Training Compute-Optimal Large Language Models

Jordan Hoffmann* **Sebastian Borgeaud*** **Arthur Mensch*** **Elena Buchatskaya**

Trevor Cai **Eliza Rutherford** **Diego de Las Casas** **Lisa Anne Hendricks**

Johannes Welbl **Aidan Clark** **Tom Hennigan** **Eric Noland** **Katie Millican**

George van den Driessche **Bogdan Damoc** **Aurelia Guy** **Simon Osindero**

Karen Simonyan **Erich Elsen** **Oriol Vinyals** **Jack W. Rae** **Laurent Sifre***

* Equal contributions

DeepMind
`(sborgeaud|amensch|sifre)@deepmind.com`

Abstract

We investigate the optimal model size and number of tokens for training a Transformer language model under a given compute budget. We find that current large language models are significantly undertrained, a consequence of the recent focus on scaling language models whilst keeping the amount of training data constant. By training over 400 language models ranging from 70 million to over 16 billion parameters on 5 to 500 billion tokens, we find that for compute-optimal training, the model size and the number of training tokens should be scaled equally: for every doubling of model size the number of training tokens should also be doubled. We test this hypothesis by training a predicted compute-optimal model, *Chinchilla*, that uses the same compute budget as *Gopher* but with 70B parameters and 4× more more data. *Chinchilla* uniformly and significantly outperforms *Gopher* (280B), GPT-3 (175B), Jurassic-1 (178B), and Megatron-Turing NLG (530B) on a large range of downstream evaluation tasks. This also means that *Chinchilla* uses substantially less compute for fine-tuning and inference, greatly facilitating downstream usage. As a highlight, *Chinchilla* reaches a state-of-the-art average accuracy of 67.5% on the MMLU benchmark, greater than a 7% improvement over *Gopher*.

1 Introduction

A series of *Large Language Models* (LLMs) have recently been introduced [6, 30, 38, 48, 52], with the largest dense language models now having over 500 billion parameters. These large autoregressive transformers [53] have demonstrated impressive performance on many tasks using a variety of evaluation protocols: zero-shot generalization, few-shot training, and as a basis for fine-tuning. The compute and energy cost for training large language models is substantial [38, 52] and rises with increasing model size. In practice, the allocated training compute budget is often known in advance: practitioners have access to a certain number of accelerators for a given period of time. Since it

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

https://doi.org/10.52202/068431-2176

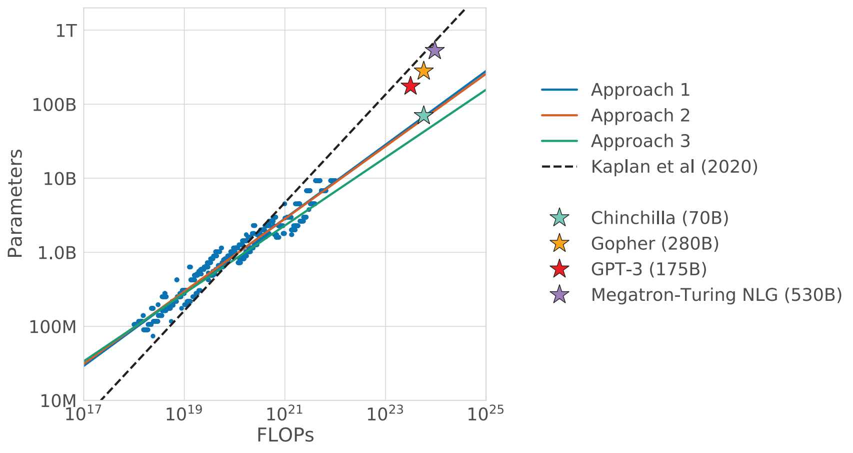

Figure 1: **Overlaid predictions.** We overlay the predictions from our three different approaches, along with projections from [23]. We find that all three methods predict that current large models should be substantially smaller and therefore trained much longer than is currently done. In Figure A3, we show the results with the predicted optimal tokens plotted against the optimal number of parameters for fixed FLOP budgets. ***Chinchilla* outperforms *Gopher* and the other large models (see Section 4.2).**

Table 1: **Current LLMs**. We show five of the current largest dense transformer models, their size, and the number of training tokens. Other than LaMDA [52], most models are trained for approximately 300 billion tokens. We introduce *Chinchilla*, a substantially smaller model, trained for much longer than 300B tokens. Table A3 shows our projected optimal relation between model size and tokens.

Model	Size (# Parameters)	Training Tokens
LaMDA [52]	137 Billion	768 Billion
GPT-3 [6]	175 Billion	300 Billion
Jurassic [30]	178 Billion	300 Billion
Gopher [38]	280 Billion	300 Billion
MT-NLG 530B [48]	530 Billion	270 Billion
Chinchilla	70 Billion	1.4 Trillion

is typically only feasible to train these large models once, accurately estimating the best model hyperparameters for a given compute budget is critical [51].

Kaplan et al. [23] showed that there is a power law relationship between the number of parameters in an autoregressive language model (LM) and its performance (measured in evaluation perplexity). One notable conclusion in [23] is that large models should not be trained to their lowest possible loss to be compute optimal; they argue that model size should grow faster than the size of the training set for a given increase of computational budget. As a result, the field has been training larger and larger models while keeping the size of the training set to approximately 300 billion tokens, expecting performance improvements (Table 1). While we find that there is effectively a trade-off between model size and training set size, we estimate that large models should be trained for many more training tokens than recommended by [23]. Specifically, given a $10\times$ increase computational budget we find that model size and the number of training tokens should be scaled in equal proportions.

In this work, we revisit the question: *Given a fixed FLOPs budget,*[1] *how should one trade-off model size and the number of training tokens?* To answer this question, we model the final pre-training loss[2] $L(N, D)$ as a function of the number of model parameters N, and the number of training tokens, D. Since the computational budget C is a deterministic function $\text{FLOPs}(N, D)$ of the number of seen training tokens and model parameters, we are interested in minimizing L under the constraint

[1]For example, knowing the number of accelerators and a target training duration.

[2]For simplicity, we perform our analysis on the smoothed training loss which is an unbiased estimate of the test loss, as the number of training tokens is less than the number of tokens in the entire corpus.

$\text{FLOPs}(N, D) = C$:

$$N_{opt}(C), D_{opt}(C) = \underset{N,D \text{ s.t. FLOPs}(N,D)=C}{\text{argmin}} L(N, D). \quad (1)$$

The functions $N_{opt}(C)$, and $D_{opt}(C)$ describe the optimal allocation of a computational budget C. We empirically estimate these functions based on the losses of over 400 models, ranging from under 70M to over 16B parameters, and trained on 5B to over 400B tokens – with each model configuration trained for several different training horizons. Our approach leads to considerably different results than that of [23]. We highlight our results in Figure 1 and how our approaches differ in Section 2.

Based on our estimated compute-optimal frontier, we predict that for the compute budget used to train *Gopher*, an optimal model should be 4 times smaller, while being training on 4 times more tokens. We verify this by training a more *compute-optimal* 70B model, called *Chinchilla*, on 1.4 trillion tokens. Not only does *Chinchilla* outperform its much larger counterpart, *Gopher*, but its reduced model size reduces inference cost considerably and greatly facilitates downstream uses on smaller hardware. The energy cost of a large language model is amortized through its usage for inference and fine-tuning. The benefits of a more optimally trained smaller model, therefore, extend beyond the immediate benefits of its improved performance.

2 Related Work

Large language models. A variety of large language models have been introduced in the last few years. These include both dense transformer models [6, 30, 48, 38, 52] and mixture-of-expert (MoE) models [11, 12, 60]. The largest dense transformers have passed 500 billion parameters [48, 8]. The drive to train larger and larger models is clear—so far increasing the size of language models has been responsible for improving the state-of-the-art in many language modelling tasks. Nonetheless, large language models face several challenges, including their overwhelming computational requirements (the cost of training and inference increase with model size) [38, 52] and the need for acquiring more high-quality training data. In fact, in this work we find that larger, high quality datasets will play a key role in any further scaling of language models. Concurrent to our work, a 540 billion parameter model trained on 768 billion tokens was released– PaLM [8]. While this model outperforms *Chinchilla*, it uses approximately $5\times$ the compute and is nearly $8\times$ larger, making it more difficult to use.

Modelling the scaling behavior. Understanding the scaling behaviour of language models and their transfer properties has been important in the development of recent large models [23, 18]. Kaplan et al. [23] first showed a predictable relationship between model size and loss over many orders of magnitude. The authors investigate the question of choosing the optimal model size to train for a given compute budget. Similar to us, they address this question by training various models. Our work differs from Kaplan et al. [23] in several important ways. First, the authors use a fixed number of training tokens and learning rate schedule for all models; this prevents them from modelling the impact of these hyperparameters on the loss. In contrast, we find that setting the learning rate schedule to approximately match the number of training tokens results in the best final loss regardless of model size—see Figure A1. For a fixed learning rate cosine schedule to 130B tokens, the intermediate loss estimates (for $D' << 130\text{B}$) are therefore overestimates of the loss of a model trained with a schedule length matching D'. Using these intermediate losses results in underestimating the effectiveness of training models on less data than 130B tokens, and eventually contributes to the conclusion that model size should increase faster than training data size as compute budget increases. In contrast, our analysis predicts that both quantities should scale at roughly the same rate. Secondly, we include models with up to 16B parameters, as we observe that there is slight curvature in the FLOP-loss frontier (see Appendix E)—in fact, the majority of the models used in our analysis have more than 500 million parameters, in contrast the majority of runs in [23] are significantly smaller—many being less than 100M parameters. Clark et al. [9] specifically looked in to the scaling properties of Mixture of Expert language models, showing that the scaling with number of experts diminishes as the model size increases—their approach models the loss as a function of two variables: the model size and the number of experts. However, the analysis is done with a fixed number of tokens, potentially underestimating the improvements of branching.

Estimating hyperparameters for large models. The model size and the number of training tokens are not the only two parameters to chose when selecting a language model and a procedure to train

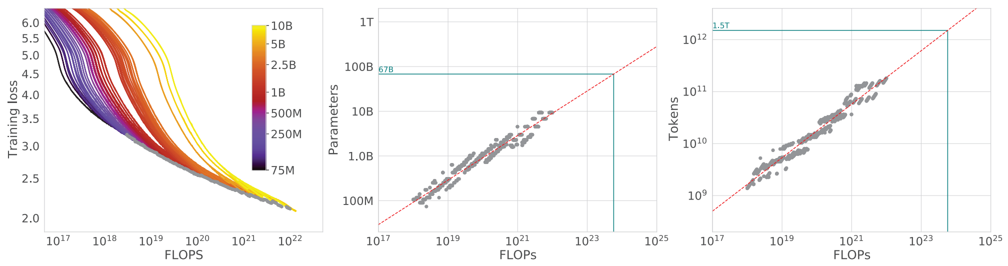

Figure 2: **Training curve envelope.** On the **left** we show all of our different runs. We launched a range of model sizes going from 70M to 10B, each for four different cosine cycle lengths. From these curves, we extracted the envelope of minimal loss per FLOP, and we used these points to estimate the optimal model size (**center**) for a given compute budget and the optimal number of training tokens (**right**). In green, we show projections of optimal model size and training token count based on the number of FLOPs used to train *Gopher* (5.76×10^{23}).

it. Other important factors include learning rate, learning rate schedule, batch size, optimiser, and width-to-depth ratio. In this work, we focus on model size and the number of training steps, and we rely on existing work and provided experimental heuristics to determine the other necessary hyperparameters. Yang et al. [57] investigates how to choose a variety of these parameters for training an autoregressive transformer, including the learning rate and batch size. McCandlish et al. [33] finds only a weak dependence between optimal batch size and model size. Shallue et al. [46], Zhang et al. [59] suggest that using larger batch-sizes than those we use is possible. Levine et al. [28] investigates the optimal depth-to-width ratio for a variety of standard model sizes. We use slightly less deep models than proposed as this translates to better wall-clock performance on our hardware.

Improved model architectures. Recently, various promising alternatives to traditional dense transformers have been proposed. For example, through the use of conditional computation large MoE models like the 1.7 trillion parameter Switch transformer [12], the 1.2 Trillion parameter GLaM model [11], and others [1, 60] are able to provide a large effective model size despite using relatively fewer training and inference FLOPs. However, for very large models the computational benefits of routed models seems to diminish [9]. An orthogonal approach to improving language models is to augment transformers with explicit retrieval mechanisms, as done by [4, 15, 29]. This approach effectively increases the number of data tokens seen during training (by a factor of ~ 10 in [4]). This suggests that the performance of language models may be more dependant on the size of the training data than previously thought.

3 Estimating the optimal parameter/training tokens allocation

We present three different approaches to answer the question driving our research: *Given a fixed FLOPs budget, how should one trade-off model size and the number of training tokens?* In all three cases we start by training a range of models varying both model size and the number of training tokens and use the resulting training curves to fit an empirical estimator of how they should scale. We assume a power-law relationship between compute and model size as done in [9, 23], though future work may want to include potential curvature in this relationship for large model sizes. The resulting predictions are similar for all three methods and suggest that parameter count and number of training tokens should be increased equally with more compute —with proportions reported in Table 2. This is in clear contrast to previous work on this topic and warrants further investigation.

3.1 Approach 1: Fix model sizes and vary number of training tokens

In our first approach we vary the number of training steps for a fixed family of models (ranging from 70M to over 10B parameters), training each model for 4 different number of training sequences. From these runs, we are able to directly extract an estimate of the minimum loss achieved for a given number of training FLOPs. Training details for this approach can be found in Appendix D.

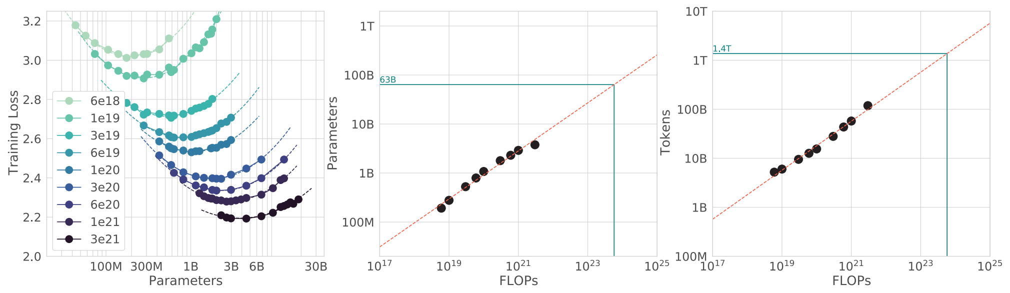

Figure 3: **IsoFLOP curves.** For various model sizes, we choose the number of training tokens such that the final FLOPs is a constant. The cosine cycle length is set to match the target FLOP count. We find a clear valley in loss, meaning that for a given FLOP budget there is an optimal model to train (**left**). Using the location of these valleys, we project optimal model size and number of tokens for larger models (**center** and **right**). In green, we show the estimated number of parameters and tokens for an *optimal* model trained with the compute budget of *Gopher*.

For each parameter count N we train 4 different models: each uses a different horizon (measured in number of training tokens) over which we decay the learning rate by a factor of $10\times$; the range of horizons varies by a factor of $16\times$ for each parameter count. We smooth and linearly interpolate each training loss curve. From this, we obtain a continuous mapping from FLOP count to training loss for each run. We then determine which run achieves the lowest loss for each FLOP count. Using these interpolants, we obtain a mapping from FLOP count C to the most efficient choice of model size N_{opt} and number of training tokens D_{opt} such that $\text{FLOPs}(N_{opt}, D_{opt}) = C$.[3] We apply this mapping onto logarithmically spaced values of C and obtain many empirical triplets $(C_i, N_{opt,i}, D_{opt,i})_i$. Finally, we fit power laws to these empirical data, estimating a and b such that $N_{opt} \propto C^a$ and $D_{opt} \propto C^b$. We find that $a = 0.50$ and $b = 0.50$—as summarized in Table 2. We perform a simple experiment for early validating our analysis: given a budget of 10^{21} FLOPs, we compare the performance of training a model with a size recommended by our analysis to training a model with a size suggested by the analysis of [23]—using the model size we predict has a clear advantage (Section D.4).

3.2 Approach 2: IsoFLOP profiles

In our second approach we vary the model size for a fixed set of 9 different training FLOP counts (ranging from 6×10^{18} to 3×10^{21} FLOPs), and consider the final training loss for each point. This differs from the first approach that considers points $(N_i, D_i, L_i)_i$ along the entire training runs; the data points are here scarcer but more representative of the performance of a fully trained model.

For each FLOP budget, we plot the final loss (after smoothing) against the parameter count in Figure 3 (left). In all cases, we ensure that we have trained a diverse enough set of model sizes to see a clear minimum in the loss. We fit a parabola to each IsoFLOPs curve to directly estimate at what model size the minimum loss is achieved (Figure 3 (left)). As with the previous approach, we then fit a power law between FLOPs and loss-optimal model size and number of training tokens, shown in Figure 3 (center, right). Again, we fit exponents of the form $N_{opt} \propto C^a$ and $D_{opt} \propto C^b$ and we find that $a = 0.49$ and $b = 0.51$—as summarized in Table 2.

3.3 Approach 3: Fitting a parametric loss function

Lastly, we model all final losses from experiments in Approach 1 & 2 as a parametric function of model parameter count and the number of seen tokens. Following a classical risk decomposition (see Section D.2), we propose the following functional form

$$\hat{L}(N, D) \triangleq E + \frac{A}{N^\alpha} + \frac{B}{D^\beta}. \tag{2}$$

The first term captures the loss for an ideal generative process on the data distribution, and should correspond to the entropy of natural text. The second term captures the fact that a perfectly trained

[3]Note that all selected points are within the last 15% of training. Thus, when training a model over D tokens, we should pick a cosine cycle length that decays $10\times$ over approximately D tokens—see Appendix B.

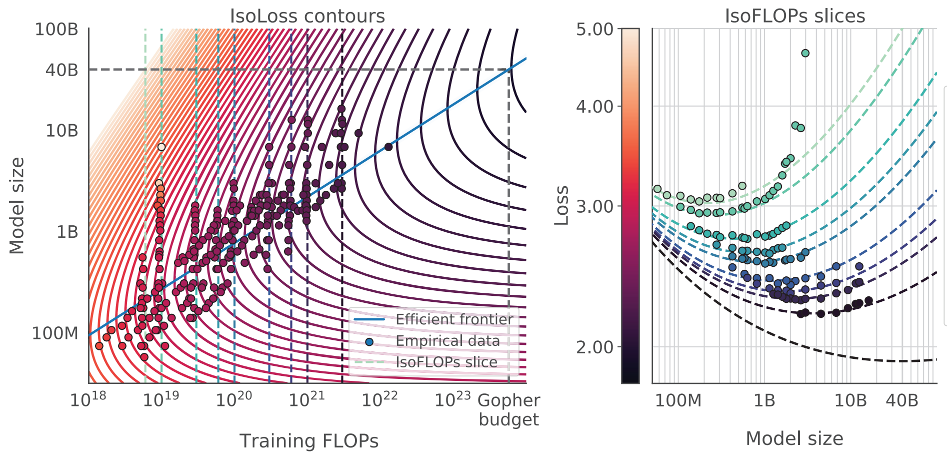

Figure 4: **Parametric fit.** We fit a parametric modelling of the loss $\hat{L}(N, D)$ and display contour (**left**) and isoFLOP slices (**right**). For each isoFLOP slice, we include a corresponding dashed line in the left plot. In the left plot, we show the efficient frontier in blue, which is a line in log-log space. Specifically, the curve goes through each iso-loss contour at the point with the fewest FLOPs. We project the optimal model size given the *Gopher* FLOP budget to be 40B parameters.

transformer with N parameters underperforms the ideal generative process. The final term captures the fact that the transformer is not trained to convergence, as we only make a finite number of optimisation steps, on a sample of the dataset distribution.

Model fitting. To estimate (A, B, E, α, β), we minimize the Huber loss [19] between the predicted and observed log loss using the L-BFGS algorithm [36]:

$$\min_{A,B,E,\alpha,\beta} \sum_{\text{Runs } i} \text{Huber}_\delta \Big(\log \hat{L}(N_i, D_i) - \log L_i \Big) \tag{3}$$

We account for possible local minima by selecting the best fit from a grid of initialisations. The Huber loss ($\delta = 10^{-3}$) is robust to outliers, which we find important for good predictive performance over held-out data points. Section D.2 details the fitting procedure and the loss decomposition.

Efficient frontier. We approximate the functions N_{opt} and D_{opt} by minimizing the parametric loss $\hat{L}$ under the constraint $\text{FLOPs}(N, D) \approx 6ND$ [23]. The resulting N_{opt} and D_{opt} balance the two terms in Equation (3) that depend on model size and data. By construction, they have a power-law form. We show contours of the fitted function $\hat{L}$ in Figure 4 (left), and the closed-form efficient computational frontier in blue. From this approach, we find that $a = 0.46$ and $b = 0.54$—as summarized in Table 2.

3.4 Optimal model scaling

We find that the three approaches, despite using different fitting methodologies and different trained models, yield comparable predictions for the optimal scaling in parameters and tokens with FLOPs (shown in Table 2). All three approaches suggest that as compute budget increases, model size and the amount of training data should be increased in approximately equal proportions. The first and second approaches yield very similar predictions for optimal model sizes, as shown in Figure 1 and Figure A3. The third approach predicts even smaller models being optimal at larger compute budgets. We note that the observed points (L, N, D) for low training FLOPs ($C \leq 1e21$) have larger residuals $\|L - \hat{L}(N, D)\|_2^2$ than points with higher computational budgets. The fitted model places increased weight on the points with more FLOPs—automatically considering the low-computational budget points as outliers due to the Huber loss. As a consequence of the empirically observed negative curvature in the frontier $C \to N_{opt}$ (see Appendix E), this results in predicting a lower N_{opt} than the two other approaches.

Table 2: **Estimated parameter and data scaling with increased training compute.** The listed values are the exponents, a and b, on the relationship $N_{opt} \propto C^a$ and $D_{opt} \propto C^b$. Our analysis suggests a near equal scaling in parameters and data with increasing compute which is in clear contrast to previous work on the scaling of large models. The 10th and 90th percentiles are estimated via bootstrapping data (80% of the dataset is sampled 100 times) and are shown in parenthesis.

Approach	Coeff. a where $N_{opt} \propto C^a$	Coeff. b where $D_{opt} \propto C^b$
1. Minimum over training curves	$0.50\ (0.488, 0.502)$	$0.50\ (0.501, 0.512)$
2. IsoFLOP profiles	$0.49\ (0.462, 0.534)$	$0.51\ (0.483, 0.529)$
3. Parametric modelling of the loss	$0.46\ (0.454, 0.455)$	$0.54\ (0.542, 0.543)$
Kaplan *et al.* (2020) [23]	0.73	0.27

In Table A3 we show the estimated number of FLOPs and tokens that would ensure that a model of a given size lies on the compute-optimal frontier. Our findings suggests that the current generation of large language models are considerably over-sized, given their respective compute budgets, as shown in Figure 1. Furthermore, the amount of training data that is projected to be needed is far beyond what is currently used to train large models, and underscores the importance of dataset collection in addition to engineering improvements that allow for model scale. While there is significant uncertainty extrapolating out many orders of magnitude, our analysis clearly suggests that given the training compute budget for many current LLMs, smaller models should have been trained on more tokens to achieve the most performant model. In Appendix C, we reproduce the IsoFLOP analysis on two additional datasets: C4 [40] and GitHub code [38]. In both cases we reach the similar conclusion that model size and number of training tokens should be scaled in equal proportions.

4 *Chinchilla*

Based on our analysis in Section 3, the optimal model size for the *Gopher* compute budget is somewhere between 40 and 70 billion parameters. We test this hypothesis by training a model on the larger end of this range—70B parameters—for 1.4T tokens, due to both dataset and computational efficiency considerations. In this section we compare this model, which we call *Chinchilla*, to *Gopher* and other LLMs. Both *Chinchilla* and *Gopher* have been trained for the same number of FLOPs but differ in the size of the model and the number of training tokens. While pre-training a large language model has a considerable compute cost, downstream fine-tuning and inference also make up substantial compute usage [38]. Due to being $4\times$ smaller than *Gopher*, both the memory footprint and inference cost of *Chinchilla* are also smaller.

4.1 Model and training details

The full set of hyperparameters used to train *Chinchilla* are given in Table 3. *Chinchilla* uses the same model architecture and training setup as *Gopher* with the exception of the differences listed below.

- We train *Chinchilla* on *MassiveText* (the same dataset as *Gopher*) but use a slightly different subset distribution (Table A1) to account for the increased number of training tokens.
- We use AdamW [32] for *Chinchilla* rather than Adam [24] as this improves the language modelling loss and the downstream task performance after finetuning.[4]
- We train *Chinchilla* with a slightly modified SentencePiece [25] tokenizer that does not apply NFKC normalisation. The vocabulary is very similar– 94.15% of tokens are the same as those used for training *Gopher*. We find that this particularly helps with the representation of mathematics and chemistry, for example.
- Whilst the forward and backward pass are computed in `bfloat16`, we store a `float32` copy of the weights in the distributed optimiser state [41]. See *Lessons Learned* from [38] for additional details.

[4]A model trained with AdamW only passes the training performance of a model trained with Adam around 80% of the way through the cosine cycle, though the ending performance is notably better– see Figure A7

In Appendix G we show the impact of the various optimiser related changes between *Chinchilla* and *Gopher*. All models in this analysis have been trained on TPUv3/TPUv4 [22] with JAX [5] and Haiku [17]. We include a *Chinchilla* model card [35] in Table A13.

4.2 Results

We perform an extensive evaluation of *Chinchilla*, comparing against various large language models. We evaluate on a large subset of the tasks presented in [38], shown in Table A6. As the focus of this work is on optimal model scaling, we included a large representative subset, and introduce a few new evaluations to allow for better comparison to other existing large models. The evaluation details for all tasks are the same as described in [38].

Language modelling. *Chinchilla* significantly outperforms *Gopher* on all evaluation subsets of The Pile [13], as shown in Figure A8. Compared to Jurassic-1 (178B) [30], *Chinchilla* is more performant on all but two subsets– `dm_mathematics` and `ubuntu_irc`– see Table A7 for a raw bits-per-byte comparison. On Wikitext103 [34], *Chinchilla* reaches 7.16 perplexity compared to 7.75 for *Gopher*.

MMLU. The Massive Multitask Language Understanding (MMLU) benchmark [16] consists of a range of exam-like questions on academic subjects. In Table A8, we report *Chinchilla*'s average 5-shot performance on MMLU (the full breakdown of results is shown in Table A9). On this benchmark, *Chinchilla* significantly outperforms *Gopher* despite being much smaller, with an average accuracy of 67.6% (improving upon *Gopher* by 7.6%). Remarkably, *Chinchilla* even outperforms the expert forecast for June 2023 of 63.4% accuracy (see Table A8) [50]. Furthermore, *Chinchilla* achieves greater than 90% accuracy on 4 different individual tasks– `high_school_gov_and_politics, international_law, sociology`, and `us_foreign_policy`. To our knowledge, no other model has achieved greater than 90% accuracy on a subset. In Figure A9, we show a comparison to *Gopher* broken down by task.

Reading comprehension. On the final word prediction dataset LAMBADA [37], *Chinchilla* achieves 77.4% accuracy, compared to 74.5% accuracy from *Gopher* and 76.6% from MT-NLG 530B (see Table 4). On RACE-h and RACE-m [27], *Chinchilla* greatly outperforms *Gopher*, improving accuracy by more than 10% in both cases—see Table 4.

BIG-bench. We analysed *Chinchilla* on the same set of BIG-bench tasks [49] reported in [38]. Similar to what we observed in MMLU, *Chinchilla* outperforms *Gopher* on the vast majority of tasks (see Figure A10). We find that *Chinchilla* improves the average performance by 10.7%, reaching an accuracy of 65.1% versus 54.4% for *Gopher*. Full accuracy results can be found in Table A10.

Common sense. We evaluate *Chinchilla* on various common sense benchmarks: PIQA [3], SIQA [45], Winogrande [44], HellaSwag [58], and BoolQ [10]. We find that *Chinchilla* outperforms both *Gopher* and GPT-3 on all tasks and outperforms MT-NLG 530B on all but one task—see Table 5. On TruthfulQA [31], *Chinchilla* reaches 43.6%, 58.5%, and 66.7% accuracy with 0-shot, 5-shot, and 10-shot respectively. In comparison, *Gopher* achieved only 29.5% 0-shot and 43.7% 10-shot accuracy. In stark contrast with the findings of [31], the large improvements (14.1% in 0-shot accuracy) achieved by Chinchilla suggest that better modelling of the pre-training data alone can lead to substantial improvements on this benchmark.

Closed-book question answering. Results on closed-book question answering benchmarks are reported in Table A11. On the Natural Questions dataset [26], *Chinchilla* achieves new closed-book SOTA accuracies: 31.5% 5-shot and 35.5% 64-shot, compared to 21% and 28% respectively, for *Gopher*. On TriviaQA [21] we show results for both the filtered (previously used in retrieval and open-

Table 3: ***Chinchilla* architecture details.** We list the number of layers, the key/value size, the bottleneck activation size d_{model}, the maximum learning rate, and the training batch size (# tokens). The feed-forward size is always set to $4 \times d_{\text{model}}$. Note that we double the batch size midway through training for both *Chinchilla* and *Gopher*.

Model	Layers	Number Heads	Key/Value Size	d_{model}	Max LR	Batch Size
Gopher 280B	80	128	128	16,384	4×10^{-5}	3M → 6M
Chinchilla 70B	80	64	128	8,192	1×10^{-4}	1.5M → 3M

Table 4: **Reading comprehension.** On RACE-h and RACE-m [27], *Chinchilla* considerably improves performance over *Gopher*. Note that GPT-3 and MT-NLG 530B use a different prompt format than we do on RACE-h/m, so results are not comparable to *Gopher* and *Chinchilla*. On LAMBADA [37], *Chinchilla* outperforms both *Gopher* and MT-NLG 530B.

	Chinchilla	*Gopher*	GPT-3	MT-NLG 530B
LAMBADA Zero-Shot	**77.4**	74.5	76.2	76.6
RACE-m Few-Shot	**86.8**	75.1	58.1	-
RACE-h Few-Shot	**82.3**	71.6	46.8	47.9

Table 5: **Zero-shot comparison on Common Sense benchmarks.** We show a comparison between *Chinchilla*, *Gopher*, and MT-NLG 530B on various Common Sense benchmarks. We see that *Chinchilla* matches or outperforms *Gopher* and GPT-3 on all tasks. On all but one *Chinchilla* outperforms the much larger MT-NLG 530B model.

	Chinchilla	*Gopher*	GPT-3	MT-NLG 530B	Supervised SOTA
HellaSWAG	**80.8%**	79.2%	78.9%	80.2%	93.9%
PIQA	81.8%	81.8%	81.0%	**82.0%**	90.1%
Winogrande	**74.9%**	70.1%	70.2%	73.0%	91.3%
SIQA	**51.3%**	50.6%	-	-	83.2%
BoolQ	**83.7%**	79.3%	60.5%	78.2%	91.4%

book work) and unfiltered set (previously used in large language model evaluations). In both cases, *Chinchilla* substantially out performs *Gopher*. On the filtered version, Chinchilla lags behind the open book SOTA [20] by 7.9%. On the unfiltered set, *Chinchilla* outperforms GPT-3 (see Table A11).

5 Discussion & Conclusion

The trend so far in large language model training has been to increase the model size, often without increasing the number of training tokens. The largest dense transformer, MT-NLG 530B, is now over $3\times$ larger than GPT-3's 170 billion parameters from just two years ago. However, this model, as well as the majority of existing large models, have all been trained for a comparable number of tokens—around 300 billion. While the desire to train these mega-models has led to substantial engineering innovation, we hypothesize that the race to train larger and larger models is resulting in models that are substantially underperforming compared to what could be achieved with the same compute budget.

We propose three predictive approaches towards optimally setting model size and training duration, based on the outcome of over 400 training runs. All three approaches predict that *Gopher* is substantially over-sized and estimate that for the same compute budget a smaller model trained on more data will perform better. We directly test this hypothesis by training *Chinchilla*, a 70B parameter model, and show that it outperforms *Gopher* and even larger models on nearly every measured evaluation task.

Whilst our method allows us to make predictions on how to scale large models when given additional compute, there are several limitations. Due to the cost of training large models, we only have two comparable training runs at large scale (*Chinchilla* and *Gopher*), and we do not have additional tests at intermediate scales. Furthermore, we assume that the efficient computational frontier can be described by a power-law relationship between the compute budget, model size, and number of training tokens. However, we observe some concavity in $\log(N_{opt})$ at high compute budgets (see Appendix E). This suggests that we may still be overestimating the optimal size of large models. Finally, the training runs for our analysis have all been trained on less than an epoch of data; future work may consider the multiple epoch regime. Despite these limitations, the comparison of *Chinchilla* to *Gopher* validates our performance predictions, that have thus enabled training a better (and more lightweight) model at the same compute budget.

Though there has been significant recent work allowing larger and larger models to be trained, our analysis suggests an increased focus on dataset scaling is needed. Speculatively, we expect that scaling to larger and larger datasets is only beneficial when the data is high-quality. This calls for responsibly collecting larger datasets with a high focus on dataset quality. Larger datasets will require extra care to ensure train-test set overlap is properly accounted for, both in the language modelling loss but also with downstream tasks. Finally, training for trillions of tokens introduces many ethical and privacy concerns. Large datasets scraped from the web will contain toxic language, biases, and private information. With even larger datasets being used, the quantity (if not the frequency) of such information increases, which makes dataset introspection all the more important. *Chinchilla* does suffer from bias and toxicity but interestingly it seems less affected than *Gopher* (see Appendix I).. Better understanding how performance of large language models and toxicity interact is an important future research question.

While we have applied our methodology towards the training of auto-regressive language models, we expect that there is a similar trade-off between model size and the amount of data in other modalities. As training large models is very expensive, choosing the optimal model size and training steps beforehand is essential. The methods we propose are easy to reproduce in new settings.

Acknowledgments and Disclosure of Funding

We'd like to thank Jean-baptiste Alayrac, Kareem Ayoub, Chris Dyer, Nando de Freitas, Demis Hassabis, Geoffrey Irving, Koray Kavukcuoglu, Nate Kushman and Angeliki Lazaridou for useful comments on the manuscript. We'd like to thank Andy Brock, Irina Higgins, Michela Paganini, Francis Song, and other colleagues at DeepMind for helpful discussions. We are also very grateful to the JAX and XLA team for their support and assistance.

References

[1] M. Artetxe, S. Bhosale, N. Goyal, T. Mihaylov, M. Ott, S. Shleifer, X. V. Lin, J. Du, S. Iyer, R. Pasunuru, G. Anantharaman, X. Li, S. Chen, H. Akin, M. Baines, L. Martin, X. Zhou, P. S. Koura, B. O'Horo, J. Wang, L. Zettlemoyer, M. Diab, Z. Kozareva, and V. Stoyanov. Efficient Large Scale Language Modeling with Mixtures of Experts. *arXiv:2112.10684*, 2021.

[2] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the ACM Conference on Fairness, Accountability, and Transparency*, 2021.

[3] Y. Bisk, R. Zellers, J. Gao, Y. Choi, et al. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 2020.

[4] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. van den Driessche, J.-B. Lespiau, B. Damoc, A. Clark, D. de Las Casas, A. Guy, J. Menick, R. Ring, T. Hennigan, S. Huang, L. Maggiore, C. Jones, A. Cassirer, A. Brock, M. Paganini, G. Irving, O. Vinyals, S. Osindero, K. Simonyan, J. W. Rae, E. Elsen, and L. Sifre. Improving language models by retrieving from trillions of tokens. In *Proceedings of the International Conference on Machine Learning*, 2021.

[5] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs. 2018. URL http://github.com/google/jax.

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

[7] S. Bubeck. Convex Optimization: Algorithms and Complexity. *Foundations and Trends in Machine Learning*, 8(3-4):231–357, 2015.

https://doi.org/10.52202/068431-2176

[8] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al. PaLM: Scaling language modeling with pathways. *arXiv:2204.02311*, 2022.

[9] A. Clark, D. d. l. Casas, A. Guy, A. Mensch, M. Paganini, J. Hoffmann, B. Damoc, B. Hechtman, T. Cai, S. Borgeaud, G. v. d. Driessche, E. Rutherford, T. Hennigan, M. Johnson, K. Millican, A. Cassirer, C. Jones, E. Buchatskaya, D. Budden, L. Sifre, S. Osindero, O. Vinyals, J. Rae, E. Elsen, K. Kavukcuoglu, and K. Simonyan. Unified scaling laws for routed language models. In *Proceedings of the International Conference on Machine Learning*, 2022.

[10] C. Clark, K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2924–2936, 2019.

[11] N. Du, Y. Huang, A. M. Dai, S. Tong, D. Lepikhin, Y. Xu, M. Krikun, Y. Zhou, A. W. Yu, O. Firat, B. Zoph, L. Fedus, M. Bosma, Z. Zhou, T. Wang, Y. E. Wang, K. Webster, M. Pellat, K. Robinson, K. Meier-Hellstern, T. Duke, L. Dixon, K. Zhang, Q. V. Le, Y. Wu, Z. Chen, and C. Cui. GLaM: Efficient scaling of language models with mixture-of-experts. *arXiv:2112.06905*, 2021.

[12] W. Fedus, B. Zoph, and N. Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv:2101.03961*, 2021.

[13] L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima, S. Presser, and C. Leahy. The Pile: An 800GB dataset of diverse text for language modeling. *arXiv:2101.00027*, 2020.

[14] S. Gehman, S. Gururangan, M. Sap, Y. Choi, and N. A. Smith. RealToxicityPrompts: Evaluating neural toxic degeneration in language models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2020.

[15] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang. REALM: retrieval-augmented language model pre-training. In *Proceedings of the International Conference on Machine Learning*, pages 3929–3938, July 2020.

[16] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding. *arXiv:2009.03300*, 2020.

[17] T. Hennigan, T. Cai, T. Norman, and I. Babuschkin. Haiku: Sonnet for JAX. 2020. URL http://github.com/deepmind/dm-haiku.

[18] D. Hernandez, J. Kaplan, T. Henighan, and S. McCandlish. Scaling laws for transfer. *arXiv:2102.01293*, 2021.

[19] P. J. Huber. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964.

[20] G. Izacard and E. Grave. Distilling knowledge from reader to retriever for question answering. In *International Conference on Learning Representations*, 2020.

[21] M. Joshi, E. Choi, D. Weld, and L. Zettlemoyer. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. *arXiv:1705.03551*, 2017.

[22] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P.-l. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, D. Killebrew, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, and D. H. Yoon. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the Annual International Symposium on Computer Architecture*, page 1–12, 2017.

[23] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *arXiv:2001.08361*, 2020.

[24] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[25] T. Kudo and J. Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv:1808.06226*, 2018.

[26] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, M. Kelcey, J. Devlin, K. Lee, K. N. Toutanova, L. Jones, M.-W. Chang, A. Dai, J. Uszkoreit, Q. Le, and S. Petrov. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*, 2019.

[27] G. Lai, Q. Xie, H. Liu, Y. Yang, and E. Hovy. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2017.

[28] Y. Levine, N. Wies, O. Sharir, H. Bata, and A. Shashua. The depth-to-width interplay in self-attention. *arXiv:2006.12467*, 2020.

[29] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, 2020.

[30] O. Lieber, O. Sharir, B. Lenz, and Y. Shoham. Jurassic-1: Technical details and evaluation. *White Paper. AI21 Labs*, 2021.

[31] S. Lin, J. Hilton, and O. Evans. TruthfulQA: Measuring how models mimic human falsehoods. *arXiv:2109.07958*, 2021.

[32] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[33] S. McCandlish, J. Kaplan, D. Amodei, and O. D. Team. An empirical model of large-batch training. *arXiv:1812.06162*, 2018.

[34] S. Merity, C. Xiong, J. Bradbury, and R. Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.

[35] M. Mitchell, S. Wu, A. Zaldivar, P. Barnes, L. Vasserman, B. Hutchinson, E. Spitzer, I. D. Raji, and T. Gebru. Model cards for model reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 2019.

[36] J. Nocedal. Updating Quasi-Newton Matrices with Limited Storage. *Mathematics of Computation*, 35(151):773–782, 1980.

[37] D. Paperno, G. Kruszewski, A. Lazaridou, Q. N. Pham, R. Bernardi, S. Pezzelle, M. Baroni, G. Boleda, and R. Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. *arXiv:1606.06031*, 2016.

[38] J. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, F. Song, J. Aslanides, S. Henderson, R. Ring, S. Young, E. Rutherford, T. Hennigan, J. Menick, A. Cassirer, R. Powell, G. van den Driessche, L. A. Hendricks, M. Rauh, P.-S. Huang, A. Glaese, J. Welbl, S. Dathathri, S. Huang, J. Uesato, J. Mellor, I. Higgins, A. Creswell, N. McAleese, A. Wu, E. Elsen, S. Jayakumar, E. Buchatskaya, D. Budden, E. Sutherland, K. Simonyan, M. Paganini, L. Sifre, L. Martens, X. L. Li, A. Kuncoro, A. Nematzadeh, E. Gribovskaya, D. Donato, A. Lazaridou, A. Mensch, J.-B. Lespiau, M. Tsimpoukelli, N. Grigorev, D. Fritz, T. Sottiaux, M. Pajarskas, T. Pohlen, Z. Gong, D. Toyama, C. de Masson d'Autume, Y. Li, T. Terzi, I. Babuschkin, A. Clark, D. de Las Casas, A. Guy, J. Bradbury, M. Johnson, L. Weidinger, I. Gabriel, W. Isaac, E. Lockhart, S. Osindero, L. Rimell, C. Dyer, O. Vinyals, K. Ayoub, J. Stanway, L. Bennett, D. Hassabis, K. Kavukcuoglu, and G. Irving. Scaling language models: Methods, analysis & insights from training Gopher. *arXiv:2112.11446*, 2021.

[39] J. W. Rae, A. Potapenko, S. M. Jayakumar, T. P. Lillicrap, K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, et al. Compressive transformers for long-range sequence modelling. *Advances in Neural Information Processing Systems*, 2020.

[40] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[41] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He. Zero: Memory optimizations toward training trillion parameter models. In *International Conference for High Performance Computing, Networking, Storage and Analysis*, 2020.

[42] H. Robbins and S. Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400–407, Sept. 1951.

[43] R. Rudinger, J. Naradowsky, B. Leonard, and B. Van Durme. Gender bias in coreference resolution. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2018.

[44] K. Sakaguchi, R. Le Bras, C. Bhagavatula, and Y. Choi. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[45] M. Sap, H. Rashkin, D. Chen, R. LeBras, and Y. Choi. SocialIQA: Commonsense reasoning about social interactions. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2019.

[46] C. J. Shallue, J. Lee, J. Antognini, J. Sohl-Dickstein, R. Frostig, and G. E. Dahl. Measuring the effects of data parallelism on neural network training. *arXiv:1811.03600*, 2018.

[47] J. W. Siegel and J. Xu. Approximation rates for neural networks with general activation functions. *Neural Networks*, 128:313–321, 2020.

[48] S. Smith, M. Patwary, B. Norick, P. LeGresley, S. Rajbhandari, J. Casper, Z. Liu, S. Prabhumoye, G. Zerveas, V. Korthikanti, E. Zhang, R. Child, R. Y. Aminabadi, J. Bernauer, X. Song, M. Shoeybi, Y. He, M. Houston, S. Tiwary, and B. Catanzaro. Using Deepspeed and Megatron to Train Megatron-turing NLG 530b, A Large-Scale Generative Language Model. *arXiv:2201.11990*, 2022.

[49] A. Srivastava et al. Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models. *arXiv:2206.04615*, 2022.

[50] J. Steinhardt. Updates and lessons from AI forecasting, 2021. URL https://bounded-regret.ghost.io/ai-forecasting/.

[51] Y. Tay, M. Dehghani, J. Rao, W. Fedus, S. Abnar, H. W. Chung, S. Narang, D. Yogatama, A. Vaswani, and D. Metzler. Scale efficiently: Insights from pre-training and fine-tuning transformers. *arXiv:2109.10686*, 2021.

[52] R. Thoppilan, D. D. Freitas, J. Hall, N. Shazeer, A. Kulshreshtha, H.-T. Cheng, A. Jin, T. Bos, L. Baker, Y. Du, Y. Li, H. Lee, H. S. Zheng, A. Ghafouri, M. Menegali, Y. Huang, M. Krikun, D. Lepikhin, J. Qin, D. Chen, Y. Xu, Z. Chen, A. Roberts, M. Bosma, Y. Zhou, C.-C. Chang, I. Krivokon, W. Rusch, M. Pickett, K. Meier-Hellstern, M. R. Morris, T. Doshi, R. D. Santos, T. Duke, J. Soraker, B. Zevenbergen, V. Prabhakaran, M. Diaz, B. Hutchinson, K. Olson, A. Molina, E. Hoffman-John, J. Lee, L. Aroyo, R. Rajakumar, A. Butryna, M. Lamm, V. Kuzmina, J. Fenton, A. Cohen, R. Bernstein, R. Kurzweil, B. Aguera-Arcas, C. Cui, M. Croak, E. Chi, and Q. Le. LaMDA: Language models for dialog applications. *arXiv:2201.08239*, 2022.

[53] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

[54] L. Weidinger, J. Mellor, M. Rauh, C. Griffin, J. Uesato, P.-S. Huang, M. Cheng, M. Glaese, B. Balle, A. Kasirzadeh, Z. Kenton, S. Brown, W. Hawkins, T. Stepleton, C. Biles, A. Birhane, J. Haas, L. Rimell, L. A. Hendricks, W. Isaac, S. Legassick, G. Irving, and I. Gabriel. Ethical and social risks of harm from language models. *arXiv:arXiv:2112.04359*, 2021.

[55] J. Welbl, A. Glaese, J. Uesato, S. Dathathri, J. Mellor, L. A. Hendricks, K. Anderson, P. Kohli, B. Coppin, and P.-S. Huang. Challenges in detoxifying language models. In *Findings of the Association for Computational Linguistics: EMNLP*, 2021.

[56] A. Xu, E. Pathak, E. Wallace, S. Gururangan, M. Sap, and D. Klein. Detoxifying language models risks marginalizing minority voices. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021.

[57] G. Yang, E. J. Hu, I. Babuschkin, S. Sidor, X. Liu, D. Farhi, N. Ryder, J. Pachocki, W. Chen, and J. Gao. Tuning large neural networks via zero-shot hyperparameter transfer. In *Advances in Neural Information Processing Systems*, 2021.

[58] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2019.

[59] G. Zhang, L. Li, Z. Nado, J. Martens, S. Sachdeva, G. Dahl, C. Shallue, and R. B. Grosse. Which algorithmic choices matter at which batch sizes? insights from a noisy quadratic model. In *Advances in Neural Information Processing Systems*, 2019.

[60] B. Zoph, I. Bello, S. Kumar, N. Du, Y. Huang, J. Dean, N. Shazeer, and W. Fedus. Designing effective sparse expert models. *arXiv:2202.08906*, 2022.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] The claims in the abstract describe the work clearly.
   (b) Did you describe the limitations of your work? [Yes] We address limitations of our work.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] We have a discussion both in a model card and in Appendix I.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A] Results are not theoretical.
   (b) Did you include complete proofs of all theoretical results? [N/A] Results are not theoretical/no proofs.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] The code and the data are proprietary. However, for the scaling methodology we provide clear instructions on how to reproduce the results.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We provided all training details and hyperparameters.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] Due to the cost of training large models, we do not have multiple runs. However, the clear & predictable trends suggest the noise is very small.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] All experiments are ran on TPUv3/TPUv4 and this is stated in the text.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We cite the preexisting work for all data/models that we use.
   (b) Did you mention the license of the assets? [Yes] We use the same data as in Rae et al. [38] which uses a proprietary dataset. We also show results with an open source dataset– C4 **?**].
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A] We do not introduce new assets.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] We include a model card which includes this information.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] There was no human subjects or crowd sourcing in this work.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] There was no human subjects or crowd sourcing in this work.
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] There was no human subjects or crowd sourcing in this work.