare MoE-Mamba to three baselines: Mamba, Transformer, and Transformer-MoE. All models in our experiments are decoder-only.

In the standard Transformer architecture, a single attention layer contains $4d^2_{\text{model}}$ parameters, whereas a feed-forward layer contains $8d^2_{\text{model}}$ parameters. A single Mamba layer contains slightly over $6d^2_{\text{model}}$ Gu & Dao (2023) parameters. To be able to compare MoE-Mamba to Transformer-based and Mamba baselines, we scale down the size of each expert in our model (we set $d_{\text{expert}} = 3d_{\text{model}}$). This way, we can keep both the number

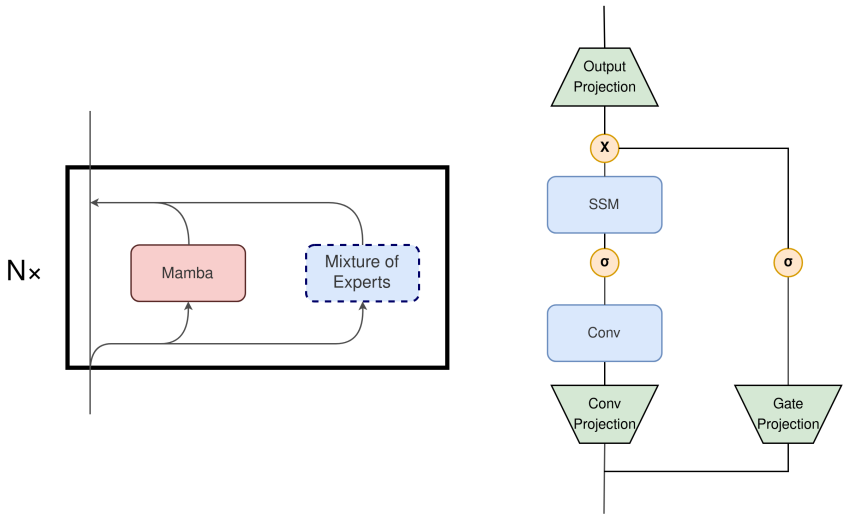


Figure 6.3: Diagram of Parallel MoE-Mamba architecture (left) and Mamba Block (right). The outputs of the Gate and Conv Projections are E (expansion factor) times bigger than the input, i.e., Conv and SSM operate on vectors $\in \mathbb{R}^{E \cdot d_{\text{model}}}$. Vanilla Mamba assumes $E = 2$ Gu & Dao (2023). Expansion factor E determines how much the input vector is scaled up by Gate and Conv Projection and then scaled down by Output Projection, and because of that, it is also proportional to the number of FLOPs and parameters in the Mamba layer.

of blocks and the number of active parameters per token roughly the same in all models of similar size. Active parameters denote those used to calculate the output for a given token (e.g., typically, only one expert in each MoE layer is active). For a discussion of the relation of active parameters and FLOPs, see Section 6.8.

Due to computational constraints, we perform most of our experiments on smaller, Transformer-MoE$_{25\text{M}}$ models and validate our findings on Transformer-MoE$_{100\text{M}}$ models.

We train the models on C4 dataset Raffel et al. (2020) on the next token prediction task using cross entropy as the loss function. We use EMA-smoothed ($\alpha = 0.001$) training log perplexity as the comparison metric for both final loss and speedup measurements as it is a more fine-grained comparison metric than test log perplexity. The test log perplexity comparison for Transformer-MoE$_{100\text{M}}$ models can be found in Section 6.13. All models use the GPT2 tokenizer Radford et al. (2019a). We tune the learning rate separately for all Transformer-MoE$_{25\text{M}}$ models and reuse it when training their Transformer-MoE$_{100\text{M}}$ counterparts. When training Transformer-MoE$_{100\text{M}}$, we divide the learning rate by two due to repeated instabilities. See Section 6.7 for further details and hyperparameters. The main experiments, described in section 6.4.2, use around 10B tokens for Transformer-MoE$_{25\text{M}}$ models and around 30B tokens for Transformer-MoE$_{100\text{M}}$ models. The experiments described in further sections use 1B tokens.

6.4.2 Main Results

Table 6.1 presents the comparison between training results of MoE-Mamba and baselines; see also Figure 6.1 for log perplexity curves. MoE-Mamba shows a remarkable improvement over the vanilla Mamba model across both model sizes. Notably, MoE-Mamba$_{100\text{M}}$ was able to perform on par with vanilla Mamba$_{100\text{M}}$ with 2.35× speedup in terms of

Model	Parameters	Active Parameters per Token	Final Log Perplexity	Speedup Over Vanilla Mamba (Training Steps)
$\text{Mamba}_{25\text{M}}$	27M	27M	3.34	1
$\text{MoE-Mamba}_{25\text{M}}$ (ours)	542M	26M	**3.19**	**1.76**
$\text{Transformer-MoE}_{25\text{M}}$	545M	25M	3.23	1.56
$\text{Transformer}_{25\text{M}}$	25M	25M	3.43	>1
$\text{Mamba}_{100\text{M}}$	121M	121M	2.99	1
$\text{MoE-Mamba}_{100\text{M}}$ (ours)	2439M	117M	**2.81**	**2.35**
$\text{Transformer-MoE}_{100\text{M}}$	2454M	114M	2.88	1.79

Table 6.1: Comparison between different architectures. The $\text{Transformer-MoE}_{25\text{M}}$ models were trained on ca. 10B tokens and the $\text{Transformer-MoE}_{100\text{M}}$ models were trained on ca. 30B tokens. Note that the parameter counts exclude embedding and output (unembedding) layers (for further discussion of reporting either non-embedding or all parameters, see Section 6.11). The numbers of total and active parameters are not matched exactly between similarly sized models due to, among other reasons, the MoE models including routers and Mamba layer not containing precisely $6d^2_{\text{model}}$ parameters - a design choice we did not want to modify. We consider those differences to be too small to be significant for our results.

processed tokens. For $\text{Transformer-MoE}_{25\text{M}}$ model size, those performance gains are lower, probably due to a lower number of training tokens. More generally, we observe that the gains increase over the training, oscillating around $1.6\times - 1.9\times$ for $\text{Transformer-MoE}_{25\text{M}}$ models after the initial training period. Further discussion of the speedup can be found in Section 6.10. We observe that MoE-Mamba performs better than the corresponding Transformer-MoE, which strengthens the findings by Gu & Dao (2023) that Mamba is a competitive alternative to the Transformer.

6.4.3 Optimal Ratio of Active Parameters in Mamba and MoE

In this section, we investigate the optimal ratio of active parameters in the Mamba layer to active parameters in the MoE layer while keeping the total number of parameters fixed. Under these constraints, a given ratio determines the so-called expansion factor E of the Mamba layer, the number of experts, and their size as detailed in Table 6.2 (see also Figure 6.3 for Mamba design).

The results are presented in Figure 6.5. We observe that increasing the number of active Mamba parameters improves the performance. However, the gains become marginal after reaching the $3:3$ ratio, and higher ratios are impractical due to inefficient hardware utilization and high routing costs caused by a large number of experts. We default to this choice in all other experiments.

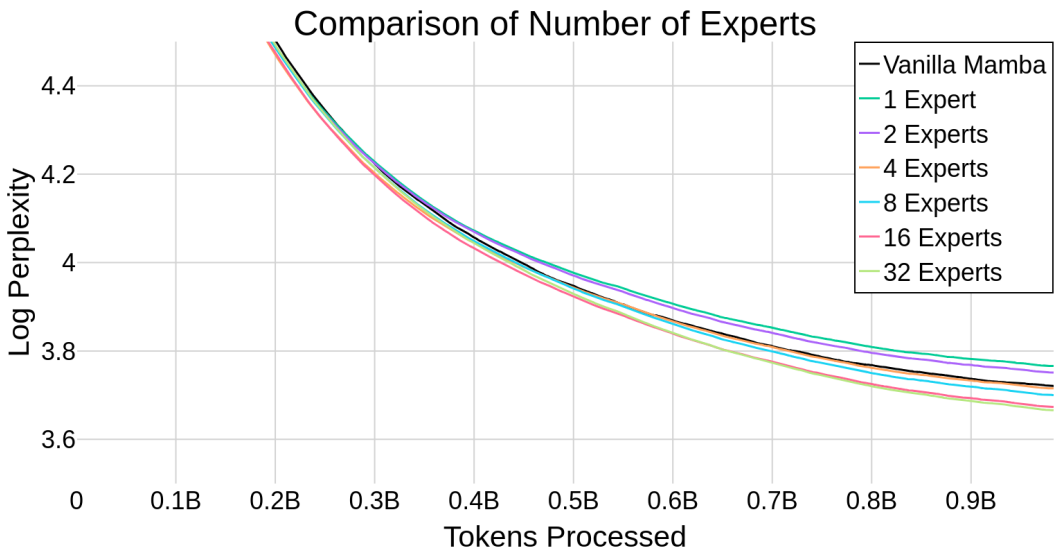


Figure 6.4: Smoothed training loss (log perplexity) for a differing number of experts for MoE-Mamba with ca. 26M active non-embedding parameters. The final log perplexity improves monotonically as the number of experts increases.

Ratio $N_{\text{Mamba}}^{\text{act. params}} : N_{\text{MoE}}^{\text{act. params}}$	Expansion Factor E (Mamba)	Expert Size	Number of Experts
$1:5$	$\frac{2}{3}$	2560	19
$2:4$	$1\frac{2}{3}$	2048	24
$3:3$	2	1536	32
$4:2$	$2\frac{2}{3}$	1024	48
$5:1$	$3\frac{1}{3}$	512	96

Table 6.2: Comparison of different ratios of parameters between Mamba and MoE. The $E = 2$ corresponds to MoE-Mamba$_{25\text{M}}$. The total number of parameters in all models is 542M and the number of active parameters per token is 26M.

6.4.4 Alternative Designs

Parallel MoE-Mamba Inspired by Wang (2021) and Chowdhery et al. (2023), we experiment with an alternative block design in which the MoE feed-forward layer and the Mamba layer are placed in parallel instead of sequentially (see Figure 6.3). We compare this design to MoE-Mamba for various numbers of experts; see Figure 6.6. MoE-Mamba outperforms this variant in all tested settings. The parallel MoE-Mamba matches vanilla Mamba when $N_{\text{experts}} \geq 8$ while requiring between 2 and 4 times as many experts and total parameters to match the performance of the sequential variant. It may be an attractive alternative at larger scales due to potentially enabling more efficient use of hardware due to different communication Wang (2021) or fused input matrix multiplications Chowdhery et al. (2023).

Inner MoE Pursuing a uniform layer design, we experimented with replacing each of the three linear projections within the Mamba block with an MoE layer; see Figure 6.3. Enumerating all the possible placements results in $2^3 - 1 = 7$ possible designs (we discard

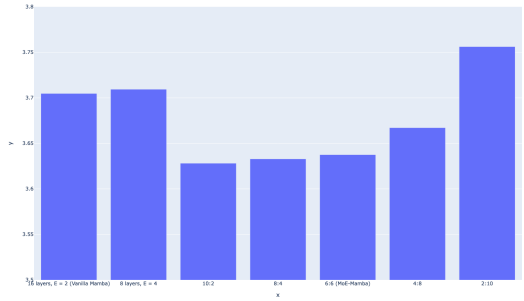


Figure 6.5: Final log perplexity at different ratios of active Mamba-to-MoE active parameters. Note that MoE contains the majority of the total parameters in each model. For further discussion of the ratios explored, see Section 6.12.

Model Name / Modified Projection	**MoE in Mamba**	
	All Layers	Every Other Layer
Vanilla Mamba	3.72	
MoE-Mamba (16 experts)	**3.67**	
Conv Projection	3.79	3.71
Gate Projection	3.89	3.70
Output Projection	4.05	3.70
Conv + Gate Projection	3.95	3.72
Conv + Output Projection	4.17	3.76
Gate + Output Projection	4.16	3.88
Conv + Gate + Output Projection	4.39	3.88

Table 6.3: Comparison of different variants of MoE in Mamba - final log perplexity (1B tokens).

one combination that would feature no MoE inside the block). We maintain a similar number of total parameters and FLOPs in all models by assuring the total number of expert feed-forward layers in a block sums up to 24 regardless of the placement, i.e., the 24 experts are split evenly between one, two or three MoE's inside the block. Inspired by Fedus et al. (2022), we also performed experiments in which only half of the Mamba blocks were modified to include MoE, but the number of experts was increased to 48 to maintain the total number of parameters.

Three of the designs (Table 6.3) achieved results marginally better than vanilla Mamba, with none outperforming MoE-Mamba. These results suggest the most promising research directions in future work.

6.4.5 Number of Experts

Figure 6.4 shows the training runs for different numbers of experts. The results show that our approach scales favorably with the number of experts. MoE-Mamba outperforms vanilla Mamba, when $N_{\text{experts}} \geq 4$. We obtain the best result with 32 experts and expect further gains with even more experts.

Interestingly, models with a small number of experts perform worse than vanilla

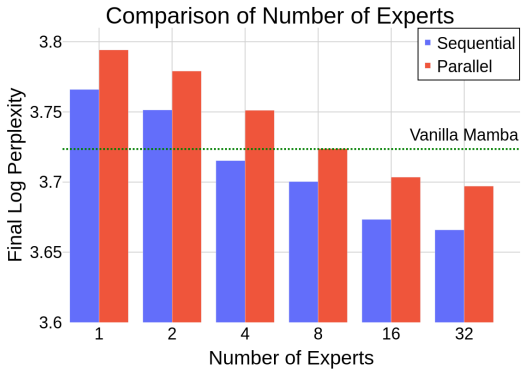


Figure 6.6: Final log perplexity comparison for varying number of experts in sequential and parallel MoE-Mamba

Number of Experts	Params	Active Parameters per Token	Log Perplexity After 1B Tokens	Speedup Over Vanilla Mamba (Training Steps)
N/A				
Vanilla Mamba	27M	27M	3.72	1
1	26M	26M	3.75	<1
4 experts	64M	26M	3.72	1.03
8 experts	114M	26M	3.70	1.10
16 experts	215M	26M	3.67	1.21
32 experts	416M	26M	**3.67**	**1.23**

Table 6.4: Log perplexity after 1B tokens for various numbers of experts. Note that the parameter counts exclude the embedding and output (unembedding) layers.

Mamba. This is consistent with Gu & Dao (2023) reporting that Mamba interleaved with feed-forward layers (which corresponds to a single-expert MoE layer) is worse than vanilla Mamba.

6.4.6 Accuracy and Perplexity

We observed that throughout the training of a variant of one of our smaller models, MoE-Mamba$_{25M}$ with 32 instead of 42 experts as presented in section 6.4.2, it maintains a lower perplexity than our strongest baseline (Transformer-MoE). However, at the same time, Transformer-MoE consistently achieves higher accuracy than MoE-Mamba. We conjecture that this might be due to the fact that attention-based models are able to copy tokens verbatim, unlike SSM-based models, whose similar abilities might be hindered by the compression of the history into a finite hidden state. We present accuracy and loss (log perplexity) plots alongside further discussion of those results in Section 6.9.

6.5 Future Work and Limitations

Scaling In this work, we perform experiments on models with the number of active parameters per token smaller than 1B, with total parameters up to 2.4B. Since MoE has

enabled Transformers to be scaled to unprecedented sizes Fedus et al. (2022), we will be excited to see the impact of scaling on the approaches proposed in our work. Developing scaling laws would be instrumental in this endeavor.

Integrating MoE Into the Mamba Layer Our experiments show that interleaving the Mamba layer with a performant sparse MoE feed-forward layer results in a promising model. However, in the dense setting, Mamba performs slightly better without the feed-forward layer. This suggests that integrating sparse computation within the Mamba layer itself could yield even better results while conserving a simple, homogeneous architecture. Our experiments, detailed in section 6.4.4, warrant some optimism, and we expect this line of research to remain relevant.

Exploration of Different Types of MoE in MoE-Mamba While we base our design on the commonly used Switch Fedus et al. (2022), numerous other MoE architectures have been proposed. Not only may those designs perform better overall, but it is possible that a different type of MoE will be optimal when combined with SSMs. Among possible changes in this regard there are Expert-Choice routers Zhou et al. (2022), fully differentiable architectures, varying number of experts and their granularity, (Puigcerver et al. (2023a); Clark et al. (2022), chapters 2 and 5), and other modifications.

Distillation Some works, e.g., Fedus et al. (2022), have shown that MoE layers can be distilled back to feed-forward layers. We expect similar results for MoE-Mamba. Interestingly, the findings by Gu & Dao (2023) indicate that a Mamba module can emulate feed-forward layers well. This raises the question of whether MoE can be distilled into a vanilla Mamba module and how that would be achieved.

Synergies We leave for future work more in-depth studies of synergies of Mamba and MoE. We suspect that there might be efficiency gains growing with the context length due to better hardware utilization; as for inference, Mamba alleviates computation and memory throughput issues stemming from larger context sizes, while MoE alleviates those same issues stemming from increasing number of parameters and knowledge stored in the model. This synergy may allow for unprecedented scaling of language models both in the number of parameters and length of the input/output.

Mamba and Attention Mechanism Mamba and Transformers make different trade-offs during data processing. This results in a different set of strengths and weaknesses, e.g., Mamba can process very long inputs but might struggle with tasks requiring detailed knowledge of the past input (e.g., some instances of copying). It would be interesting to explore combining those two architectures to achieve the best of both worlds.

Long Context Utilization Mamba and other SSMs are praised for their ability to process long context. However, the extent to which they can utilize it effectively and techniques for improving the utilization have not yet been studied in depth. To that end, some methods developed for Transformers Shi et al. (2023); Tworkowski et al. (2023); Staniszewski et al. (2024) might be applicable.

Other Modalities This work explores one direction in which Mamba can be extended. Mamba is a general architecture, and it is not limited to language modeling. We expect that it will be possible to apply MoE-Mamba to other tasks, like non-textual sequence modeling presented by Gu & Dao (2023), and different modalities, such as vision, with initial work presented by Zhu et al. (2024).

6.6 Conclusions

In this work, we presented the first integration of Mixture of Experts with Mamba architecture, MoE-Mamba. This novel method shares the inference benefits of Mamba while requiring 2.35× fewer training steps to reach the same performance. We showed possible ways of combining those techniques and positively verified performance improvements achieved with their combination. We confirmed with experiments on models up to 2.4B parameters and training lengths up to 30B tokens that those improvements over Mamba are robust to model sizes, length of training, and the number of experts.

In addition to the above, we explored and evaluated numerous alternative designs integrating Mixture of Experts within the Mamba block. While none of those variants outperformed MoE-Mamba, we think that those investigations can help prune ineffective research directions and point to promising ones.

Our work opens a new research direction of combining Mixture of Experts with State Space Models. We believe that this path will enable more efficient scaling to even larger language models.

6.7 Hyperparameters and Training Setup

Basic model hyperameters (d_{model}, d_{ff}, the number of attention heads, the number of layers) used in this work were inspired by BERT Devlin et al. (2019); Turc et al. (2019), with the Transformer-MoE$_{\text{25M}}$ models being equivalent to BERT$_{\text{Medium}}$ and Transformer-MoE$_{\text{100M}}$ models copying BERT$_{\text{Base}}$ configuration while increasing the number of blocks from 12 to 16. The learning rate schedule, as well as weight decay and gradient clipping values were set per community's standard practices. We used the AdamW optimizer Loshchilov & Hutter (2019). We tune the maximum learning rate value for each of the Transformer-MoE$_{\text{25M}}$ models separately and divide it by 2 when training Transformer-MoE$_{\text{100M}}$ counterparts. We train the models using PyTorch Paszke et al. (2019) and utilize FSDP Zhao et al. (2023b) for facilitating multi-GPU setup.

6.8 Active Parameters vs FLOPs

In this work, we report the number of active parameters (excluding embedding and unembedding layers) and not the number of floating-point operations (FLOPs), following Zhou et al. (2022). Both numbers will be roughly proportional Kaplan et al. (2020), but the number of FLOPs is both harder to calculate and less relevant for hardware-aware architecture like Mamba with its optimizations, especially during inference.

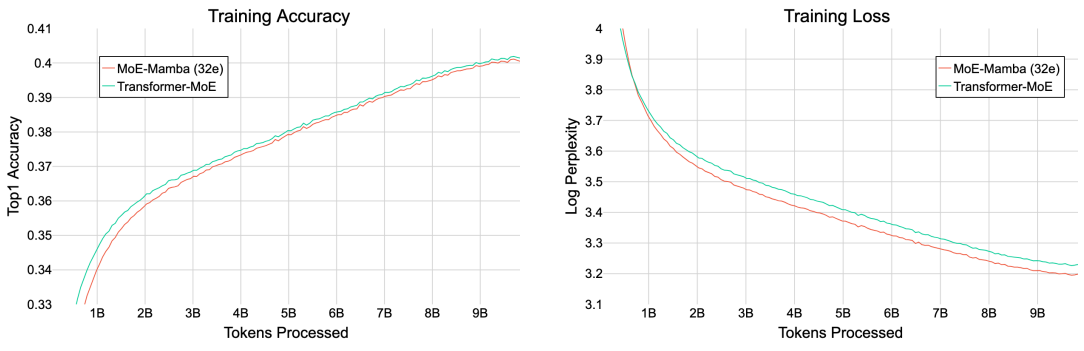
Hyperparameter		Transformer$_{25M}$	Mamba$_{25M}$	Transformer-MoE$_{25M}$	MoE-Mamba$_{25M}$
Model	Total Blocks	8	16	8	8
	d_{model}	512	512	512	512
	# Parameters	25M	27M	545M	542M
	# Active Parameters per Token	25M	27M	25M	26M
Feed-Forward	d_{ff}	2048	-	-	-
Mixture of Experts	d_{expert}	-	-	2048	1536
	N_{experts}	-	-	32	42
Position Embedding		RoPE	-	RoPE	-
Attention	N_{heads}	8	-	8	-
Training	Training Steps	150K	150K	150K	150K
	Context Length	1024	1024	1024	1024
	Batch Size	64	64	64	64
	Max Learning Rate	5e-4	1e-3	5e-4	5e-4
	LR Warmup	1%	1%	1%	1%
	LR Schedule	Cosine	Cosine	Cosine	Cosine
	Final LR Ratio	0.1	0.1	0.1	0.1
	Weight Decay	0.1	0.1	0.1	0.1
	Gradient Clipping	0.5	0.5	0.5	0.5

Table 6.5: Hyperparameters (Transformer-MoE$_{25M}$ Models). In Transformer models we use Rotary Position Embedding Su et al. (2023a).

Hyperparameters		Mamba$_{100M}$	Transformer-MoE$_{100M}$	MoE-Mamba$_{100M}$
Model	Total Blocks	32	16	16
	d_{model}	768	768	768
	# Parameters	121M	2454M	2439M
	# Active Parameters per Token	121M	114M	117M
Mixture of Experts	d_{expert}	-	3072	2304
	N_{experts}	-	32	42
Position Embedding		-	RoPE	-
Attention	N_{heads}	-	12	-
Training	Training Steps	30K	30K	30K
	Context Length	1024	1024	1024
	Batch Size	1024	1024	1024
	Max Learning Rate	1e-3	2.5e-4	5e-4
	LR Warmup	1%	1%	1%
	LR Schedule	Cosine	Cosine	Cosine
	Final LR Ratio	0.1	0.1	0.1
	Weight Decay	0.1	0.1	0.1
	Gradient Clipping	0.5	0.5	0.5

Table 6.6: Hyperparameters (Transformer-MoE$_{100M}$ Models). In Transformer-MoE$_{100M}$ we use Rotary Position Embedding Su et al. (2023a).

6.9 Accuracy and Perplexity



# of Experts	MoE-Mamba	
	Sequential	Parallel
1	3.76	3.79
2	3.74	3.77
4	3.71	3.74
8	3.69	3.72
16	3.67	3.70
32	**3.66**	3.69

Table 6.7: Comparison of sequential and parallel MoE-Mamba - final log perplexity (1B tokens).

As mentioned in section 6.4.6, we have observed a curious case of metric inconsistency between two models that achieved similar performance but were based on different architectures. We hypothesize that this discrepancy hints at a potential failure mode of Mamba and other SSMs. Due to the compression of the history into a finite hidden state, their ability for verbatim token-copying is limited. The related ability to predict the token $[B]$ given a prefix $...[A][B]...[A]$ (where $[A], [B]$ can be any tokens) has been mechanistically studied by Elhage et al. (2021) and has been conjectured to be responsible for Transformer's remarkable in-context learning capabilities Olsson et al. (2022).

Peng et al. (2023) mention that their attention-free model, RWKV, may have limited performance on tasks that require recalling precise information over long contexts due to a fixed-sized hidden state, a property that Mamba and other SSMs share. However, since the perplexity of Mamba can match the perplexity of a similarly-sized Transformer, we can suspect that Mamba compensates for that failure mode in other ways and might show a relative advantage on other tasks when compared to Transformer. In particular, it might outperform Transformers in 0-shot tasks in contrast to tasks allowing few-shot demonstrations or requiring in-context learning.

6.10 Relation between Speedup and Training Time

In our experiments, we notice that generally, as the training continues, the speedup of MoE-Mamba compared to vanilla Mamba increases (see Fig. 6.8). That is, the ratio

$$\text{speedup}(l) = \frac{\#\text{ processed tokens vanilla Mamba took to reach loss } l}{\#\text{ processed tokens MoE-Mamba took to reach loss } l}$$

increases as l decreases. Speedup in Transformer-MoE$_{25\text{M}}$ models oscillates between 1.6 and 1.9, while the speedup in Transformer-MoE$_{100\text{M}}$ models rises steadily.

6.11 Counting Model Parameters

For all models and their variants, we report the number of trainable, non-embedding parameters, i.e., we exclude the parameters in the input (embedding) and output (unembedding) layers. This convention is proposed by Kaplan et al. (2020), who note that using just non-embedding parameters gives their scaling laws a clearer form. The relatively low importance of the number of embedding parameters for the final performance has been noted by Lan et al. (2020).

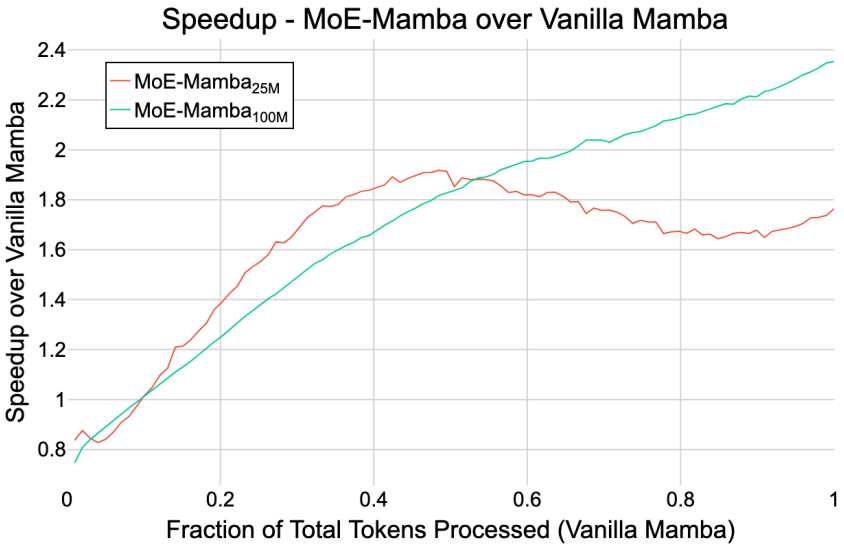


Figure 6.8: Speedup of different sizes of MoE-Mamba compared to their vanilla Mamba counterparts as training progresses.

6.12 Exploring the Optimal Mamba to MoE Active Parameters Ratio

The assignment of FLOPs and parameters to different components is an important design choice in heterogeneous architectures. For example, in Transformer, the shape of the model has been studied extensively by Kaplan et al. (2020). In our work, we investigate the optimal ratio of active parameters in the Mamba layer to the number of active parameters in the MoE layer; see Section 6.4.3. Figure 6.5 may suggest that increasing the ratio strengthens the performance and maybe assigning all active parameters to Mamba would result in the best performance (ratio *"6:0"*). It should, however, be noted, that all the investigated models contain the same number of both total parameters and active parameters per token. A hypothetical model described above could not achieve this property. If we loosen the requirement and place all the parameters in Mamba, the resulting model is the same as $\text{Mamba}_{25\text{M}}$ with the expansion factor $E = 4$ and 8 instead of 16 Mamba layers. This model achieves marginally worse final log perplexity than $\text{Mamba}_{25\text{M}}$ (3.73).

6.13 Train and Test Set Performance

In the main text, we report the loss values obtained on the train set. Our training procedure samples from the dataset, so even without processing more tokens than there are in the C4 dataset, the same documents may be encountered multiple times. However, as we process less than 20% of the tokens, the difference in performance on the train set and on the test set should be negligible. For transparency, we provide the results on the test set as well (Figure 6.9). Their variance may be high due to a limited number of sequences in each evaluation step.

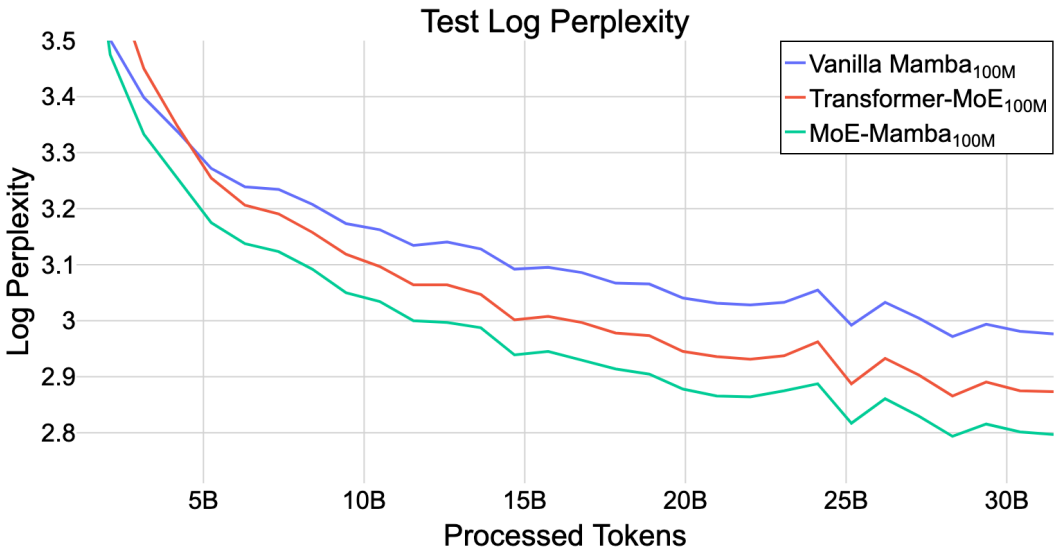


Figure 6.9: Test set loss.

Chapter 7

Conclusion

When this doctoral project began in 2020, large language models capable of holding a reasonable conversation did not yet exist. The idea that they could solve difficult problems for humans or assist with programming was closer to science fiction than to reality. With the release of ChatGPT (OpenAI, 2022), the landscape changed dramatically, and my focus shifted entirely towards improving the pretraining of large language models. In hindsight, it was exceptionally good timing to begin this research.

Our first experiments with Mixture-of-Experts (MoE) models took place when they were regarded mostly as a curiosity, and the only published scaling laws predicted that they would not scale effectively (Clark et al., 2022). Apart from introducing granularity into the MoE framework, our work was the first to demonstrate that these architectures can scale efficiently and outperform dense models. Today, the majority of recent LLMs with a disclosed architecture are known to be MoEs, and most of them explicitly adopt granular experts as standard practice. This trajectory illustrates how quickly a once-marginal idea can become central to the design of frontier models.

The broader context is no less striking. Expert forecasts now place a non-negligible probability on Artificial General Intelligence (AGI) arriving within the next decade. Surveys compiled by 80,000 Hours report (80,000 Hours, 2025a) that many researchers give a 25% chance of "high-level machine intelligence" by the early 2030s. At the same time, large language models have reached billions of users (Warren, 2024), becoming tools for education, work, and creativity. As programming itself becomes increasingly automated, the possibility emerges that models will accelerate their own development, creating a feedback loop of rapid capability gains.

This brings both opportunity and responsibility. In his essay "Machines of Loving Grace" (Amodei, 2024) Anthropic's CEO argued that advanced models could dramatically accelerate discovery in fields such as drug development, materials science, and mathematics. Yet, the risks of economic disruption, misuse, and concentration of power are equally significant.

In sum, this dissertation luckily captures a pivotal moment in the evolution of artificial intelligence. While the technical contributions focus on scaling and efficiency, the progress in AI is inseparable from its social context. The models shape not only research, but also the daily lives of billions. As we move closer to systems with general capabilities, pairing advances in efficiency with equal advances in safety and governance will be essential.

Editorial Note

This thesis was edited and corrected for grammatical and stylistic errors using LLM-based systems. These systems also helped ensure the consistency and clarity of the text. We verified the suggested changes before incorporating them to ensure the meaning remained unchanged.

Acknowledgments

This work was funded and supported by IDEAS NCBR, which also provided an excellent research environment. (Add thanks to LLM-random team, which creaed all of these papers together)

Bibliography

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

Huggingface phixtral-4x2_8 page. https://huggingface.co/mlabonne/phixtral-4x2_8. Accessed: 2014-01-25.

80,000 Hours. Shrinking agi timelines: a review of expert forecasts. https://80000hours.org/2025/03/when-do-experts-expect-agi-to-arrive/, 2025a.

80,000 Hours. Risks from power-seeking ai systems, 2025b.

Agostinelli, A., Denk, T. I., Borsos, Z., Engel, J., Verzetti, M., Caillon, A., Huang, Q., Jansen, A., Roberts, A., Tagliasacchi, M., Sharifi, M., Zeghidour, N., and Frank, C. Musiclm: Generating music from text, 2023.

Alabdulmohsin, I., Neyshabur, B., and Zhai, X. Revisiting neural scaling laws in language and vision, 2022.

Amodei, D. Machines of loving grace. https://www.darioamodei.com/essay/machines-of-loving-grace, 2024.

Anil, R., Dai, A. M., Firat, O., Johnson, M., Lepikhin, D., Passos, A., Shakeri, S., Taropa, E., Bailey, P., Chen, Z., Chu, E., Clark, J. H., Shafey, L. E., Huang, Y., Meier-Hellstern, K., Mishra, G., Moreira, E., Omernick, M., Robinson, K., Ruder, S., Tay, Y., Xiao, K., Xu, Y., Zhang, Y., Abrego, G. H., Ahn, J., Austin, J., Barham, P., Botha, J., Bradbury, J., Brahma, S., Brooks, K., Catasta, M., Cheng, Y., Cherry, C., Choquette-Choo, C. A., Chowdhery, A., Crepy, C., Dave, S., Dehghani, M., Dev, S., Devlin, J., Díaz, M., Du, N., Dyer, E., Feinberg, V., Feng, F., Fienber, V., Freitag, M., Garcia, X., Gehrmann, S., Gonzalez, L., Gur-Ari, G., Hand, S., Hashemi, H., Hou, L., Howland, J., Hu, A., Hui, J., Hurwitz, J., Isard, M., Ittycheriah, A., Jagielski, M., Jia, W., Kenealy, K., Krikun, M., Kudugunta, S., Lan, C., Lee, K., Lee, B., Li, E., Li, M., Li, W., Li, Y., Li, J., Lim, H., Lin, H., Liu, Z., Liu, F., Maggioni, M., Mahendru, A., Maynez, J., Misra, V., Moussalem, M., Nado, Z., Nham, J., Ni, E., Nystrom, A., Parrish, A., Pellat, M., Polacek, M., Polozov, A., Pope, R., Qiao, S., Reif, E., Richter, B., Riley, P., Ros, A. C., Roy, A., Saeta, B., Samuel, R., Shelby, R., Slone, A., Smilkov, D., So, D. R., Sohn, D., Tokumine, S., Valter, D., Vasudevan, V., Vodrahalli, K., Wang, X., Wang, P., Wang, Z., Wang, T., Wieting, J., Wu, Y., Xu, K., Xu, Y., Xue, L., Yin, P., Yu, J., Zhang, Q., Zheng, S., Zheng, C., Zhou, W., Zhou, D., Petrov, S., and Wu, Y. Palm 2 technical report, 2023a.

Anil, R., Dai, A. M., Firat, O., Johnson, M., Lepikhin, D., Passos, A., Shakeri, S., Taropa, E., Bailey, P., Chen, Z., Chu, E., Clark, J. H., Shafey, L. E., Huang, Y., Meier-Hellstern, K., Mishra, G., Moreira, E., Omernick, M., Robinson, K., Ruder, S., Tay, Y., Xiao, K., Xu, Y., Zhang, Y., Abrego, G. H., Ahn, J., Austin, J., Barham, P., Botha, J., Bradbury,

J., Brahma, S., Brooks, K., Catasta, M., Cheng, Y., Cherry, C., Choquette-Choo, C. A., Chowdhery, A., Crepy, C., Dave, S., Dehghani, M., Dev, S., Devlin, J., Díaz, M., Du, N., Dyer, E., Feinberg, V., Feng, F., Fienber, V., Freitag, M., Garcia, X., Gehrmann, S., Gonzalez, L., Gur-Ari, G., Hand, S., Hashemi, H., Hou, L., Howland, J., Hu, A., Hui, J., Hurwitz, J., Isard, M., Ittycheriah, A., Jagielski, M., Jia, W., Kenealy, K., Krikun, M., Kudugunta, S., Lan, C., Lee, K., Lee, B., Li, E., Li, M., Li, W., Li, Y., Li, J., Lim, H., Lin, H., Liu, Z., Liu, F., Maggioni, M., Mahendru, A., Maynez, J., Misra, V., Moussalem, M., Nado, Z., Nham, J., Ni, E., Nystrom, A., Parrish, A., Pellat, M., Polacek, M., Polozov, A., Pope, R., Qiao, S., Reif, E., Richter, B., Riley, P., Ros, A. C., Roy, A., Saeta, B., Samuel, R., Shelby, R., Slone, A., Smilkov, D., So, D. R., Sohn, D., Tokumine, S., Valter, D., Vasudevan, V., Vodrahalli, K., Wang, X., Wang, P., Wang, Z., Wang, T., Wieting, J., Wu, Y., Xu, K., Xu, Y., Xue, L., Yin, P., Yu, J., Zhang, Q., Zheng, S., Zheng, C., Zhou, W., Zhou, D., Petrov, S., and Wu, Y. Palm 2 technical report, 2023b.

Anthony, Q., Tokpanov, Y., Glorioso, P., and Millidge, B. Blackmamba: Mixture of experts for state-space models, 2024a.

Anthony, Q., Tokpanov, Y., Glorioso, P., and Millidge, B. Blackmamba: Mixture of experts for state-space models, 2024b. URL https://arxiv.org/abs/2402.01771.

Antoniak, S., Jaszczur, S., Krutul, M., Pióro, M., Krajewski, J., Ludziejewski, J., Odrzygóźdź, T., and Cygan, M. Mixture of tokens: Efficient llms through cross-example aggregation, 2023a.

Antoniak, S., Krutul, M., Pióro, M., Krajewski, J., Ludziejewski, J., Ciebiera, K., Król, K., Odrzygóźdź, T., Cygan, M., and Jaszczur, S. Mixture of tokens: Continuous moe through cross-example aggregation, 2023b. URL https://arxiv.org/abs/2310.15961.

Artetxe, M., Bhosale, S., Goyal, N., Mihaylov, T., Ott, M., Shleifer, S., Lin, X. V., Du, J., Iyer, S., Pasunuru, R., et al. Efficient large scale language modeling with mixtures of experts. arXiv preprint arXiv:2112.10684, 2021.

Artetxe, M., Bhosale, S., Goyal, N., Mihaylov, T., Ott, M., Shleifer, S., Lin, X. V., Du, J., Iyer, S., Pasunuru, R., Anantharaman, G., Li, X., Chen, S., Akin, H., Baines, M., Martin, L., Zhou, X., Koura, P. S., O'Horo, B., Wang, J., Zettlemoyer, L., Diab, M., Kozareva, Z., and Stoyanov, V. Efficient large scale language modeling with mixtures of experts, 2022.

Author, N. N. Suppressed for anonymity, 2021.

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate, 2016.

Bahri, Y., Dyer, E., Kaplan, J., Lee, J., and Sharma, U. Explaining neural scaling laws, 2021.

Bengio, E., Bacon, P.-L., Pineau, J., and Precup, D. Conditional computation in neural networks for faster models, 2016.

Bisk, Y., Zellers, R., Bras, R. L., Gao, J., and Choi, Y. Piqa: Reasoning about physical commonsense in natural language, 2019. URL https://arxiv.org/abs/1911.11641.

Borgeaud, S., Mensch, A., Hoffmann, J., Cai, T., Rutherford, E., Millican, K., Van Den Driessche, G. B., Lespiau, J.-B., Damoc, B., Clark, A., et al. Improving language models by retrieving from trillions of tokens. In International conference on machine learning, pp. 2206–2240. PMLR, 2022.

Brent, R. P. An algorithm with guaranteed convergence for finding a zero of a function. Comput. J., 14:422–425, 1971. URL `https://api.semanticscholar.org/CorpusID:10312755`.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners, 2020.

Cerisara, C. SlowLLM: large language models on consumer hardware. PhD thesis, CNRS, 2023.

Chi, Z., Dong, L., Huang, S., Dai, D., Ma, S., Patra, B., Singhal, S., Bajaj, P., Song, X., Mao, X.-L., Huang, H., and Wei, F. On the representation collapse of sparse mixture of experts, 2022.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S., and Fiedel, N. Palm: Scaling language modeling with pathways, 2022.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. Palm: Scaling language modeling with pathways. Journal of Machine Learning Research, 24(240):1–113, 2023.

Clark, A., de las Casas, D., Guy, A., Mensch, A., Paganini, M., Hoffmann, J., Damoc, B., Hechtman, B., Cai, T., Borgeaud, S., van den Driessche, G., Rutherford, E., Hennigan, T., Johnson, M., Millican, K., Cassirer, A., Jones, C., Buchatskaya, E., Budden, D., Sifre, L., Osindero, S., Vinyals, O., Rae, J., Elsen, E., Kavukcuoglu, K., and Simonyan, K. Unified scaling laws for routed language models, 2022.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL `https://arxiv.org/abs/1803.05457`.

Conneau, A. Unsupervised cross-lingual representation learning at scale. arXiv preprint arXiv:1911.02116, 2019.

Dai, D., Dong, L., Ma, S., Zheng, B., Sui, Z., Chang, B., and Wei, F. Stablemoe: Stable routing strategy for mixture of experts, 2022.

Dai, D., Deng, C., Zhao, C., Xu, R. X., Gao, H., Chen, D., Li, J., Zeng, W., Yu, X., Wu, Y., Xie, Z., Li, Y. K., Huang, P., Luo, F., Ruan, C., Sui, Z., and Liang, W. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models, 2024.

DeepSeek-AI, :, Bi, X., Chen, D., Chen, G., Chen, S., Dai, D., Deng, C., Ding, H., Dong, K., Du, Q., Fu, Z., Gao, H., Gao, K., Gao, W., Ge, R., Guan, K., Guo, D., Guo, J., Hao, G., Hao, Z., He, Y., Hu, W., Huang, P., Li, E., Li, G., Li, J., Li, Y., Li, Y. K., Liang, W., Lin, F., Liu, A. X., Liu, B., Liu, W., Liu, X., Liu, X., Liu, Y., Lu, H., Lu, S., Luo, F., Ma, S., Nie, X., Pei, T., Piao, Y., Qiu, J., Qu, H., Ren, T., Ren, Z., Ruan, C., Sha, Z., Shao, Z., Song, J., Su, X., Sun, J., Sun, Y., Tang, M., Wang, B., Wang, P., Wang, S., Wang, Y., Wang, Y., Wu, T., Wu, Y., Xie, X., Xie, Z., Xie, Z., Xiong, Y., Xu, H., Xu, R. X., Xu, Y., Yang, D., You, Y., Yu, S., Yu, X., Zhang, B., Zhang, H., Zhang, L., Zhang, L., Zhang, M., Zhang, M., Zhang, W., Zhang, Y., Zhao, C., Zhao, Y., Zhou, S., Zhou, S., Zhu, Q., and Zou, Y. Deepseek llm: Scaling open-source language models with longtermism, 2024. URL `https://arxiv.org/abs/2401.02954`.

DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., Xue, B., Wang, B., Wu, B., Feng, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., Dai, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Bao, H., Xu, H., Wang, H., Ding, H., Xin, H., Gao, H., Qu, H., Li, H., Guo, J., Li, J., Wang, J., Chen, J., Yuan, J., Qiu, J., Li, J., Cai, J. L., Ni, J., Liang, J., Chen, J., Dong, K., Hu, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Zhao, L., Wang, L., Zhang, L., Xu, L., Xia, L., Zhang, M., Zhang, M., Tang, M., Li, M., Wang, M., Li, M., Tian, N., Huang, P., Zhang, P., Wang, Q., Chen, Q., Du, Q., Ge, R., Zhang, R., Pan, R., Wang, R., Chen, R. J., Jin, R. L., Chen, R., Lu, S., Zhou, S., Chen, S., Ye, S., Wang, S., Yu, S., Zhou, S., Pan, S., Li, S. S., Zhou, S., Wu, S., Ye, S., Yun, T., Pei, T., Sun, T., Wang, T., Zeng, W., Zhao, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Xiao, W. L., An, W., Liu, X., Wang, X., Chen, X., Nie, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yang, X., Li, X., Su, X., Lin, X., Li, X. Q., Jin, X., Shen, X., Chen, X., Sun, X., Wang, X., Song, X., Zhou, X., Wang, X., Shan, X., Li, Y. K., Wang, Y. Q., Wei, Y. X., Zhang, Y., Xu, Y., Li, Y., Zhao, Y., Sun, Y., Wang, Y., Yu, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Ou, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Xiong, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Zhu, Y. X., Xu, Y., Huang, Y., Li, Y., Zheng, Y., Zhu, Y., Ma, Y., Tang, Y., Zha, Y., Yan, Y., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Xie, Z., Zhang, Z., Hao, Z., Ma, Z., Yan, Z., Wu, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Pan, Z., Huang, Z., Xu, Z., Zhang, Z., and Zhang, Z. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL `https://arxiv.org/abs/2501.12948`.

Dehghani, M., Djolonga, J., Mustafa, B., Padlewski, P., Heek, J., Gilmer, J., Steiner, A., Caron, M., Geirhos, R., Alabdulmohsin, I., Jenatton, R., Beyer, L., Tschannen, M., Arnab, A., Wang, X., Riquelme, C., Minderer, M., Puigcerver, J., Evci, U., Kumar, M., van Steenkiste, S., Elsayed, G. F., Mahendran, A., Yu, F., Oliver, A., Huot, F., Bastings, J., Collier, M. P., Gritsenko, A., Birodkar, V., Vasconcelos, C., Tay, Y., Mensink, T., Kolesnikov, A., Pavetić, F., Tran, D., Kipf, T., Lučić, M., Zhai, X., Keysers, D., Harmsen, J., and Houlsby, N. Scaling vision transformers to 22 billion parameters, 2023.

Devlin, J. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

Du, N., Huang, Y., Dai, A. M., Tong, S., Lepikhin, D., Xu, Y., Krikun, M., Zhou, Y., Yu, A. W., Firat, O., Zoph, B., Fedus, L., Bosma, M., Zhou, Z., Wang, T., Wang, Y. E., Webster, K., Pellat, M., Robinson, K., Meier-Hellstern, K., Duke, T., Dixon, L., Zhang, K., Le, Q. V., Wu, Y., Chen, Z., and Cui, C. Glam: Efficient scaling of language models with mixture-of-experts, 2022.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Roziere, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C. C., Nikolaidis, C., Allonsius, D., Song, D., Pintz, D., Livshits, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E., Lobanova, E., Dinan, E., Smith, E. M., Radenovic, F., Zhang, F., Synnaeve, G., Lee, G., Anderson, G. L., Nail, G., Mialon, G., Pang, G., Cucurell, G., Nguyen, H., Korevaar, H., Xu, H., Touvron, H., Zarov, I., Ibarra, I. A., Kloumann, I., Misra, I., Evtimov, I., Copet, J., Lee, J., Geffert, J., Vranes, J., Park, J., Mahadeokar, J., Shah, J., van der Linde, J., Billock, J., Hong, J., Lee, J., Fu, J., Chi, J., Huang, J., Liu, J., Wang, J., Yu, J., Bitton, J., Spisak, J., Park, J., Rocca, J., Johnstun, J., Saxe, J., Jia, J., Alwala, K. V., Upasani, K., Plawiak, K., Li, K., Heafield, K., Stone, K., El-Arini, K., Iyer, K., Malik, K., Chiu, K., Bhalla, K., Rantala-Yeary, L., van der Maaten, L., Chen, L., Tan, L., Jenkins, L., Martin, L., Madaan, L., Malo, L., Blecher, L., Landzaat, L., de Oliveira, L., Muzzi, M., Pasupuleti, M., Singh, M., Paluri, M., Kardas, M., Oldham, M., Rita, M., Pavlova, M., Kambadur, M., Lewis, M., Si, M., Singh, M. K., Hassan, M., Goyal, N., Torabi, N., Bashlykov, N., Bogoychev, N., Chatterji, N., Duchenne, O., Çelebi, O., Alrassy, P., Zhang, P., Li, P., Vasic, P., Weng, P., Bhargava, P., Dubal, P., Krishnan, P., Koura, P. S., Xu, P., He, Q., Dong, Q., Srinivasan, R., Ganapathy, R., Calderer, R., Cabral, R. S., Stojnic, R., Raileanu, R., Girdhar, R., Patel, R., Sauvestre, R., Polidoro, R., Sumbaly, R., Taylor, R., Silva, R., Hou, R., Wang, R., Hosseini, S., Chennabasappa, S., Singh, S., Bell, S., Kim, S. S., Edunov, S., Nie, S., Narang, S., Raparthy, S., Shen, S., Wan, S., Bhosale, S., Zhang, S., Vandenhende, S., Batra, S., Whitman, S., Sootla, S., Collot, S., Gururangan, S., Borodinsky, S., Herman, T., Fowler, T., Sheasha, T., Georgiou, T., Scialom, T., Speckbacher, T., Mihaylov, T., Xiao, T., Karn, U., Goswami, V., Gupta, V., Ramanathan, V., Kerkez, V., Gonguet, V., Do, V., Vogeti, V., Petrovic, V., Chu, W., Xiong, W., Fu, W., Meers, W., Martinet, X., Wang, X., Tan, X. E., Xie, X., Jia, X., Wang, X., Goldschlag, Y., Gaur, Y., Babaei, Y., Wen, Y., Song, Y., Zhang, Y., Li, Y., Mao, Y., Coudert, Z. D., Yan, Z., Chen, Z., Papakipos, Z., Singh, A., Grattafiori, A., Jain, A., Kelsey, A., Shajnfeld, A., Gangidi, A., Victoria, A., Goldstand, A., Menon, A., Sharma, A., Boesenberg, A., Vaughan, A., Baevski, A., Feinstein, A., Kallet, A., Sangani, A., Yunus, A., Lupu, A., Alvarado, A., Caples, A., Gu, A., Ho, A., Poulton, A., Ryan, A., Ramchandani, A., Franco, A., Saraf, A., Chowdhury, A., Gabriel, A., Bharambe, A., Eisenman, A., Yazdan, A., James, B., Maurer, B., Leonhardi, B., Huang, B., Loyd, B., Paola, B. D., Paranjape, B., Liu, B., Wu, B., Ni, B., Hancock, B., Wasti, B., Spence, B., Stojkovic, B., Gamido, B., Montalvo, B., Parker, C., Burton, C., Mejia, C., Wang, C., Kim, C., Zhou, C., Hu, C., Chu, C.-H., Cai, C., Tindal, C., Feichtenhofer, C., Civin, D., Beaty, D., Kreymer, D., Li, D., Wyatt, D., Adkins, D., Xu, D., Testuggine, D., David, D., Parikh, D., Liskovich,

D., Foss, D., Wang, D., Le, D., Holland, D., Dowling, E., Jamil, E., Montgomery, E., Presani, E., Hahn, E., Wood, E., Brinkman, E., Arcaute, E., Dunbar, E., Smothers, E., Sun, F., Kreuk, F., Tian, F., Ozgenel, F., Caggioni, F., Guzmán, F., Kanayet, F., Seide, F., Florez, G. M., Schwarz, G., Badeer, G., Swee, G., Halpern, G., Thattai, G., Herman, G., Sizov, G., Guangyi, Zhang, Lakshminarayanan, G., Shojanazeri, H., Zou, H., Wang, H., Zha, H., Habeeb, H., Rudolph, H., Suk, H., Aspegren, H., Goldman, H., Damlaj, I., Molybog, I., Tufanov, I., Veliche, I.-E., Gat, I., Weissman, J., Geboski, J., Kohli, J., Asher, J., Gaya, J.-B., Marcus, J., Tang, J., Chan, J., Zhen, J., Reizenstein, J., Teboul, J., Zhong, J., Jin, J., Yang, J., Cummings, J., Carvill, J., Shepard, J., McPhie, J., Torres, J., Ginsburg, J., Wang, J., Wu, K., U, K. H., Saxena, K., Prasad, K., Khandelwal, K., Zand, K., Matosich, K., Veeraraghavan, K., Michelena, K., Li, K., Huang, K., Chawla, K., Lakhotia, K., Huang, K., Chen, L., Garg, L., A, L., Silva, L., Bell, L., Zhang, L., Guo, L., Yu, L., Moshkovich, L., Wehrstedt, L., Khabsa, M., Avalani, M., Bhatt, M., Tsimpoukelli, M., Mankus, M., Hasson, M., Lennie, M., Reso, M., Groshev, M., Naumov, M., Lathi, M., Keneally, M., Seltzer, M. L., Valko, M., Restrepo, M., Patel, M., Vyatskov, M., Samvelyan, M., Clark, M., Macey, M., Wang, M., Hermoso, M. J., Metanat, M., Rastegari, M., Bansal, M., Santhanam, N., Parks, N., White, N., Bawa, N., Singhal, N., Egebo, N., Usunier, N., Laptev, N. P., Dong, N., Zhang, N., Cheng, N., Chernoguz, O., Hart, O., Salpekar, O., Kalinli, O., Kent, P., Parekh, P., Saab, P., Balaji, P., Rittner, P., Bontrager, P., Roux, P., Dollar, P., Zvyagina, P., Ratanchandani, P., Yuvraj, P., Liang, Q., Alao, R., Rodriguez, R., Ayub, R., Murthy, R., Nayani, R., Mitra, R., Li, R., Hogan, R., Battey, R., Wang, R., Maheswari, R., Howes, R., Rinott, R., Bondu, S. J., Datta, S., Chugh, S., Hunt, S., Dhillon, S., Sidorov, S., Pan, S., Verma, S., Yamamoto, S., Ramaswamy, S., Lindsay, S., Lindsay, S., Feng, S., Lin, S., Zha, S. C., Shankar, S., Zhang, S., Zhang, S., Wang, S., Agarwal, S., Sajuyigbe, S., Chintala, S., Max, S., Chen, S., Kehoe, S., Satterfield, S., Govindaprasad, S., Gupta, S., Cho, S., Virk, S., Subramanian, S., Choudhury, S., Goldman, S., Remez, T., Glaser, T., Best, T., Kohler, T., Robinson, T., Li, T., Zhang, T., Matthews, T., Chou, T., Shaked, T., Vontimitta, V., Ajayi, V., Montanez, V., Mohan, V., Kumar, V. S., Mangla, V., Albiero, V., Ionescu, V., Poenaru, V., Mihailescu, V. T., Ivanov, V., Li, W., Wang, W., Jiang, W., Bouaziz, W., Constable, W., Tang, X., Wang, X., Wu, X., Wang, X., Xia, X., Wu, X., Gao, X., Chen, Y., Hu, Y., Jia, Y., Qi, Y., Li, Y., Zhang, Y., Zhang, Y., Adi, Y., Nam, Y., Yu, Wang, Hao, Y., Qian, Y., He, Y., Rait, Z., DeVito, Z., Rosnbrick, Z., Wen, Z., Yang, Z., and Zhao, Z. The llama 3 herd of models, 2024a. URL `https://arxiv.org/abs/2407.21783`.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. The llama 3 herd of models. arXiv preprint arXiv:2407.21783, 2024b.

Duda, R. O., Hart, P. E., and Stork, D. G. Pattern Classification. John Wiley and Sons, 2nd edition, 2000.

Eigen, D., Ranzato, M., and Sutskever, I. Learning factored representations in a deep mixture of experts, 2014.

Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., DasSarma, N., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. A mathematical framework

for transformer circuits. Transformer Circuits Thread, 2021. https://transformer-circuits.pub/2021/framework/index.html.

Everett, K., Xiao, L., Wortsman, M., Alemi, A. A., Novak, R., Liu, P. J., Gur, I., Sohl-Dickstein, J., Kaelbling, L. P., Lee, J., and Pennington, J. Scaling exponents across parameterizations and optimizers. arXiv preprint arXiv:2407.05872, 2024.

Faiz, A., Kaneda, S., Wang, R., Osi, R., Sharma, P., Chen, F., and Jiang, L. Llmcarbon: Modeling the end-to-end carbon footprint of large language models, 2024.

Fedus, W., Zoph, B., and Shazeer, N. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, 2022.

Frantar, E., Riquelme, C., Houlsby, N., Alistarh, D., and Evci, U. Scaling laws for sparsely-connected foundation models, 2023.

Fu, D. Y., Dao, T., Saab, K. K., Thomas, A. W., Rudra, A., and Ré, C. Hungry hungry hippos: Towards language modeling with state space models, 2023.

Gale, T., Narayanan, D., Young, C., and Zaharia, M. Megablocks: Efficient sparse training with mixture-of-experts, 2022. URL https://arxiv.org/abs/2211.15841.

Gale, T., Narayanan, D., Young, C., and Zaharia, M. Megablocks: Efficient sparse training with mixture-of-experts. In Song, D., Carbin, M., and Chen, T. (eds.), Proceedings of Machine Learning and Systems, volume 5, pp. 288–304. Curan, 2023. URL https://proceedings.mlsys.org/paper_files/paper/2023/file/5a54f79333768effe7e8927bcccffe40-Paper-mlsys2023.pdf.

Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac'h, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. A framework for few-shot language model evaluation, 12 2023. URL https://zenodo.org/records/10256836.

Ghorbani, B., Firat, O., Freitag, M., Bapna, A., Krikun, M., Garcia, X., Chelba, C., and Cherry, C. Scaling laws for neural machine translation, 2021.

Gu, A. and Dao, T. Mamba: Linear-time sequence modeling with selective state spaces, 2023.

Gu, A., Johnson, I., Goel, K., Saab, K., Dao, T., Rudra, A., and Ré, C. Combining recurrent, convolutional, and continuous-time models with linear state-space layers, 2021.

Gu, A., Goel, K., Gupta, A., and Ré, C. On the parameterization and initialization of diagonal state space models. Advances in Neural Information Processing Systems, 35:35971–35983, 2022a.

Gu, A., Goel, K., and Ré, C. Efficiently modeling long sequences with structured state spaces, 2022b.

Gupta, A., Gu, A., and Berant, J. Diagonal state spaces are as effective as structured state spaces. Advances in Neural Information Processing Systems, 35:22982–22994, 2022.

Hägele, A., Bakouch, E., Kosson, A., Allal, L. B., Werra, L. V., and Jaggi, M. Scaling Laws and Compute-Optimal Training Beyond Fixed Training Durations. Advances in Neural Information Processing Systems, 2024. URL http://arxiv.org/abs/2405.18392.

Hayou, S., Ghosh, N., and Yu, B. Lora+: Efficient low rank adaptation of large models. arXiv preprint arXiv:2402.12354, 2024.

Hazimeh, H., Zhao, Z., Chowdhery, A., Sathiamoorthy, M., Chen, Y., Mazumder, R., Hong, L., and Chi, E. H. Dselect-k: Differentiable selection in the mixture of experts with applications to multi-task learning, 2021.

He, H. Making deep learning go brrrr from first principles. 2022. URL https://horace.io/brrr_intro.html.

Henighan, T., Kaplan, J., Katz, M., Chen, M., Hesse, C., Jackson, J., Jun, H., Brown, T. B., Dhariwal, P., Gray, S., Hallacy, C., Mann, B., Radford, A., Ramesh, A., Ryder, N., Ziegler, D. M., Schulman, J., Amodei, D., and McCandlish, S. Scaling laws for autoregressive generative modeling, 2020.

Hernandez, D., Kaplan, J., Henighan, T., and McCandlish, S. Scaling laws for transfer, 2021.

Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., Patwary, M. M. A., Yang, Y., and Zhou, Y. Deep learning scaling is predictable, empirically, 2017.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. Neural computation, 9 (8):1735–1780, 1997.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Rae, J. W., Vinyals, O., and Sifre, L. Training compute-optimal large language models, 2022.

Howard, J. and Ruder, S. Universal language model fine-tuning for text classification. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers), 2018.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685, 2021.

Huber, P. J. Robust Estimation of a Location Parameter. The Annals of Mathematical Statistics, 35(1):73 – 101, 1964. doi: 10.1214/aoms/1177703732. URL https://doi.org/10.1214/aoms/1177703732.

Hägele, A., Bakouch, E., Kosson, A., Allal, L. B., Werra, L. V., and Jaggi, M. Scaling laws and compute-optimal training beyond fixed training durations, 2024. URL https://arxiv.org/abs/2405.18392.

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., and Wilson, A. G. Averaging weights leads to wider optima and better generalization. arXiv preprint arXiv:1803.05407, 2018.

Jacobs, R. A., Jordan, M. I., and Barto, A. G. Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks. Cognitive Science, 15(2):219–250, 1991a. ISSN 0364-0213. doi: https://doi.org/10.1016/0364-0213(91)80006-Q. URL https://www.sciencedirect.com/science/article/pii/036402139180006Q.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. Adaptive mixtures of local experts. Neural Computation, 3(1):79–87, 1991b. doi: 10.1162/neco.1991.3.1.79.

Jaszczur, S., Chowdhery, A., Mohiuddin, A., Łukasz Kaiser, Gajewski, W., Michalewski, H., and Kanerva, J. Sparse is enough in scaling transformers, 2021.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mistral 7b, 2023.

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

Jordan, M. and Jacobs, R. Hierarchical mixtures of experts and the em algorithm. In Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan), volume 2, pp. 1339–1344 vol.2, 1993. doi: 10.1109/IJCNN.1993.716791.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models, 2020.

Kearns, M. J. Computational Complexity of Machine Learning. PhD thesis, Department of Computer Science, Harvard University, 1989.

Krajewski, J., Ludziejewski, J., Adamczewski, K., Pióro, M., Krutul, M., Antoniak, S., Ciebiera, K., Król, K., Odrzygóźdź, T., Sankowski, P., Cygan, M., and Jaszczur, S. Scaling laws for fine-grained mixture of experts, 2024a.

Krajewski, J., Ludziejewski, J., Adamczewski, K., Pióro, M., Krutul, M., Antoniak, S., Ciebiera, K., Król, K., Odrzygóźdź, T., Sankowski, P., Cygan, M., and Jaszczur, S. Scaling laws for fine-grained mixture of experts, 2024b. URL https://arxiv.org/abs/2402.07871.

Kumar, T., Ankner, Z., Spector, B. F., Bordelon, B., Muennighoff, N., Paul, M., Pehlevan, C., Re, C., and Raghunathan, A. Scaling laws for precision. In The Thirteenth International Conference on Learning Representations, 2024a.

Kumar, T., Ankner, Z., Spector, B. F., Bordelon, B., Muennighoff, N., Paul, M., Pehlevan, C., Ré, C., and Raghunathan, A. Scaling laws for precision, 2024b. URL https://arxiv.org/abs/2411.04330.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. Albert: A lite bert for self-supervised learning of language representations, 2020.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), Proceedings of the 17th International Conference on Machine Learning (ICML 2000), pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Lepikhin, D., Lee, H., Xu, Y., Chen, D., Firat, O., Huang, Y., Krikun, M., Shazeer, N., and Chen, Z. Gshard: Scaling giant models with conditional computation and automatic sharding, 2020.

Lewis, M., Bhosale, S., Dettmers, T., Goyal, N., and Zettlemoyer, L. Base layers: Simplifying training of large, sparse models, 2021.

Lewkowycz, A., Andreassen, A. J., Dohan, D., Dyer, E., Michalewski, H., Ramasesh, V. V., Slone, A., Anil, C., Schlag, I., Gutman-Solo, T., Wu, Y., Neyshabur, B., Gur-Ari, G., and Misra, V. Solving quantitative reasoning problems with language models. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), Advances in Neural Information Processing Systems, 2022. URL https://openreview.net/forum?id=IFXTZERXdM7.

Li, Y., Cai, T., Zhang, Y., Chen, D., and Dey, D. What makes convolutional models great on long sequence modeling? arXiv preprint arXiv:2210.09298, 2022.

Lingle, L. A large-scale exploration of μ-transfer. arXiv preprint arXiv:2404.05728, 2024.

Liu, Z. L., Dettmers, T., Lin, X. V., Stoyanov, V., and Li, X. Towards a unified view of sparse feed-forward network in pretraining large language model, 2023.

Loshchilov, I. and Hutter, F. Sgdr: Stochastic gradient descent with warm restarts. ICLR 2017 (5th International Conference on Learning Representations), 2016.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization, 2019.

Ludziejewski, J., Krajewski, J., Adamczewski, K., Pióro, M., Krutul, M., Antoniak, S., Ciebiera, K., Król, K., Odrzygóźdź, T., Sankowski, P., et al. Scaling laws for fine-grained mixture of experts. In Forty-first International Conference on Machine Learning.

Ludziejewski, J., Krajewski, J., Adamczewski, K., Pióro, M., Krutul, M., Antoniak, S., Ciebiera, K., Król, K., Odrzygóźdź, T., Sankowski, P., Cygan, M., and Jaszczur, S. Scaling laws for fine-grained mixture of experts. In Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F. (eds.), Proceedings of the 41st International Conference on Machine Learning, volume 235 of Proceedings of Machine Learning Research, pp. 33270–33288. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/ludziejewski24a.html.

Ma, X., Zhou, C., Kong, X., He, J., Gui, L., Neubig, G., May, J., and Zettlemoyer, L. Mega: moving average equipped gated attention. arXiv preprint arXiv:2209.10655, 2022.

Mars, M. From word embeddings to pre-trained language models: A state-of-the-art walkthrough. Applied Sciences, 2022. doi: 10.3390/app12178805.

McCandlish, S., Kaplan, J., Amodei, D., and Team, O. D. An empirical model of large-batch training, 2018. URL https://arxiv.org/abs/1812.06162.

McCulloch, W. S. and Pitts, W. A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 5:115–133, 1943.

McDonald, J., Li, B., Frey, N., Tiwari, D., Gadepally, V., and Samsi, S. Great power, great responsibility: Recommendations for reducing energy for training language models. In Findings of the Association for Computational Linguistics: NAACL 2022. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.findings-naacl.151. URL http://dx.doi.org/10.18653/v1/2022.findings-naacl.151.

Michalski, R. S., Carbonell, J. G., and Mitchell, T. M. (eds.). Machine Learning: An Artificial Intelligence Approach, Vol. I. Tioga, Palo Alto, CA, 1983.

Microsoft. Microsoft 2024 annual report. https://www.microsoft.com/investor/reports/ar24/, 2024.

Mistral. Mixtral of experts, Dec 2023. URL https://mistral.ai/news/mixtral-of-experts/.

Mitchell, T. M. The need for biases in learning generalizations. Technical report, Computer Science Department, Rutgers University, New Brunswick, MA, 1980.

Muennighoff, N., Soldaini, L., Groeneveld, D., Lo, K., Morrison, J., Min, S., Shi, W., Walsh, P., Tafjord, O., Lambert, N., Gu, Y., Arora, S., Bhagia, A., Schwenk, D., Wadden, D., Wettig, A., Hui, B., Dettmers, T., Kiela, D., Farhadi, A., Smith, N. A., Koh, P. W., Singh, A., and Hajishirzi, H. Olmoe: Open mixture-of-experts language models, 2024. URL https://arxiv.org/abs/2409.02060.

Muqeeth, M., Liu, H., and Raffel, C. Soft merging of experts with adaptive routing, 2023.

Mustafa, B., Riquelme, C., Puigcerver, J., Jenatton, R., and Houlsby, N. Multimodal contrastive learning with limoe: the language-image mixture of experts, 2022.

Nawrot, P., Tworkowski, S., Tyrolski, M., Łukasz Kaiser, Wu, Y., Szegedy, C., and Michalewski, H. Hierarchical transformers are more efficient language models, 2022.

Newell, A. and Rosenbloom, P. S. Mechanisms of skill acquisition and the law of practice. In Anderson, J. R. (ed.), Cognitive Skills and Their Acquisition, chapter 1, pp. 1–51. Lawrence Erlbaum Associates, Inc., Hillsdale, NJ, 1981.

Niizumi, D., Takeuchi, D., Ohishi, Y., Harada, N., and Kashino, K. Byol for audio: Exploring pre-trained general-purpose audio representations. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 31:137–151, 2022. doi: 10.1109/TASLP.2022.3221007.

Nostalgebraist. chinchilla's wild implications, 7 2022. URL https://www.lesswrong.com/posts/6Fpvch8RR29qLEWNH/chinchilla-s-wild-implications.

Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Johnston, S., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. In-context learning and induction heads. arXiv preprint arXiv:2209.11895, 2022.

OpenAI. Introducing chatgpt. https://openai.com/index/chatgpt/, 2022.

OpenAI, :, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J., Bello, I., Berdine, J., Bernadett-Shapiro, G., Berner, C., Bogdonoff, L., Boiko, O., Boyd, M., Brakman, A.-L., Brockman, G., Brooks, T., Brundage, M., Button, K., Cai, T., Campbell, R., Cann, A., Carey, B., Carlson, C., Carmichael, R., Chan, B., Chang, C., Chantzis, F., Chen, D., Chen, S., Chen, R., Chen, J., Chen, M., Chess, B., Cho, C., Chu, C., Chung, H. W., Cummings, D., Currier, J., Dai, Y., Decareaux, C., Degry, T., Deutsch, N., Deville, D., Dhar, A., Dohan, D., Dowling, S., Dunning, S., Ecoffet, A., Eleti, A., Eloundou, T., Farhi, D., Fedus, L., Felix, N., Fishman, S. P., Forte, J., Fulford, I., Gao, L., Georges, E., Gibson, C., Goel, V., Gogineni, T., Goh, G., Gontijo-Lopes, R., Gordon, J., Grafstein, M., Gray, S., Greene, R., Gross, J., Gu, S. S., Guo, Y., Hallacy, C., Han, J., Harris, J., He, Y., Heaton, M., Heidecke, J., Hesse, C., Hickey, A., Hickey, W., Hoeschele, P., Houghton, B., Hsu, K., Hu, S., Hu, X., Huizinga, J., Jain, S., Jain, S., Jang, J., Jiang, A., Jiang, R., Jin, H., Jin, D., Jomoto, S., Jonn, B., Jun, H., Kaftan, T., Łukasz Kaiser, Kamali, A., Kanitscheider, I., Keskar, N. S., Khan, T., Kilpatrick, L., Kim, J. W., Kim, C., Kim, Y., Kirchner, H., Kiros, J., Knight, M., Kokotajlo, D., Łukasz Kondraciuk, Kondrich, A., Konstantinidis, A., Kosic, K., Krueger, G., Kuo, V., Lampe, M., Lan, I., Lee, T., Leike, J., Leung, J., Levy, D., Li, C. M., Lim, R., Lin, M., Lin, S., Litwin, M., Lopez, T., Lowe, R., Lue, P., Makanju, A., Malfacini, K., Manning, S., Markov, T., Markovski, Y., Martin, B., Mayer, K., Mayne, A., McGrew, B., McKinney, S. M., McLeavey, C., McMillan, P., McNeil, J., Medina, D., Mehta, A., Menick, J., Metz, L., Mishchenko, A., Mishkin, P., Monaco, V., Morikawa, E., Mossing, D., Mu, T., Murati, M., Murk, O., Mély, D., Nair, A., Nakano, R., Nayak, R., Neelakantan, A., Ngo, R., Noh, H., Ouyang, L., O'Keefe, C., Pachocki, J., Paino, A., Palermo, J., Pantuliano, A., Parascandolo, G., Parish, J., Parparita, E., Passos, A., Pavlov, M., Peng, A., Perelman, A., de Avila Belbute Peres, F., Petrov, M., de Oliveira Pinto, H. P., Michael, Pokorny, Pokrass, M., Pong, V., Powell, T., Power, A., Power, B., Proehl, E., Puri, R., Radford, A., Rae, J., Ramesh, A., Raymond, C., Real, F., Rimbach, K., Ross, C., Rotsted, B., Roussez, H., Ryder, N., Saltarelli, M., Sanders, T., Santurkar, S., Sastry, G., Schmidt, H., Schnurr, D., Schulman, J., Selsam, D., Sheppard, K., Sherbakov, T., Shieh, J., Shoker, S., Shyam, P., Sidor, S., Sigler, E., Simens, M., Sitkin, J., Slama, K., Sohl, I., Sokolowsky, B., Song, Y., Staudacher, N., Such, F. P., Summers, N., Sutskever, I., Tang, J., Tezak, N., Thompson, M., Tillet, P., Tootoonchian, A., Tseng, E., Tuggle, P., Turley, N., Tworek, J., Uribe, J. F. C., Vallone, A., Vijayvergiya, A., Voss, C., Wainwright, C., Wang, J. J., Wang, A., Wang, B., Ward, J., Wei, J., Weinmann, C., Welihinda, A., Welinder, P., Weng, J., Weng, L., Wiethoff, M., Willner, D., Winter, C., Wolrich, S., Wong, H., Workman, L., Wu, S., Wu, J., Wu, M., Xiao, K., Xu, T., Yoo, S., Yu, K., Yuan, Q., Zaremba, W., Zellers, R., Zhang, C., Zhang, M., Zhao, S., Zheng, T., Zhuang, J., Zhuk, W., and Zoph, B. Gpt-4 technical report, 2023.

Orvieto, A., Smith, S. L., Gu, A., Fernando, A., Gulcehre, C., Pascanu, R., and De, S. Resurrecting recurrent neural networks for long sequences. arXiv preprint arXiv:2303.06349, 2023.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library, 2019.

Pearce, T. and Song, J. Reconciling kaplan and chinchilla scaling laws, 2024a. URL https://arxiv.org/abs/2406.12907.

Pearce, T. and Song, J. Reconciling kaplan and chinchilla scaling laws. Transactions on Machine Learning Research, 2024b.

Penedo, G., Kydlíček, H., Lozhkov, A., Mitchell, M., Raffel, C. A., Von Werra, L., Wolf, T., et al. The fineweb datasets: Decanting the web for the finest text data at scale. Advances in Neural Information Processing Systems, 37:30811–30849, 2024a.

Penedo, G., Kydlíček, H., allal, L. B., Lozhkov, A., Mitchell, M., Raffel, C., Werra, L. V., and Wolf, T. The fineweb datasets: Decanting the web for the finest text data at scale, 2024b. URL https://arxiv.org/abs/2406.17557.

Peng, B., Alcaide, E., Anthony, Q., Albalak, A., Arcadinho, S., Biderman, S., Cao, H., Cheng, X., Chung, M., Grella, M., GV, K. K., He, X., Hou, H., Lin, J., Kazienko, P., Kocon, J., Kong, J., Koptyra, B., Lau, H., Mantri, K. S. I., Mom, F., Saito, A., Song, G., Tang, X., Wang, B., Wind, J. S., Wozniak, S., Zhang, R., Zhang, Z., Zhao, Q., Zhou, P., Zhou, Q., Zhu, J., and Zhu, R.-J. Rwkv: Reinventing rnns for the transformer era, 2023.

Pióro, M., Ciebiera, K., Król, K., Ludziejewski, J., and Jaszczur, S. Moe-mamba: Efficient selective state space models with mixture of experts, 2024.

Poli, M., Massaroli, S., Nguyen, E., Fu, D. Y., Dao, T., Baccus, S., Bengio, Y., Ermon, S., and Ré, C. Hyena hierarchy: Towards larger convolutional language models, 2023.

Porian, T., Wortsman, M., Jitsev, J., Schmidt, L., and Carmon, Y. Resolving discrepancies in compute-optimal scaling of language models. Advances in Neural Information Processing Systems, 37:100535–100570, 2024.

Porian, T., Wortsman, M., Jitsev, J., Schmidt, L., and Carmon, Y. Resolving discrepancies in compute-optimal scaling of language models, 2025. URL https://arxiv.org/abs/2406.19146.

Press, O. and Wolf, L. Using the output embedding to improve language models. arXiv preprint arXiv:1608.05859, 2017.

Puigcerver, J., Riquelme, C., Mustafa, B., and Houlsby, N. From sparse to soft mixtures of experts, 2023a.

Puigcerver, J., Riquelme, C., Mustafa, B., and Houlsby, N. From sparse to soft mixtures of experts, 2023b.

Puigcerver, J., Riquelme, C., Mustafa, B., and Houlsby, N. From sparse to soft mixtures of experts, 2024. URL https://arxiv.org/abs/2308.00951.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. Improving language understanding by generative pre-training. 2018a.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. 2018b. URL https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. 2019a.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. OpenAI blog, 1(8):9, 2019b.

Rae, J. W., Borgeaud, S., Cai, T., Millican, K., Hoffmann, J., Song, F., Aslanides, J., Henderson, S., Ring, R., Young, S., Rutherford, E., Hennigan, T., Menick, J., Cassirer, A., Powell, R., van den Driessche, G., Hendricks, L. A., Rauh, M., Huang, P.-S., Glaese, A., Welbl, J., Dathathri, S., Huang, S., Uesato, J., Mellor, J., Higgins, I., Creswell, A., McAleese, N., Wu, A., Elsen, E., Jayakumar, S., Buchatskaya, E., Budden, D., Sutherland, E., Simonyan, K., Paganini, M., Sifre, L., Martens, L., Li, X. L., Kuncoro, A., Nematzadeh, A., Gribovskaya, E., Donato, D., Lazaridou, A., Mensch, A., Lespiau, J.-B., Tsimpoukelli, M., Grigorev, N., Fritz, D., Sottiaux, T., Pajarskas, M., Pohlen, T., Gong, Z., Toyama, D., de Masson d'Autume, C., Li, Y., Terzi, T., Mikulik, V., Babuschkin, I., Clark, A., de Las Casas, D., Guy, A., Jones, C., Bradbury, J., Johnson, M., Hechtman, B., Weidinger, L., Gabriel, I., Isaac, W., Lockhart, E., Osindero, S., Rimell, L., Dyer, C., Vinyals, O., Ayoub, K., Stanway, J., Bennett, L., Hassabis, D., Kavukcuoglu, K., and Irving, G. Scaling language models: Methods, analysis & insights from training gopher, 2022.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. The Journal of Machine Learning Research, 21(1):5485–5551, 2020.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023.

Rajbhandari, S., Li, C., Yao, Z., Zhang, M., Aminabadi, R. Y., Awan, A. A., Rasley, J., and He, Y. DeepSpeed-MoE: Advancing mixture-of-experts inference and training to power next-generation AI scale. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), Proceedings of the 39th International Conference on Machine Learning, volume 162 of Proceedings of Machine Learning Research, pp. 18332–18346. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/rajbhandari22a.html.

Rasch, M. J., Mackin, C., Gallo, M. L., Chen, A., Fasoli, A., Odermatt, F., Li, N., Nandakumar, S. R., Narayanan, P., Tsai, H., Burr, G. W., Sebastian, A., and Narayanan, V. Hardware-aware training for large-scale and diverse deep learning inference workloads using in-memory computing-based accelerators, 2023.

Riquelme, C., Puigcerver, J., Mustafa, B., Neumann, M., Jenatton, R., Pinto, A. S., Keysers, D., and Houlsby, N. Scaling vision with sparse mixture of experts, 2021a. URL https://arxiv.org/abs/2106.05974.

Riquelme, C., Puigcerver, J., Mustafa, B., Neumann, M., Jenatton, R., Susano Pinto, A., Keysers, D., and Houlsby, N. Scaling vision with sparse mixture of experts. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), Advances in Neural Information Processing Systems, volume 34, pp. 8583–8595. Curran Associates, Inc., 2021b. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/48237d9f2dea8c74c2a72126cf63d933-Paper.pdf.

Roller, S., Sukhbaatar, S., Szlam, A., and Weston, J. Hash layers for large sparse models, 2021.

Samsi, S., Zhao, D., McDonald, J., Li, B., Michaleas, A., Jones, M., Bergeron, W., Kepner, J., Tiwari, D., and Gadepally, V. From words to watts: Benchmarking the energy costs of large language model inference, 2023.

Samuel, A. L. Some studies in machine learning using the game of checkers. IBM Journal of Research and Development, 3(3):211–229, 1959.

Sanseviero, O., Tunstall, L., Schmid, P., Mangrulkar, S., Belkada, Y., and Cuenca, P. Mixture of experts explained, 2023a. URL https://huggingface.co/blog/moe.

Sanseviero, O., Tunstall, L., Schmid, P., Mangrulkar, S., Belkada, Y., and Cuenca, P. Mixture of experts explained, 2023b. URL https://huggingface.co/blog/moe.

Sardana, N., Portes, J., Doubov, S., and Frankle, J. Beyond chinchilla-optimal: Accounting for inference in language model scaling laws, 2024a. URL https://arxiv.org/abs/2401.00448.

Sardana, N., Portes, J., Doubov, S., and Frankle, J. Beyond chinchilla-optimal: accounting for inference in language model scaling laws. In Proceedings of the 41st International Conference on Machine Learning, pp. 43445–43460, 2024b.

Sevilla, J., Heim, L., Ho, A., Besiroglu, T., Hobbhahn, M., and Villalobos, P. Compute trends across three eras of machine learning. In 2022 International Joint Conference on Neural Networks (IJCNN). IEEE, July 2022. doi: 10.1109/ijcnn55064.2022.9891914. URL http://dx.doi.org/10.1109/IJCNN55064.2022.9891914.

Shazeer, N. Glu variants improve transformer, 2020a.

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

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017.

Shazeer, N., Cheng, Y., Parmar, N., Tran, D., Vaswani, A., Koanantakool, P., Hawkins, P., Lee, H., Hong, M., Young, C., Sepassi, R., and Hechtman, B. Mesh-tensorflow: Deep learning for supercomputers, 2018.

Shi, W., Min, S., Lomeli, M., Zhou, C., Li, M., James, R., Lin, X. V., Smith, N. A., Zettlemoyer, L., Yih, S., and Lewis, M. In-context pretraining: Language modeling beyond document boundaries, 2023.

Smith, J. T. H., Warrington, A., and Linderman, S. W. Simplified state space layers for sequence modeling, 2023.

Staff, T. Demis hassabis is preparing for ai's endgame. TIME, April 2025. URL https://time.com/7277608/demis-hassabis-interview-time100-2025/.

Staniszewski, K., Tworkowski, S., Jaszczur, S., Michalewski, H., Łukasz Kuciński, and Miłoś, P. Structured packing in llm training improves long context utilization, 2024.

Strubell, E., Ganesh, A., and McCallum, A. Energy and policy considerations for deep learning in nlp, 2019.

Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding, 2023a.

Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding, 2023b. URL https://arxiv.org/abs/2104.09864.

Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding. Neurocomputing, 568:127063, 2024.

Sun, C., Qiu, X., Xu, Y., and Huang, X. How to fine-tune bert for text classification? In Chinese computational linguistics: 18th China national conference, CCL 2019, Kunming, China, October 18–20, 2019, proceedings 18, pp. 194–206. Springer, 2019.

Sun, Y., Dong, L., Huang, S., Ma, S., Xia, Y., Xue, J., Wang, J., and Wei, F. Retentive network: A successor to transformer for large language models, 2023.

Sutton, R. The bitter lesson. Incomplete Ideas (blog), 13(1), 2019a.

Sutton, R. The bitter lesson, 2019b. URL http://www.incompleteideas.net/IncIdeas/BitterLesson.html. Accessed: 2024-08-02.

Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., and Metzler, D. Long range arena: A benchmark for efficient transformers. arXiv preprint arXiv:2011.04006, 2020.

Team, G. Gemini: A family of highly capable multimodal models, 2023.

Team, Q. Qwen2.5 technical report. arXiv preprint arXiv:2412.15115, 2024a.

Team, Q. Qwen1.5-moe: Matching 7b model performance with 1/3 activated parameters", February 2024b. URL https://qwenlm.github.io/blog/qwen-moe/.

TogetherComputer. Redpajama: An open source recipe to reproduce llama training dataset, April 2023. URL https://github.com/togethercomputer/RedPajama-Data.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. Llama: Open and efficient foundation language models, 2023a.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models, 2023b.

Turc, I., Chang, M.-W., Lee, K., and Toutanova, K. Well-read students learn better: On the importance of pre-training compact models, 2019.

Tworkowski, S., Staniszewski, K., Pacek, M., Wu, Y., Michalewski, H., and Miłoś, P. Focused transformer: Contrastive training for context scaling, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. CoRR, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need, 2023.

Villalobos, P., Ho, A., Sevilla, J., Besiroglu, T., Heim, L., and Hobbhahn, M. Will we run out of data? limits of llm scaling based on human-generated data, 2024. URL https://arxiv.org/abs/2211.04325.

Wang, B. Mesh-Transformer-JAX: Model-Parallel Implementation of Transformer Language Model with JAX. https://github.com/kingoflolz/mesh-transformer-jax, May 2021.

Warren, T. Chatgpt now has over 300 million weekly users. https://www.theverge.com/2024/12/4/24313097/chatgpt-300-million-weekly-users, December 2024.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E. H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., and Fedus, W. Emergent abilities of large language models, 2022.

Workshop, B., :, Scao, T. L., Fan, A., Akiki, C., Pavlick, E., Ilić, S., Hesslow, D., Castagné, R., Luccioni, A. S., Yvon, F., Gallé, M., Tow, J., Rush, A. M., Biderman, S., Webson, A., Ammanamanchi, P. S., Wang, T., Sagot, B., Muennighoff, N., del Moral, A. V., Ruwase, O., Bawden, R., Bekman, S., McMillan-Major, A., Beltagy, I., Nguyen, H., Saulnier, L., Tan, S., Suarez, P. O., Sanh, V., Laurençon, H., Jernite, Y., Launay, J., Mitchell, M., Raffel, C., Gokaslan, A., Simhi, A., Soroa, A., Aji, A. F., Alfassy, A., Rogers, A., Nitzav, A. K., Xu, C., Mou, C., Emezue, C., Klamm, C., Leong, C., van Strien, D., Adelani, D. I., Radev, D., Ponferrada, E. G., Levkovizh, E., Kim, E., Natan, E. B., Toni, F. D., Dupont, G., Kruszewski, G., Pistilli, G., Elsahar, H., Benyamina, H., Tran, H., Yu, I., Abdulmumin, I., Johnson, I., Gonzalez-Dios, I., de la Rosa, J., Chim, J., Dodge, J., Zhu, J., Chang, J., Frohberg, J., Tobing, J., Bhattacharjee, J., Almubarak, K., Chen, K., Lo, K., Werra, L. V., Weber, L., Phan, L., allal, L. B., Tanguy, L., Dey, M., Muñoz, M. R., Masoud, M., Grandury, M., Šaško, M., Huang, M., Coavoux, M., Singh, M., Jiang, M. T.-J., Vu, M. C., Jauhar, M. A., Ghaleb, M., Subramani, N., Kassner, N., Khamis, N., Nguyen, O., Espejel, O., de Gibert, O., Villegas, P., Henderson, P., Colombo, P., Amuok, P., Lhoest, Q., Harliman, R., Bommasani, R., López, R. L., Ribeiro, R., Osei, S., Pyysalo, S., Nagel, S., Bose, S., Muhammad, S. H., Sharma, S., Longpre, S., Nikpoor, S., Silberberg, S., Pai, S., Zink, S., Torrent, T. T., Schick, T., Thrush, T., Danchev, V., Nikoulina, V., Laippala, V., Lepercq, V., Prabhu, V., Alyafeai, Z., Talat, Z., Raja, A., Heinzerling, B., Si, C., Taşar, D. E., Salesky, E., Mielke, S. J., Lee, W. Y., Sharma, A., Santilli, A., Chaffin, A., Stiegler, A., Datta, D., Szczechla, E., Chhablani, G., Wang, H., Pandey, H., Strobelt, H., Fries, J. A., Rozen, J., Gao, L., Sutawika, L., Bari, M. S., Al-shaibani, M. S., Manica, M., Nayak, N., Teehan, R., Albanie, S., Shen, S., Ben-David, S., Bach, S. H., Kim, T., Bers, T., Fevry, T., Neeraj, T., Thakker, U., Raunak, V., Tang, X., Yong, Z.-X., Sun, Z., Brody, S., Uri, Y., Tojarieh, H., Roberts, A., Chung, H. W., Tae, J., Phang, J., Press, O., Li, C.,

Narayanan, D., Bourfoune, H., Casper, J., Rasley, J., Ryabinin, M., Mishra, M., Zhang, M., Shoeybi, M., Peyrounette, M., Patry, N., Tazi, N., Sanseviero, O., von Platen, P., Cornette, P., Lavallée, P. F., Lacroix, R., Rajbhandari, S., Gandhi, S., Smith, S., Requena, S., Patil, S., Dettmers, T., Baruwa, A., Singh, A., Cheveleva, A., Ligozat, A.-L., Subramonian, A., Névéol, A., Lovering, C., Garrette, D., Tunuguntla, D., Reiter, E., Taktasheva, E., Voloshina, E., Bogdanov, E., Winata, G. I., Schoelkopf, H., Kalo, J.-C., Novikova, J., Forde, J. Z., Clive, J., Kasai, J., Kawamura, K., Hazan, L., Carpuat, M., Clinciu, M., Kim, N., Cheng, N., Serikov, O., Antverg, O., van der Wal, O., Zhang, R., Zhang, R., Gehrmann, S., Mirkin, S., Pais, S., Shavrina, T., Scialom, T., Yun, T., Limisiewicz, T., Rieser, V., Protasov, V., Mikhailov, V., Pruksachatkun, Y., Belinkov, Y., Bamberger, Z., Kasner, Z., Rueda, A., Pestana, A., Feizpour, A., Khan, A., Faranak, A., Santos, A., Hevia, A., Unldreaj, A., Aghagol, A., Abdollahi, A., Tammour, A., HajiHosseini, A., Behroozi, B., Ajibade, B., Saxena, B., Ferrandis, C. M., McDuff, D., Contractor, D., Lansky, D., David, D., Kiela, D., Nguyen, D. A., Tan, E., Baylor, E., Ozoani, E., Mirza, F., Ononiwu, F., Rezanejad, H., Jones, H., Bhattacharya, I., Solaiman, I., Sedenko, I., Nejadgholi, I., Passmore, J., Seltzer, J., Sanz, J. B., Dutra, L., Samagaio, M., Elbadri, M., Mieskes, M., Gerchick, M., Akinlolu, M., McKenna, M., Qiu, M., Ghauri, M., Burynok, M., Abrar, N., Rajani, N., Elkott, N., Fahmy, N., Samuel, O., An, R., Kromann, R., Hao, R., Alizadeh, S., Shubber, S., Wang, S., Roy, S., Viguier, S., Le, T., Oyebade, T., Le, T., Yang, Y., Nguyen, Z., Kashyap, A. R., Palasciano, A., Callahan, A., Shukla, A., Miranda-Escalada, A., Singh, A., Beilharz, B., Wang, B., Brito, C., Zhou, C., Jain, C., Xu, C., Fourrier, C., Periñán, D. L., Molano, D., Yu, D., Manjavacas, E., Barth, F., Fuhrimann, F., Altay, G., Bayrak, G., Burns, G., Vrabec, H. U., Bello, I., Dash, I., Kang, J., Giorgi, J., Golde, J., Posada, J. D., Sivaraman, K. R., Bulchandani, L., Liu, L., Shinzato, L., de Bykhovetz, M. H., Takeuchi, M., Pàmies, M., Castillo, M. A., Nezhurina, M., Sänger, M., Samwald, M., Cullan, M., Weinberg, M., Wolf, M. D., Mihaljcic, M., Liu, M., Freidank, M., Kang, M., Seelam, N., Dahlberg, N., Broad, N. M., Muellner, N., Fung, P., Haller, P., Chandrasekhar, R., Eisenberg, R., Martin, R., Canalli, R., Su, R., Su, R., Cahyawijaya, S., Garda, S., Deshmukh, S. S., Mishra, S., Kiblawi, S., Ott, S., Sang-aroonsiri, S., Kumar, S., Schweter, S., Bharati, S., Laud, T., Gigant, T., Kainuma, T., Kusa, W., Labrak, Y., Bajaj, Y. S., Venkatraman, Y., Xu, Y., Xu, Y., Xu, Y., Tan, Z., Xie, Z., Ye, Z., Bras, M., Belkada, Y., and Wolf, T. Bloom: A 176b-parameter open-access multilingual language model, 2023.

Wortsman, M., Liu, P. J., Xiao, L., Everett, K., Alemi, A., Adlam, B., Co-Reyes, J. D., Gur, I., Kumar, A., Novak, R., Pennington, J., Sohl-dickstein, J., Xu, K., Lee, J., Gilmer, J., and Kornblith, S. Small-scale proxies for large-scale transformer training instabilities. arXiv preprint arXiv:2309.14322, 2023.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A. C., Salakhutdinov, R., Zemel, R. S., and Bengio, Y. Show, attend and tell: Neural image caption generation with visual attention. CoRR, abs/1502.03044, 2015. URL `http://arxiv.org/abs/1502.03044`.

Xue, F., Zheng, Z., Fu, Y., Ni, J., Zheng, Z., Zhou, W., and You, Y. Openmoe: Open mixture-of-experts language models. `https://github.com/XueFuzhao/OpenMoE`, 2023.

Xue, F., Zheng, Z., Fu, Y., Ni, J., Zheng, Z., Zhou, W., and You, Y. Openmoe: An early effort on open mixture-of-experts language models. arXiv preprint arXiv:2402.01739, 2024.

Yang, G. Tensor programs ii: Neural tangent kernel for any architecture. arXiv preprint arXiv:2006.14548, 2020.

Yang, G., Hu, E. J., Babuschkin, I., Sidor, S., Liu, X., Farhi, D., Ryder, N., Pachocki, J., Chen, W., and Gao, J. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer. arXiv preprint arXiv:2203.03466, 2022.

Yin, S., Fu, C., Zhao, S., Li, K., Sun, X., Xu, T., and Chen, E. A survey on multimodal large language models, 2023.

Yun, L., Zhuang, Y., Fu, Y., Xing, E. P., and Zhang, H. Toward inference-optimal mixture-of-expert large language models, 2024. URL https://arxiv.org/abs/2404.02852.

Zadouri, T., Üstün, A., Ahmadian, A., Ermiş, B., Locatelli, A., and Hooker, S. Pushing mixture of experts to the limit: Extremely parameter efficient moe for instruction tuning. arXiv preprint arXiv:2309.05444, 2023.

Zadouri, T., Üstün, A., Ahmadian, A., Ermis, B., Locatelli, A., and Hooker, S. Pushing mixture of experts to the limit: Extremely parameter efficient moe for instruction tuning. In The Twelfth International Conference on Learning Representations, 2024.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence?, 2019. URL https://arxiv.org/abs/1905.07830.

Zhai, S., Talbott, W., Srivastava, N., Huang, C., Goh, H., Zhang, R., and Susskind, J. An attention free transformer, 2021.

Zhai, X., Kolesnikov, A., Houlsby, N., and Beyer, L. Scaling vision transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12104–12113, 2022.

Zhang, Q., Chen, M., Bukharin, A., Karampatziakis, N., He, P., Cheng, Y., Chen, W., and Zhao, T. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. In International Conference on Learning Representations (ICLR), 2023.

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., Liu, P., Nie, J.-Y., and Wen, J.-R. A survey of large language models, 2023a.

Zhao, Y., Gu, A., Varma, R., Luo, L., Huang, C.-C., Xu, M., Wright, L., Shojanazeri, H., Ott, M., Shleifer, S., Desmaison, A., Balioglu, C., Damania, P., Nguyen, B., Chauhan, G., Hao, Y., Mathews, A., and Li, S. Pytorch fsdp: Experiences on scaling fully sharded data parallel, 2023b.

Zhou, Y., Lei, T., Liu, H., Du, N., Huang, Y., Zhao, V., Dai, A., Chen, Z., Le, Q., and Laudon, J. Mixture-of-experts with expert choice routing, 2022.

Zhou, Y., Du, N., Huang, Y., Peng, D., Lan, C., Huang, D., Shakeri, S., So, D., Dai, A., Lu, Y., Chen, Z., Le, Q., Cui, C., Laundon, J., and Dean, J. Brainformers: Trading simplicity for efficiency, 2023.

Zhu, L., Liao, B., Zhang, Q., Wang, X., Liu, W., and Wang, X. Vision mamba: Efficient visual representation learning with bidirectional state space model, 2024.

Zohourianshahzadi, Z. and Kalita, J. K. Neural attention for image captioning: review of outstanding methods. Artificial Intelligence Review, 55(5):3833–3862, November 2021. ISSN 1573-7462. doi: 10.1007/s10462-021-10092-2. URL http://dx.doi.org/10.1007/s10462-021-10092-2.

Zoph, B., Bello, I., Kumar, S., Du, N., Huang, Y., Dean, J., Shazeer, N., and Fedus, W. St-moe: Designing stable and transferable sparse expert models. arXiv preprint arXiv:2202.08906, 2022a.

Zoph, B., Bello, I., Kumar, S., Du, N., Huang, Y., Dean, J., Shazeer, N., and Fedus, W. St-moe: Designing stable and transferable sparse expert models, 2022b.