Beyond Chinchilla-Optimal: Accounting for Inference in Language Model Scaling Laws

Nikhil Sardana*
MosaicML
nikhil@mosaicml.com

Jonathan Frankle
MosaicML
jonathan@mosaicml.com

Abstract

Large language model (LLM) scaling laws are empirical formulas that estimate changes in model quality as a result of increasing parameter count and training data. However, these formulas, including the popular DeepMind Chinchilla scaling laws, neglect to include the cost of inference. We modify the Chinchilla scaling laws to calculate the optimal LLM parameter count and pre-training data size to train and deploy a model of a given quality and inference demand. We conduct our analysis both in terms of a compute budget and real-world costs and find that LLM researchers expecting reasonably large inference demand (~1B requests) should train models smaller and longer than Chinchilla-optimal.

1 Introduction

Large language models (LLM) have substantial training and inference compute and energy costs [6, 12]. Training computation costs are primarily determined by the size of the model and the amount of data it sees during training [4]. State-of-the-art models have tens of billions of parameters and are trained on trillions of tokens [17]. Inference costs depend on the size of the model and the volume of user queries over the lifetime of the model. This volume can be significant: Demand for popular models can exceed billions of tokens per day [11, 15].

Accounting for both *training and inference*, how does one minimize the cost required to produce and serve a high quality model?

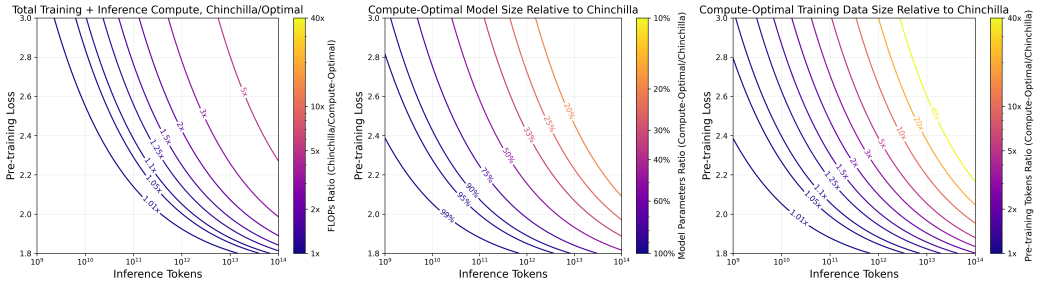
Significant prior research has been conducted to find scaling laws, empirical formulas that estimate how changes in model and training data size impact model quality [5, 4]. Hoffmann et al. [4] is perhaps the most influential of these works; finding that to scale language models most efficiently, parameters and tokens should grow approximately equally. The authors applied this scaling law to train a 70B parameter model, *Chinchilla*, that outperformed much larger, more expensive models, including GPT-3. Subsequent LLMs have been trained following the Chinchilla scaling laws [2, 9].

However, the Chinchilla scaling laws only account for the computational costs of training. By contrast, the LLaMA and LLaMA-2 family of models were trained on 1-2 trillion tokens, far more data than the Chinchilla scaling laws would deem "optimal" [16, 17]. Since inference costs are lower for smaller models, the extra training compute required to train a LLaMA-style model over a Chinchilla-style model of equivalent quality pays off after enough inference requests.

Prior work has discussed the training-inference compute trade-off [16, 17, 18, 1, 19]. Touvron et al. [16] cites the lower inference cost of smaller models as inspiration for the LLaMA series. De Vries [1] calculates the compute overhead of training longer than Chinchilla, but does not discuss quantify

*Corresponding author.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).



(a) (b) (c)

Figure 1: Ratios of (a) total FLOPs, (b) model parameters, and (c) pre-training tokens, for optimal models estimated via our method vs. Chinchilla-style models. For each point (x, y) in the figures, we compute the Chinchilla model parameter count and training data required to reach the loss y, and the number of combined FLOPs required to train and run inference for x tokens using the Chinchilla model. Then, we compute the same values (total FLOPs, parameter count, training data size) for the compute-optimal models returned by our method, and plot the ratios.

compute savings from inference. Recently, Villalobos and Atkinson [19] discusses this trade-off in more detail, but shows the shift in scaling laws for only a single particular number of inferences.

In this paper, we modify Chinchilla scaling laws to account for inference costs, calculating the optimal parameter and training token counts—both in terms of compute and dollar costs—to train and deploy a model of any given quality and inference demand. Our principled derivation estimates that LLM practitioners expecting significant demand (~10^9 inference requests) should train models substantially smaller and longer than Chinchilla-optimal.

2 Computational Optimality

We seek to minimize the computational costs of a model of a given quality and inference demand. We closely follow the methodology in Hoffmann et al. [4] (henceforth referred to as "the Chinchilla paper"), using pre-training cross-entropy loss as a proxy for quality, and floating-point operations (FLOPs) as our unit of computational cost.

We model our pre-training loss $L(N, D_{\text{tr}})$ in terms of the number of parameters, N, and pre-training tokens, D_{tr}, according to the Chinchilla paper's third scaling law:

$$L(N, D_{\text{tr}}) \triangleq E + \frac{A}{N^{\alpha}} + \frac{B}{D_{\text{tr}}^{\beta}} \tag{1}$$

The Chinchilla paper derived the parametric loss function in Eq. 1 and fit values for A, B, E, α, and β from the authors' empirical training results. The best-fit values for these constants depend on the exact dataset and model architecture; however, the Chinchilla paper found largely consistent results across the MassiveText, Github [13], and C4 [14] datasets, and subsequent work has replicated these scaling laws on other internet corpora and transformer variants [2]. Thus, we use the constant values from the Chinchilla paper in our analysis.

Additionally, we assume that conditioned on pre-training loss, inference demand is independent of model size and token count. In other words, models of equivalent quality but different parameter counts will see the same requests.[2]

Let $\text{T}_{\text{FLOPs}}(N, D)$ and $\text{I}_{\text{FLOPs}}(N, D)$ be the number of FLOPs required to train and run inference, respectively, on a model with N parameters for D tokens. Denote the number of tokens (input + output) of a single inference request i as $D_{\text{inf}}^{(i)}$. Let $D_{\text{inf}} = \sum_i D_{\text{inf}}^{(i)}$ be the sum of all tokens over all inference requests.

[2]In practice, smaller models of equivalent quality may have greater demand since they can have lower inference latency.

Formally, we are interested in minimizing the sum of our training and inference FLOPs under the constraint $L(N, D_{\text{tr}}) = \ell$:

$$N^*(\ell, D_{\text{inf}}), D^*_{\text{tr}}(\ell, D_{\text{inf}}) = \underset{N, D_{\text{tr}} | L(N, D_{\text{tr}}) = \ell}{\arg\min} \mathrm{T}_{\text{FLOPs}}(N, D_{\text{tr}}) + \sum_i \mathrm{I}_{\text{FLOPs}}(N, D^{(i)}_{\text{inf}}). \tag{2}$$

N^* and D^*_{tr} are functions that describe the optimal parameters and pre-training tokens, respectively, that minimize total training and inference compute. The pre-training loss constraint ensures that we minimize compute for a given quality.

We use the standard approximation of FLOPs for transformer models with N parameters: $6N$ per training token and $2N$ per inference token [5]. Thus, our objective simplifies to:

$$N^*(\ell, D_{\text{inf}}), D^*_{\text{tr}}(\ell, D_{\text{inf}}) = \underset{N, D_{\text{tr}} | L(N, D_{\text{tr}}) = \ell}{\arg\min} 6ND_{\text{tr}} + 2ND_{\text{inf}}. \tag{3}$$

We note that this is the "converse" of the Chinchilla optimization problem. In the Chinchilla paper, the authors assumed a *fixed* compute budget and found N^* and D^*_{tr} that *minimized* pre-training loss. Our objective is to *fix* pre-training loss and find N^* and D^*_{tr} that *minimize* compute costs.

Crucially, our total computational cost depends on the inference demand over the lifetime of the model, but our model's parameter count and data size are determined prior to training. Thus, our analysis is predicated on the assumption that LLM practitioners can estimate their inference demand prior to training.

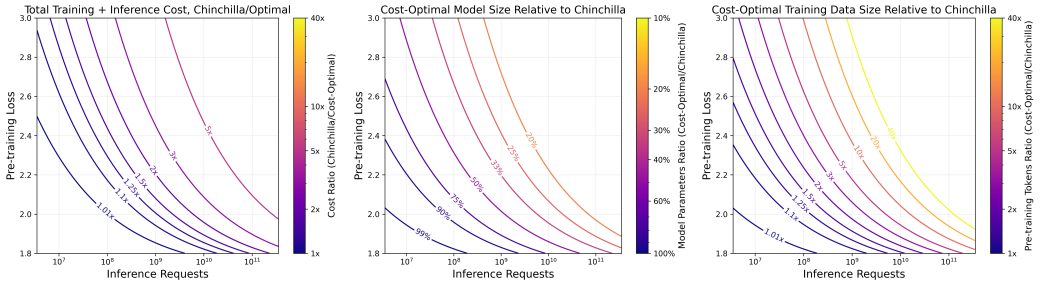
Without inference ($D_{\text{inf}} = 0$), the optimization problem in Eq. 3 can be solved analytically. Unfortunately, accounting for inference ($D_{\text{inf}} > 0$), determining N^* and D^*_{tr} analytically as functions of ℓ and D_{inf} is intractable (we defer our proof to Appendix A). Instead, we computationally solve for N^* and D^*_{tr} across a range of values of ℓ and D_{inf} using the Newton root-finding method. In practice, this method converges for relevant inputs and we are able to determine optimal parameter/token counts.

In Figure 1, we show how our inference-adjusted model's FLOP counts, parameters, and pre-training tokens compare to Chinchilla-style models across a range of loss values and inference demands. When inference usage is significantly less than the number of pre-training tokens, Chinchilla models are essentially compute-optimal. However, as demand increases, inference costs becomes a significant factor. For a Chinchilla-7B-quality model with an inference demand of 10^{11} tokens, our formula suggests the compute-optimal method is to train a 6B parameter model on $1.18\times$ the original data. For higher quality (i.e. larger and longer) models, the volume of inference demand required to shift the scaling law increases: An LLM developer that expects a 30B-Chinchilla-quality model will see 10^{13} tokens during inference can reduce their total FLOPs by 28% by training a 13.6B model on $2.84\times$ the data. We provide additional results in Sec. B.1 in the Appendix.

3 Estimating Real-World Cost Optimality

Optimizing purely for minimum FLOPs has significant drawbacks which limit the applicability of our analysis in Section 2 to real-world deployments. The real-world cost of an inference request of $3D$ tokens is generally different than the cost to train on D tokens. For instance, inference hardware utilization can be much lower than training utilization, since small batch size computation can result in low Model FLOPs Utilization (MFU). MFU can be as low as ~1% for inference [12] but is typically 40-60% during training [7]. Utilization is also different for input tokens vs. output tokens — since input tokens (prompts) are typically processed in a single forward pass, utilization is typically near training levels. By contrast, during generation, output tokens must be produced sequentially, resulting in low utilization due to memory bandwidth constraints. Another complicating factor is that inference operations can sometimes be cheaper than training FLOPs, since models can be quantized before inference time, turning 16- or 32-bit floating-point operations into 4- or 8-bit integer operations which run more efficiently on the same hardware. Quantization can also enable LLMs to fit on GPUs with less VRAM, so training and inference may occur on different hardware altogether [3].

To estimate the real-world cost of inference, we modify Eq. 2 to account for hardware utilization: MFU_{tr}, $\mathrm{MFU}_{\text{inp}}$, and $\mathrm{MFU}_{\text{out}}$ are our training, inference input, and inference output MFUs, respectively. In addition, we add parameters for training and inference cost per FLOP, C_{tr} and C_{inf}. Our



(a) (b) (c)

Figure 2: Ratios of (a) total cost, (b) model parameters, and (c) pre-training tokens, for cost-optimal models via our real-world estimation method vs. Chinchilla-style models. Results in this figure are shown with the following settings: training with 50% MFU, inference input with 50% MFU, generation with 1% MFU. Inference requests have 70 input tokens and 215 output tokens each, aligning with averages from real-world data [21]. To mimic a realistic scenario, we calculate costs assuming training occurs on A100-80GB and inference occurs on A100-40GB accelerators after INT8 quantization (see Sec. B.3 for details).

new objective is:

$$N^*(\ell, D_{\text{inp}}, D_{\text{out}})^*(\ell, D_{\text{inp}}, D_{\text{out}}) = \underset{N, D \mid L(N, D_{\text{tr}}) = \ell}{\arg\min} \left[\frac{C_{\text{tr}}}{\text{MFU}_{\text{tr}}} \, \text{T}_{\text{FLOPs}}(N, D_{\text{tr}}) \right. \tag{4}$$

$$\left. + \sum_i \frac{C_{\text{inf}}}{\text{MFU}_{\text{inp}}} \, \text{I}_{\text{FLOPs}}(N, D_{\text{inp}}^{(i)}) + \sum_i \frac{C_{\text{inf}}}{\text{MFU}_{\text{out}}} \, \text{I}_{\text{FLOPs}}(N, D_{\text{out}}^{(i)}) \right]. \tag{5}$$

We again use the approximations for FLOPs for transformer models, reducing the above equation to:

$$N^*(\ell, D_{\text{inp}}, D_{\text{out}}), D_{\text{tr}}^*(\ell, D_{\text{inp}}, D_{\text{out}}) = \underset{N, D_{\text{tr}} \mid L(N, D_{\text{tr}}) = \ell}{\arg\min} \frac{6 N D_{\text{tr}} C_{\text{tr}}}{\text{MFU}_{\text{tr}}} + 2 N C_{\text{inf}} \left[\frac{D_{\text{inp}}}{\text{MFU}_{\text{inp}}} + \frac{D_{\text{out}}}{\text{MFU}_{\text{out}}} \right] \tag{6}$$

Eq. 6 is a simplified model of real-world costs: we leave aside latency requirements and assume MFU and cost per FLOP do not depend on model size, configuration, or sequence length. Still, our approximation is flexible enough to account for heterogeneous hardware utilization and costs.

In Figure 2, we show how inference-adjusted cost-optimal models compare to Chinchilla-style models, assuming typical training and inference hardware costs and MFU. For a 30B-Chinchilla-quality model, LLM practitioners expecting 1.5B inference requests can reduce costs by 17% by instead training a 16B model on 3.35T tokens. In Sec. B.2, we show further results for various configurations.

Comparing our compute-optimal analysis in Fig. 1 to our real-world cost analysis in Fig. 2, we see that for the same inference demand of 2T tokens (7.02B requests), a Chinchilla-70B model requires only **1.3%** extra FLOPs compared to an equal-quality *compute-optimal* model, but costs **36%** more than a *cost-optimal* model. This difference is attributable to the 50× lower MFU of each inference output token compared to training, which our FLOP-based analysis in Sec. 2 fails to capture.

4 Conclusion

In this work, we modify the Chinchilla scaling laws to account for both the computational and real-world costs of inference. As inference demand approaches pre-training data size, the additional cost pushes the optimal parameters-to-tokens ratio towards smaller and longer-trained models.

We make strong assumptions about the Chinchilla scaling laws and our analysis only applies insofar as these laws hold true. Further work is needed to experimentally validate our formulas and determine if scaling laws apply in the extreme ranges, where pre-training tokens exceed model parameters by orders of magnitudes.

Acknowledgements

We thank Sasha Doubov for helpful discussions and Daya Khudia, Mihir Patel, and Linden Li for their feedback on the manuscript.

References

[1] H. De Vries. Go smol or go home, 2023. URL https://www.harmdevries.com/post/model-size-vs-compute-overhead/.

[2] N. Dey, G. Gosal, Zhiming, Chen, H. Khachane, W. Marshall, R. Pathria, M. Tom, and J. Hestness. Cerebras-gpt: Open compute-optimal language models trained on the cerebras wafer-scale cluster, 2023.

[3] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers, 2023.

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

[5] 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, 2020.

[6] W. Knight. Openai's ceo says the age of giant ai models is already over. *Wired*. ISSN 1059-1028. URL https://www.wired.com/story/openai-ceo-sam-altman-the-age-of-giant-ai-models-is-already-over/.

[7] V. Korthikanti, J. Casper, S. Lym, L. McAfee, M. Andersch, M. Shoeybi, and B. Catanzaro. Reducing activation recomputation in large transformer models, 2022.

[8] L. Labs. Gpu cloud - vms for deep learning. https://lambdalabs.com/service/gpu-cloud.

[9] N. Muennighoff, A. M. Rush, B. Barak, T. L. Scao, A. Piktus, N. Tazi, S. Pyysalo, T. Wolf, and C. Raffel. Scaling data-constrained language models, 2023.

[10] NVIDIA. Nvidia a100 datasheet, 2021. URL https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet-us-nvidia-1758950-r4-web.pdf.

[11] OpenAI and A. Pilipiszyn. Gpt-3 powers the next generation of apps, Mar 2021. URL https://openai.com/blog/gpt-3-apps.

[12] R. Pope, S. Douglas, A. Chowdhery, J. Devlin, J. Bradbury, A. Levskaya, J. Heek, K. Xiao, S. Agrawal, and J. Dean. Efficiently scaling transformer inference, 2022.

[13] J. W. 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, V. Mikulik, I. Babuschkin, A. Clark, D. de Las Casas, A. Guy, C. Jones, J. Bradbury, M. Johnson, B. Hechtman, 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, 2022.

[14] 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, 2023.

[15] N. Shazeer and D. d. Freitas. Introducing character, Dec 2022. URL https://blog.character.ai/introducing-character/.

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

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

[18] J. Tow, D. M. Marco Bellagente, and C. R. Ruiz. Technical report for stablelm-3b-4e1t. https://stability.wandb.io/stability-llm/stable-lm/reports/StableLM-3B-4E1T--VmlldzoyMjU4?accessToken=u3zujipenkx5g7rtcj9qojjgxpconyjktjkli2po09nffrffdhhchq045vp0wyfo, 2023. Accessed 02-10-2023.

[19] P. Villalobos and D. Atkinson. Trading off compute in training and inference, 2023. URL https://epochai.org/blog/trading-off-compute-in-training-and-inference. Accessed: 2023-9-26.

[20] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han. Smoothquant: Accurate and efficient post-training quantization for large language models, 2023.

[21] L. Zheng, W.-L. Chiang, Y. Sheng, T. Li, S. Zhuang, Z. Wu, Y. Zhuang, Z. Li, Z. Lin, E. P. Xing, J. E. Gonzalez, I. Stoica, and H. Zhang. Lmsys-chat-1m: A large-scale real-world llm conversation dataset, 2023.

A No Analytic Solution for Inference-Compute Optimality

In this section, we prove that it is not possible to analytically derive the optimal model size and pre-training token count according to the third Chinchilla law, after accounting for the computational cost of inference. Conditioned on model quality, we assume that inference demand does not depend on model size and can be estimated prior to training.

Theorem A.1. *Given a fixed model quality and inference demand, there exists no analytic solution for the compute-optimal model size and pre-training token count according to the third Chinchilla law, after accounting for the computational cost of inference.*

Proof. By Eq. 3, the overall compute cost in FLOPs for training a model with N parameters on D_{tr} tokens and running inference on D_{inf} tokens is given by $C(N, D_{\text{tr}}, D_{\text{inf}}) = 6ND_{\text{tr}} + 2ND_{\text{inf}}$.

We seek the minimum overall compute budget to train and deploy a model of a given quality and inference demand. Formally, we optimize the objective:

$$\min C(N, D_{\text{tr}}, D_{\text{inf}}) \tag{7}$$

subject to the constraint $L(N, D_{\text{tr}}) = E + \frac{A}{N^\alpha} + \frac{B}{D_{\text{tr}}^\beta} = \ell$.

This constraint, from the third Chinchilla law, ensures we are minimizing compute while fixing model quality (pre-training loss). $A = 406.4, B = 410.7, E = 1.69, \alpha = 0.336$, and $\beta = 0.283$ are constants determined empirically by Hoffmann et al. [4].[3]

We solve this optimization problem via the method of Lagrange multipliers. The gradients are:

$$\nabla C(N, D_{\text{tr}}) = (6D + 2D_{\text{inf}})\hat{i} + 6N\hat{j} \tag{8}$$

$$\nabla L(N, D_{\text{tr}}) = -\alpha A N^{-\alpha-1}\hat{i} - \beta B D^{-\beta-1}\hat{j} \tag{9}$$

We have three equations and three variables ($D_{\text{tr}}, N, \lambda$), where λ is our Lagrange multiplier:

$$6D_{\text{tr}} + 2D_{\text{inf}} = -\lambda\alpha A N^{-\alpha-1} \qquad 6N = -\lambda\beta B D_{\text{tr}}^{-\beta-1} \qquad E + \frac{A}{N^\alpha} + \frac{B}{D_{\text{tr}}^\beta} = \ell \tag{10}$$

With some algebraic manipulation, we can eliminate λ and write $\frac{A}{N^\alpha}$ in terms of D_{tr}:

$$\frac{A}{N^\alpha} = \frac{3\beta B D_{\text{tr}}^{-\beta} + D_{\text{inf}}\beta B D_{\text{tr}}^{-\beta-1}}{3\alpha} \tag{11}$$

We are left to solve the following equation for D_{tr}:

$$0 = (E - \ell) + \Big[\frac{\beta B}{\alpha} + B\Big] D_{\text{tr}}^{-\beta} + \frac{D_{\text{inf}}\beta B}{3\alpha} D_{\text{tr}}^{-\beta-1} \tag{12}$$

Thus, determining D_{tr} as a function of D_{inf} and ℓ involves finding the roots of equations of the form $ax^{-1.283} + 756.6x^{-0.283} + c = 0$ for arbitrary a and $c > 0$, which is not possible in general. □

B Further Results

B.1 Compute-Optimal Results

We present further results from our analysis in Sec. 2. In Table 1, we show the computational cost (in FLOPs) to train and run inference for Chinchilla-style models of various sizes and inference demands. We then calculate the *compute-optimal* model configuration to reach the same quality (equal loss) and run inference, and note the overall compute reduction.

[3]The Chinchilla paper reports $\alpha = 0.34$ and $\beta = 0.28$. However, these are rounded values; to better fit the results reported in Table A.3 of Hoffmann et al. [4], we use $\alpha = 0.336$ and $\beta = 0.283$, as in De Vries [1].

Table 1: Compute-Optimal vs. Chinchilla-style Models for Selected Configurations.

		Chinchilla Model			Compute-Optimal Model			
Inference Tokens	Train Loss	Params	Training Tokens	FLOPs	Params	Training Tokens	FLOPs	FLOP Reduction
50B	2.53	1B	27.4B	2.64e20	6.33M	46.8B	2.41e20	9.1%
200B	2.13	7B	276B	1.44e22	5.4B	367B	1.40e22	2.6%
1T	2.05	13B	577B	7.10e22	8.32B	967B	6.49e22	8.5%
5T	1.96	30B	1.56T	5.80e23	16.4B	3.27T	4.86e23	16%
10T	1.89	70B	4.26T	3.19e24	41.6B	7.92T	2.81e24	12%

Table 2: Cost-Optimal vs. Chinchilla-style Models for Selected Configurations.

		Chinchilla Model			Cost-Optimal Model			
Inference Requests	Train Loss	Params	Training Tokens	Total Cost ($)	Params	Training Tokens	Total Cost ($)	Cost Savings
175M	2.53	1B	27.4B	3.77K	327M	152B	1.89K	50%
702M	2.13	7B	276B	124K	2.90B	929B	81.8K	34%
3.51B	2.05	13B	577B	987K	430B	3.1T	500K	49%
17.5B	1.96	30B	1.56T	10.8M	8.58B	12.1T	4.52M	58%
35.1B	1.89	70B	4.26T	51.5M	21.5B	27T	23.8M	54%

B.2 Cost-Optimal Results

We show additional results from our cost-optimality analysis in Sec. 3. In Table 2, we show the total training + inference costs for Chinchilla models of various sizes at different levels of inference demands. We then calculate costs for equivalent-quality (i.e. same pre-training loss) *cost-optimal* models and show the overall savings. We use the same settings from Figure 2, designed to mimic a typical real-world deployment: training and inference input at 50% MFU, generation at 1% MFU [7, 12]. Each inference request has 70 input tokens and 215 output tokens, in accordance with averages from the LMSYS-Chat dataset of 1M inference requests from Zheng et al. [21]. Costs are calculated assuming training and inference on A100-80GB and A100-40GB accelerators, respectively. We further assume the model is quantized to INT8 prior to inference, which is commonly done with no quality reduction [20]. All costs are reported in US dollars.

B.3 GPU Details

GPU pricing varies based on vendor and fluctuates over time. At the time of writing, an A100-40GB costs USD $1.10/hr and an A100-80GB costs $1.50/hr on Lambda Labs [8]. We use these values in our cost analysis in Sec. 3 and in Table 2. Both variants have a peak performance of 3.12×10^{14} dense FP16/BF16 operations and 6.24×10^{14} INT8 operations per second [10].