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

Jan Ludziejewski [*1 2] **Maciej Pióro** [*2 3] **Jakub Krajewski** [*1 2]
Maciej Stefaniak [1] **Michał Krutul** [1 2] **Jan Małaśnicki** [1 2]
Marek Cygan [1 4] **Piotr Sankowski** [5 1 6] **Kamil Adamczewski** [2 7]
Piotr Miłoś [2 8] **Sebastian Jaszczur** [1 2]

Abstract

Mixture of Experts (MoE) architectures have significantly increased computational efficiency in both research and real-world applications of large-scale machine learning models. However, their scalability and efficiency under memory constraints remain relatively underexplored. In this work, we present joint scaling laws for dense and MoE models, incorporating key factors such as the number of active parameters, dataset size, and the number of experts. Our findings provide a principled framework for selecting the optimal MoE configuration under fixed memory and compute budgets. Surprisingly, we show that MoE models can be more memory-efficient than dense models, contradicting conventional wisdom. Extensive empirical validation confirms the theoretical predictions of our scaling laws. These results offer actionable insights for designing and deploying MoE models in practical large-scale training scenarios.

1. Introduction

Recently, language models have grown increasingly large, a trend accelerated by Mixture of Experts (MoE) techniques (Fedus et al., 2022; Du et al., 2022). MoE models are now widely adopted (Jiang et al., 2024; Dai et al., 2024) and are generally considered compute-efficient (Ludziejewski et al., 2024; Clark et al., 2022), though often considered to be memory-inefficient (Zadouri et al., 2024). However, the precise trade-offs between compute and memory efficiency remain unclear.

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.[†]

In summary, the key contributions of this work are:

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

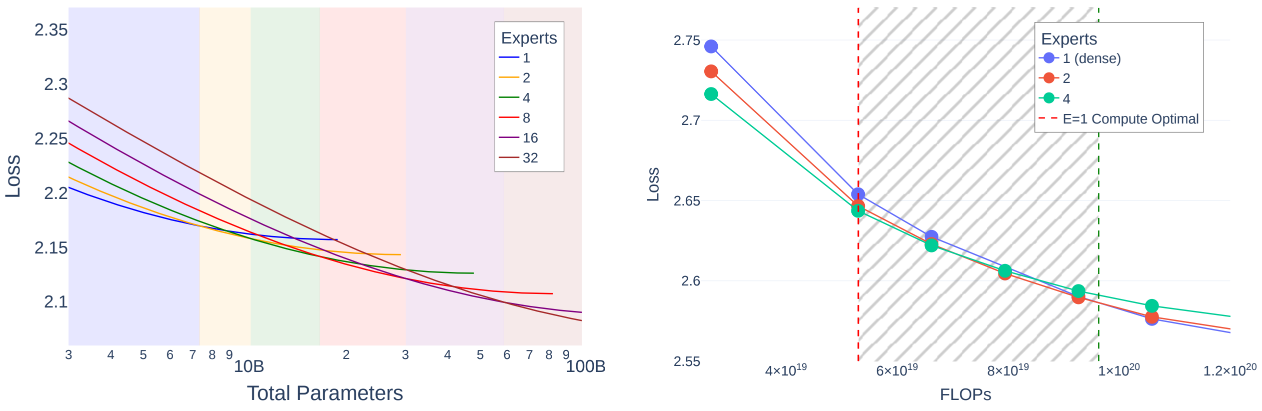
$$\mathcal{L}(N_{\text{act}}, D, \hat{E}) = a\hat{E}^{\delta} N_{\text{act}}^{\alpha+\gamma\ln(\hat{E})} + b\hat{E}^{\omega} D^{\beta+\zeta\ln(\hat{E})} + c, \qquad (1)$$

  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.

[*] Core contributors [1] University of Warsaw [2] IDEAS NCBR [3] Institute of Fundamental Technological Research, Polish Academy of Sciences [4] Nomagic [5] Research Institute IDEAS [6] MIM Solutions [7] Wroclaw University of Science and Technology [8] Institute of Mathematics, Polish Academy of Sciences. Correspondence to: Jan Ludziejewski <ludziej@mimuw.edu.pl>, Maciej Pióro <maciej.pioro@gmail.com>, Jakub Krajewski <gim.jakubk@gmail.com>, Sebastian Jaszczur <sebastian.jaszczur@gmail.com>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

[†] Checkpoints and inference code are available on Hugging Face. Codebase used to run the experiments can be found on GitHub.



(a)

(b)

Figure 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.

- 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 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 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.

2. Related Work

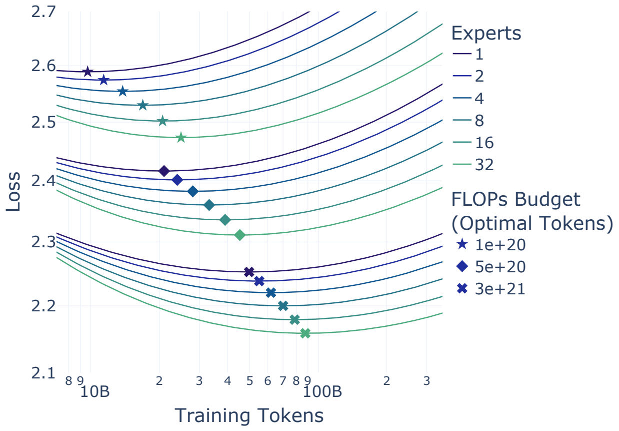
Mixture of Experts. Mixture of Experts (MoE) was introduced by Jacobs et al. (1991), who combined a gating network with a set of expert networks. Shazeer et al. (2017) applied MoE to an LSTM-based model (Hochreiter & Schmidhuber, 1997), scaling the architecture up to 137 billion parameters. In Transformer-based LLMs, MoE is most often applied as a replacement for the feed-forward layer (Lepikhin et al., 2020; Shazeer et al., 2018). It replaces the feed-forward's MLP with a set of expert MLPs along with a router, which selects one or more MLPs for each token. With the recent surge in LLM research, MoE models are gaining even more traction. This is exemplified by the development of extremely large-scale models such as DeepSeek-R1 and Qwen2.5-Max (DeepSeek-AI et al., 2025; Team, 2024a). Apart from language, MoE has also been shown to be an effective measure of scaling in vision (Riquelme et al., 2021).

In our work, we use the standard Switch MoE layer (Fedus et al., 2022), which routes each token to one expert and encourages even token-to-expert assignment via the addition of a differentiable load-balancing loss.

Scaling Laws. Scaling laws refer to empirically derived equations that relate model loss to factors such as the number of parameters, the quantity of training data, or the computational budget. For dense Transformers, scaling laws were initially explored by Hestness et al. (2017) and Kaplan et al. (2020), who identified power-law relationships between the final loss, model size, and dataset size. Hoffmann et al. (2022) expanded this by incorporating variable cosine cycle lengths and adjusting the functional form of the equation:

$$\mathcal{L}(N_{\text{act}}, D) = mN_{\text{act}}^{\mu} + nD^{\nu} + c. \tag{2}$$

Scaling laws have also been applied to other architectures and training setups. Zhai et al. (2022) derived scaling laws for Vision Transformers, scaling the models up to 2B parameters. Henighan et al. (2020) examined autoregressive modeling across multiple modalities, while Ghorbani et al. (2021) focused on machine translation. Frantar et al. (2023) studied the effects of pruning on vision and language Transformers, determining optimal sparsity given a fixed compute budget.



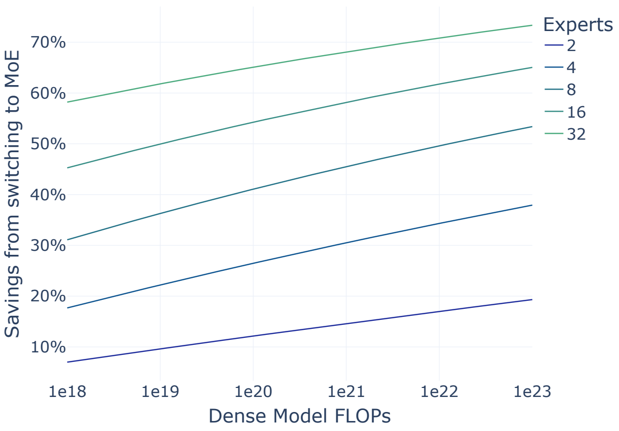


Figure 2. **(a)** IsoFLOP profiles for selected training budgets. Compute optimal points are marked for each curve. **(b)** Savings from switching from a compute-optimal dense model to MoE . The advantage of using MoE increases with larger models and expert counts.

Clark et al. (2022) investigated scaling in MoE models, varying model size and the number of experts on a fixed dataset, and concluded that routed models are more efficient only up to a certain size. Their formula took the form:

$$\mathcal{L}(N_{\text{act}}, \hat{E}) = a\hat{E}^{\delta} N_{\text{act}}^{\alpha+\gamma\ln(\hat{E})}, \qquad (3)$$

where $\hat{E}$ is a monotonic transformation of the number of experts E defined as:

$$\frac{1}{\hat{E}} = \frac{1}{E - 1 + \left(\frac{1}{E_{\text{start}}} - \frac{1}{E_{\max}}\right)^{-1}} + \frac{1}{E_{\max}}. \qquad (4)$$

These analyses have since been extended by Ludziejewski et al. (2024) and Dai et al. (2024), who considered variable dataset size as well as the granularity of experts. In our work, we keep the experts non-granular; however, we treat the number of experts and the number of training tokens as variables. Sardana et al. (2024) assumes a fixed joint inference and training budget. We make similar assumptions; however, we consider accelerator memory as a limiting factor and extend the analysis to MoE models, which can serve as a more compute-friendly alternative to dense models. Yun et al. (2024) have focused on MoE inference optimality and measuring real hardware efficiency.

3. 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., 2024; Ludziejewski et al., 2024), we postulate the following form of the equation:

$$\mathcal{L}(N_{\text{act}}, D, E) = m(E)N_{\text{act}}^{\mu(E)} + n(E)D^{\nu(E)} + c(E), \qquad (5)$$

assuming that if we fix the number of experts the model performance can be described using Equation 2. 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):

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

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 (Ludziejewski et al., 2024).

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 5, this yields a functional form identical to Equation 3.

Empirically, we observe a good fit for our formula, as described in Section 5. 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 use a transformation of the number of experts $\hat{E}$ given in Equation 4.

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

$$\mathcal{L}(N_{\text{act}}, D, \hat{E}) = a\hat{E}^{\delta} N_{\text{act}}^{\alpha+\gamma\ln(\hat{E})} + b\hat{E}^{\omega} D^{\beta+\zeta\ln(\hat{E})} + c. \tag{6}$$

We fit the coefficients in Equation 6 based on the results of our experiments; see Table 3. In Section 4, 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 5.

4. 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 Appendix A for details on counting FLOPs, the relations between active and total parameters, and other technical details.

4.1. 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:

$$\underset{N_{\text{act}}, D, E}{\arg\min}\, \mathcal{L}(N_{\text{act}}, D, E) \tag{7}$$
$$\text{s.t. } 6N_{\text{act}}D = F \tag{8}$$

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 2 (a).

Table 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.

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

For any fixed E our scaling law has the Chinchilla functional form of Equation 2. 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

$$N_{\text{act}}^{\text{opt}}(F) = G\left(\frac{F}{6}\right)^{a}, \quad D^{\text{opt}}(F) = G^{-1}\left(\frac{F}{6}\right)^{b}, \tag{9}$$

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.C. We compare the optimal configurations for several compute budgets in Table 1.

Both from comparing the IsoFLOP slices (Figure 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.

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 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 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 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.

4.2. 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 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:

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

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 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 1). Moreover, such an MoE model is more compute- *and* memory-efficient at inference.

4.3. 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 size yields the optimization criterion:

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

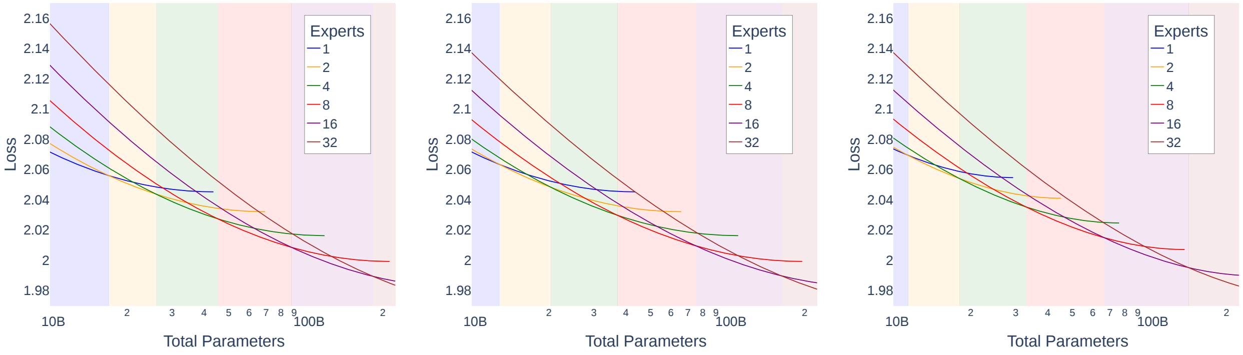
For practical values of T, a fair comparison of memory requirements should include the size of KV cache in addition to the model size. Figure 3 (b) presents the optimal models for a given compute and varying memory constraints when the size of the KV cache is included. Importantly, MoE models compare more favorably to dense models in this graph, and as T increases, they outperform dense models at ever smaller model sizes. In Figure 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.

4.4. 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:

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

Figure 3 (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



(a) (b) (c)

Figure 3. Loss predicted for various expansion rates at a FLOPs budget $F = 5\mathrm{e}22$. 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.

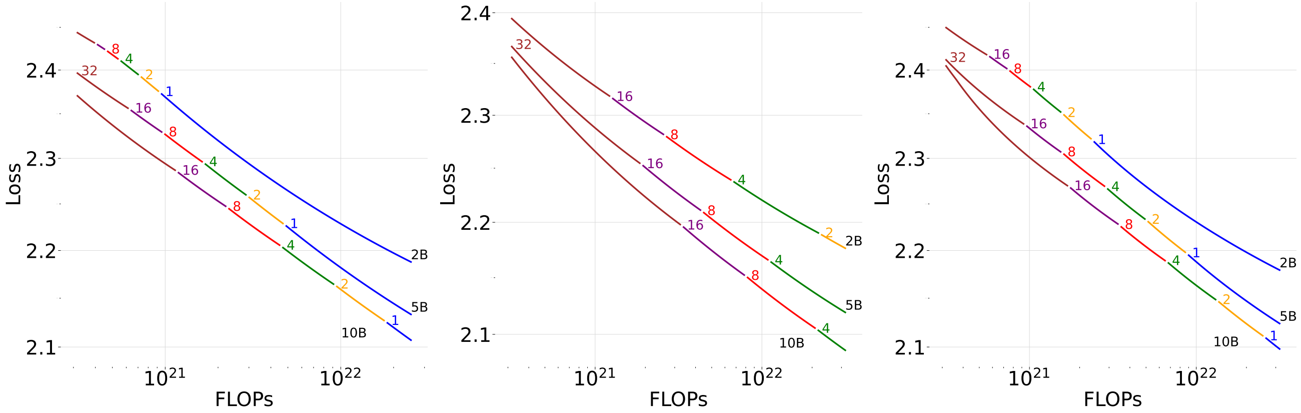


Figure 4. Investigation of the optimal number of experts for three different model sizes, 2B, 5B, and 10B; and in three different scenarios, from left to right: simply measuring the size of the model, including the size of a KV-cache with 32k tokens, and including the inference cost of processing 100B tokens.

at smaller scales than in simple compute-optimality due to decreased inference FLOPs. The $E = 2$ and $E = 4$ models shown in the Figure 1 use respectively 36% and 61% less FLOPs per token than their dense counterpart.

4.5. Summary

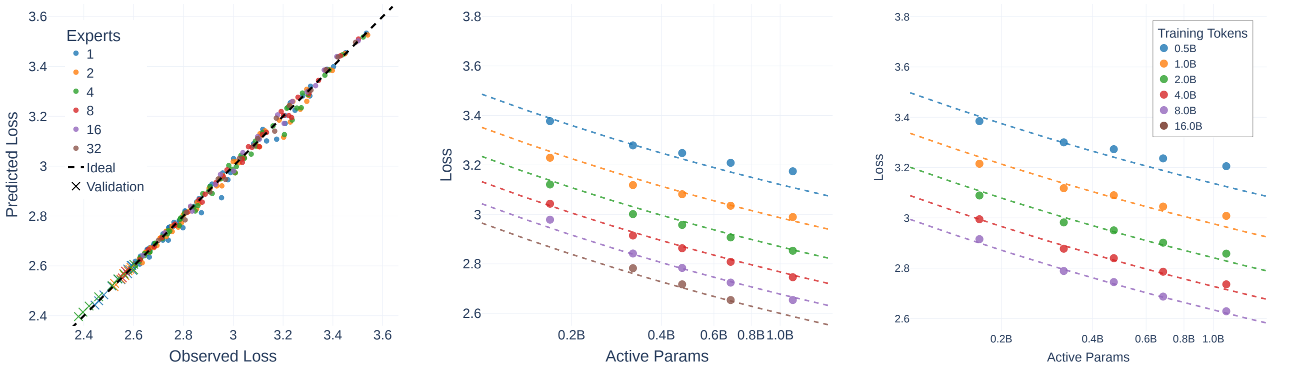
The notions of inference optimality and total memory optimality can naturally be combined. Figure 3 (c) presents a comparison between different numbers of experts, where the KV cache is included in the model's memory requirements and the compute budget is shared between training and inference. Finally, Figure 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 1 & 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.



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

Figure 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.

Table 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.

	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

5. Fitting the Scaling Law

In this section, we present details of experiments and the procedure of fitting the scaling law parameters, see Table 3 in the Appendix. 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 Appendix H.

5.1. Model Hyperparameters

The selection of hyperparameters and training details is crucial for ensuring the robustness of scaling laws (Porian et al., 2024; Pearce & Song, 2024). 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., 2024). 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., 2018) 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, 2020) 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.

5.1.1. 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., 2024). 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.

5.1.2. 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., 2022) 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:

$$LR(N_{act \setminus e}, E) = \exp(8.39 - 0.81 \ln(N_{act \setminus e}) - 0.25 \ln(E)), \tag{10}$$

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 8 in Appendix.

> **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 8 in Appendix).

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 8 in Appendix 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 9 in Appendix, 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 Appendix G.

5.1.3. 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.

5.2. Optimization of Formula Coefficients

Following Hoffmann et al. (2022), we use the LBFGS algorithm to optimize the coefficients of formula 6. See Appendix B 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 2 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 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.

6. 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 (Dai et al., 2024; Ludziejewski 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 (2) 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.

7. 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.

Acknowledgments

We would like to express sincere gratitude to Szymon Antoniak and Piotr Padlewski for their detailed comments and invaluable discussions. We also thank Konrad Staniszewski for his feedback on the draft of this paper.

We gratefully acknowledge the Polish high-performance computing infrastructure PLGrid (HPC Center: ACK Cyfronet AGH) for providing computer facilities and support within the computational grant no. PLG/2024/017060. This research was partially supported by the National Science Centre (Poland) grant Sonata Bis 2024/54/E/ST6/00388, the National Science Centre (NCN) Grant no. 2020/37/B/ST6/04179, the National Science Centre (NCN) Preludium Grant no. 2022/45/N/ST6/02222, the "European Lighthouse of AI for Sustainability" - ELIAS grant no. 101120237, and the NCBiR grant POIR.01.01.01-00-0433/20. Part of the experiments utilized computational resources provided by Writer.

Impact Statement

Recent rapid advancements in language models have introduced numerous new capabilities while simultaneously sparking concerns about their societal impact and prompting intense discussions within the community. Our work enhances the understanding of scaling laws and offers guidance on improving the efficiency of LLMs, potentially amplifying existing risks by extending LLM capabilities. However, our approach does not introduce any novel threats. Therefore, we refer to the existing literature on the broader impact of language models (e.g., Borgeaud et al. (2022)).

References

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.

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.

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, 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.

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., et al. The llama 3 herd of models. arXiv preprint arXiv:2407.21783, 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.

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.

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.

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.

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.

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

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.

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.

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, 2024.

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.

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.

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.

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.

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

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, 2024.

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.

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

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 and insights from training gopher, 2022.

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., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/48237d9f2dea8c74c2a72126cf63d933-Paper.pdf.

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, 2024.

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

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

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/.

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., 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.

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.

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, 2022.

A. Technical Details

A.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:

$$N_{\text{total}} = 2 d_{\text{model}} d_{\text{vocab}} + (4 + 9E) N_{\text{blocks}} d_{\text{model}}^2 \tag{11}$$

$$N_{\text{act}} = 2 d_{\text{model}} d_{\text{vocab}} + 13 N_{\text{blocks}} d_{\text{model}}^2 \tag{12}$$

A.2. Counting FLOPs

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

A.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).

B. Fit Details

Table 3. Fitted coefficients of our joined formula.

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 4. The fitted coefficients of our joint formula, Equation (6), reduced to the Chinchilla scaling law, Equation (2), 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.

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

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 6, optimizing the Huber loss with $\delta = 0.01$ over the set of our training runs described in table in Appendix H. 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 6 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 the training and a held-out extrapolation validation set. The formula in Equation 6 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.

C. Derivation of $N_{\text{act}}^{\text{opt}}$ and D^{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:

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

After rearranging:

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

The derivation of D^{opt} is analogous.

D. 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.

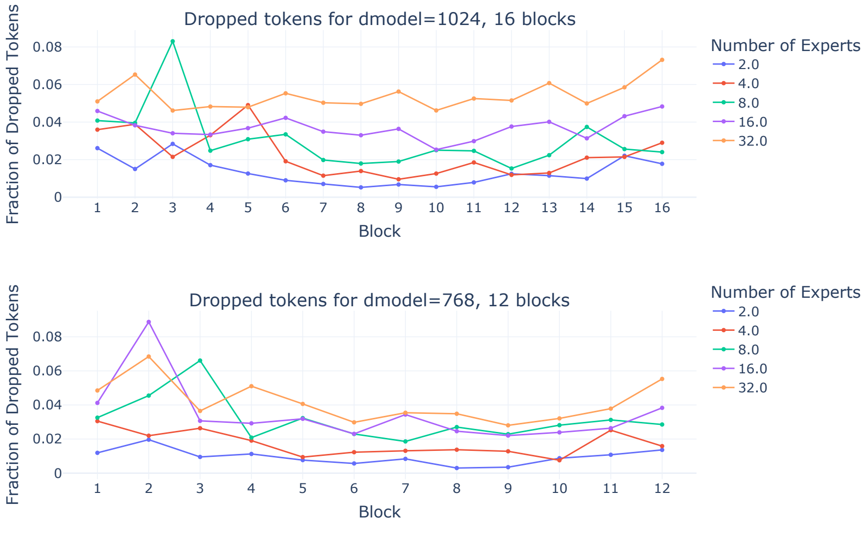


Figure 6. Dropped tokens for selected models.

E. 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 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.

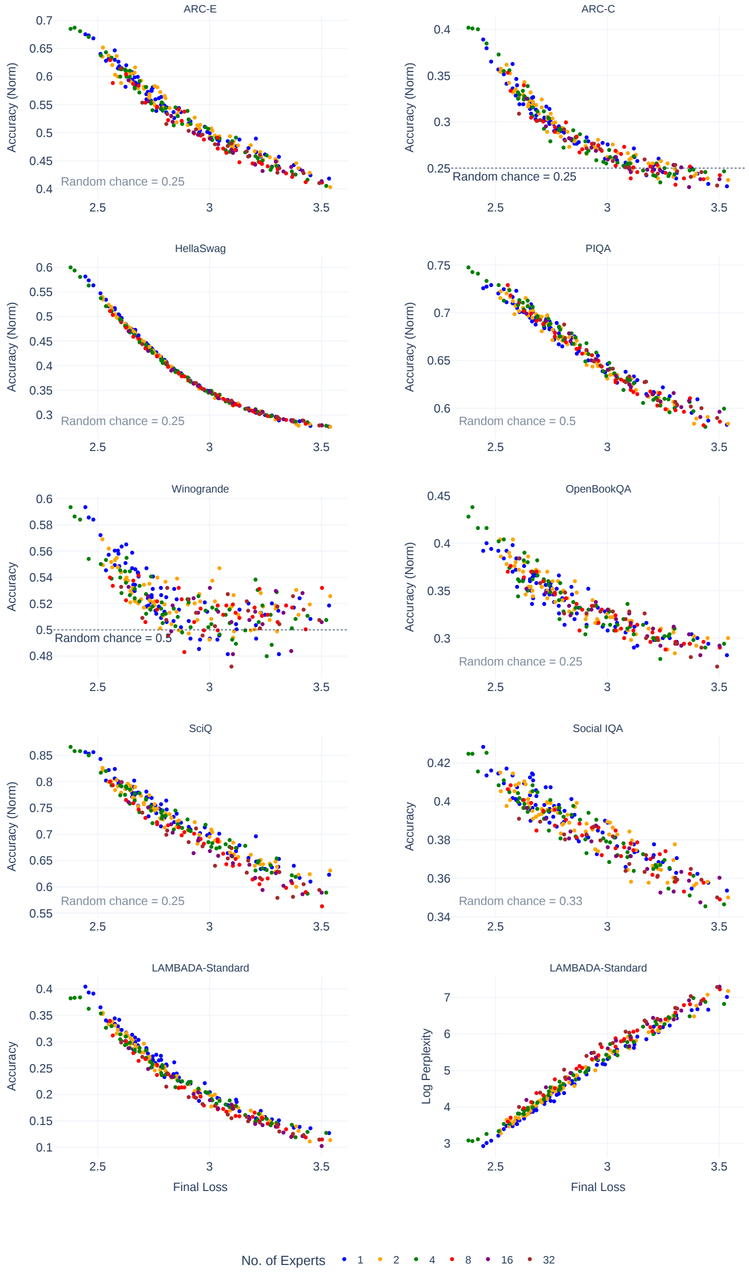


Figure 7. Downstream performance of the models.

F. 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 5. We sample 80% of data 100 times and report the 10th and 90th percentiles.

Table 5. Bootstrap intervals for the optimal $N_{\text{act}}^{\text{opt}}$ and D^{opt} across training budgets and expert counts.

Training Budget	**Experts**	$\mathbf{N_{\text{act}}^{\text{opt}}}$ **(90% interval)**	$\mathbf{D^{\text{opt}}}$ **(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

G. Learning Rate Scaling Fit

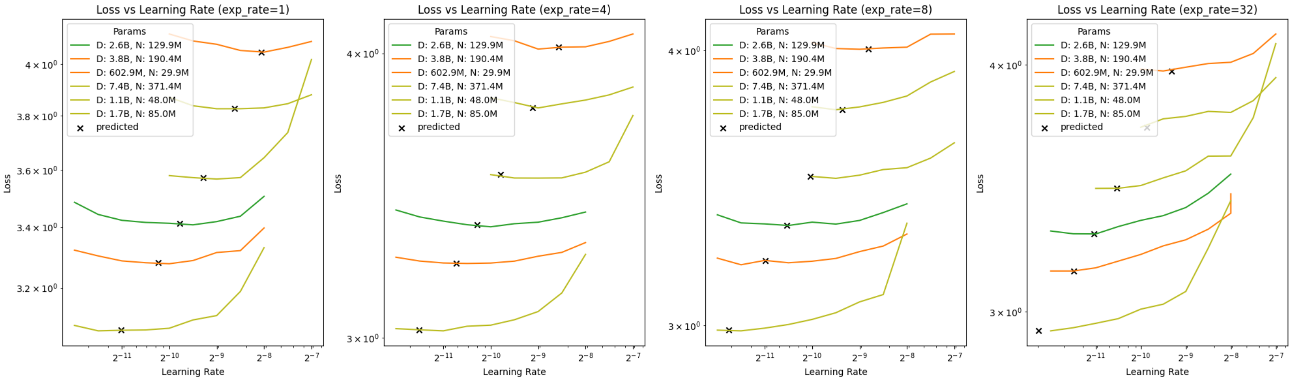


Figure 8. Visualization of the fit ($E \in \{1, 8\}$) of our LR scaling rule, interpolation ($E = 4$) and extrapolation ($E = 32$).

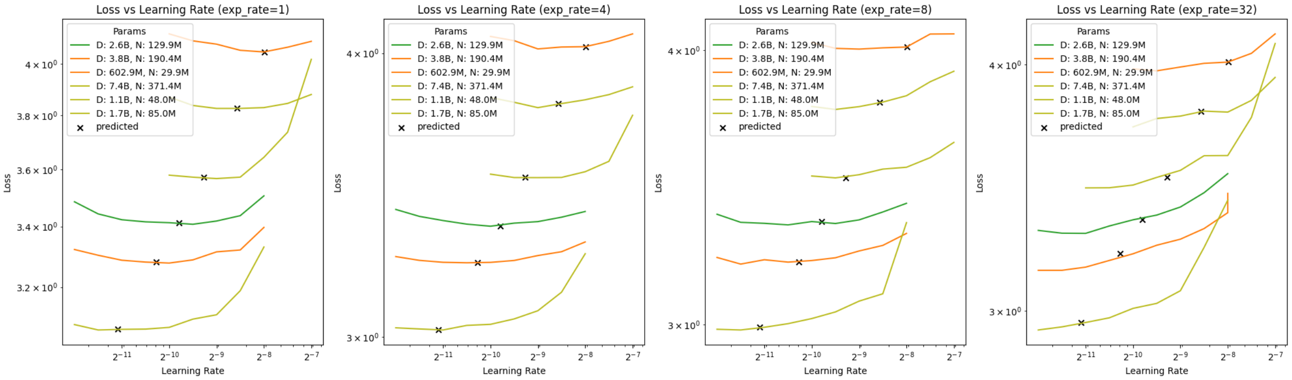


Figure 9. Ablation for the LR scaling rule fit without considering the number of experts E. While performance on the training set ($E \in \{1, 8\}$) looks acceptable, the extrapolation on $E = 32$ is clearly suboptimal, validating the need for considering E.

H. Experiments Listing

N_{total}	$\mathrm{N}_{\text{attn\_heads}}$	$\mathrm{N}_{\text{blocks}}$	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, 500M
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