CS 336: Assignment 3

Brandon Snider

May 06, 2025

Contents

2 Scaling Laws Review

Problem (`chinchilla_isoflops`): 5 points

a) See `cs336_scaling/chinchilla_isoflops.py`

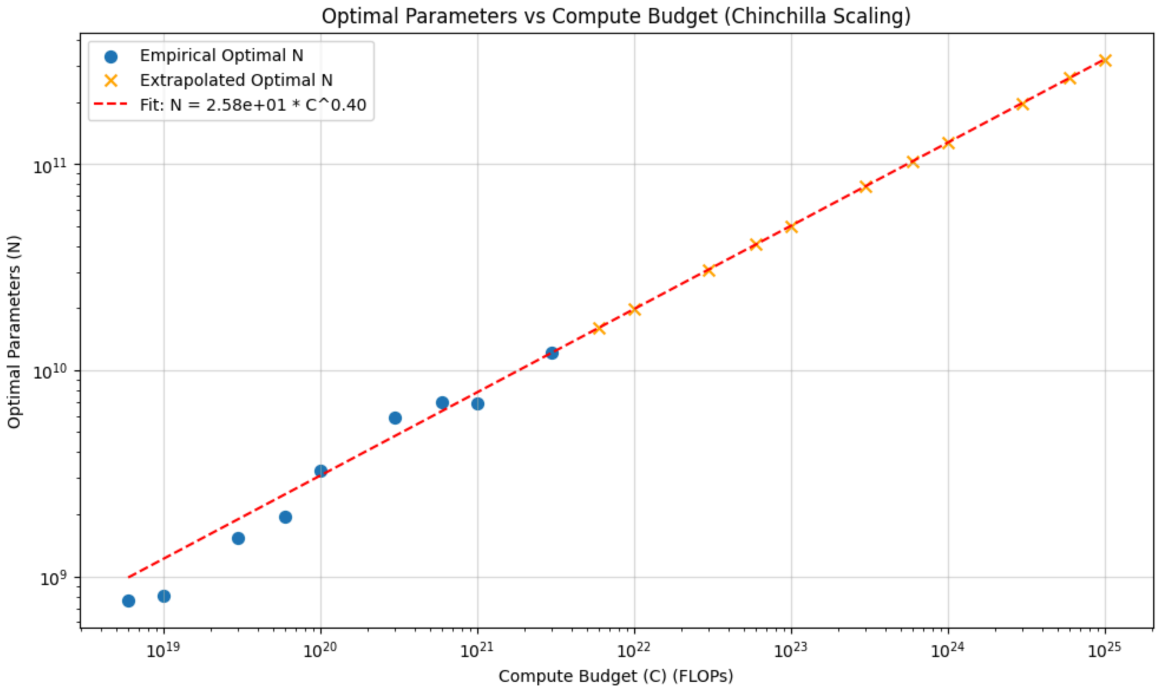


Figure 1: Chinchilla-optimal compute budget vs. parameter count

Empirical $\langle C_i, N_{\text{opt}(C_i)} \rangle$ points obtained:

Compute (C)	Optimal Params (N)	Human-readable	Scientific
6e+18	762,093,419	762M	7.62e+08
1e+19	806,647,749	806M	8.07e+08
3e+19	1,536,852,354	1.54B	1.54e+09
6e+19	1,952,041,776	1.95B	1.95e+09
1e+20	3,253,402,960	3.25B	3.25e+09
3e+20	5,903,836,027	5.90B	5.90e+09
6e+20	6,971,055,968	6.97B	6.97e+09
1e+21	6,859,328,563	6.86B	6.86e+09
3e+21	12,148,905,329	12.15B	1.21e+10

Table 1: Empirically optimal parameter count for various compute budgets

Predicted optimal model sizes:

$10^{23} = \text{1e23 FLOPs} \to 50,022,254,912 \ (50B \text{ or } 5\text{e}10)$ parameters
$10^{24} = \text{1e24 FLOPs} \to 126,757,785,319 \ (126B \text{ or } 1.27\text{e}11)$ parameters

b)

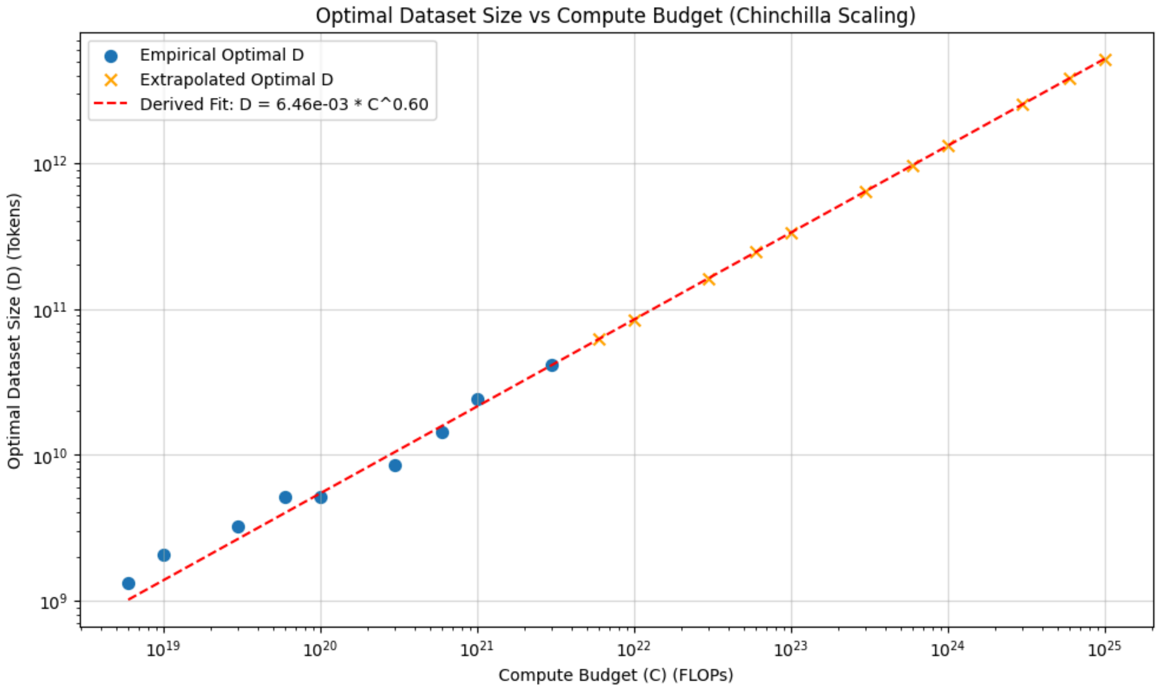


Figure 2: Predicted optimal dataset size for various compute budgets

Empirical $\langle C_i, D_{\mathrm{opt}(C_i)} \rangle$ points obtained:

Compute (C)	Optimal Tokens (D)	Human-readable	Scientific
6e+18	1,312,175,089	1.31B	1.31e+09
1e+19	2,066,164,157	2.07B	2.07e+09
3e+19	3,253,402,961	3.25B	3.25e+09
6e+19	5,122,841,182	5.12B	5.12e+09
1e+20	5,122,841,182	5.12B	5.12e+09
3e+20	8,469,069,901	8.47B	8.47e+09
6e+20	14,345,028,996	14.35B	1.43e+10
1e+21	24,297,810,658	24.30B	2.43e+10
3e+21	41,155,971,378	41.16B	4.12e+10

Table 2: Empirically optimal total token count for various compute budgets

Predicted optimal token dataset sizes:

$10^{23} = 1\mathrm{e}23\ \mathrm{FLOPs} \to 333,185,033,259\ (333B \text{ or } 3.33\mathrm{e}11)$ tokens

$10^{24} = 1\mathrm{e}24\ \mathrm{FLOPs} \to 1,314,843,630,676\ (1.31T \text{ or } 1.31\mathrm{e}12)$ tokens

3 Constructing Scaling Laws

Problem (`scaling_laws`): 50 points

High-Level Approach

- Run small-scale experiments (below `3e16` FLOPs) to learn about hyperparameter choices (aspect ratio, head dimension, learning rate, batch size) as cheaply as possible.
- Fix as many hyperparameters as possible (i.e. those that don't appear to affect loss at different scales) to maximize the probability of getting a clean scaling law at larger scales.
- Progressively scale up from `3e16` FLOPs to `3e17` FLOPs, fitting IsoFLOP profiles (approach 2 from J. Hoffmann *et al.* [1]) at each scale as follows:
  - Use prior experiments (and heuristics from J. Hoffmann *et al.* [1] and J. Kaplan *et al.* [2]) to guess at the optimal number of non-embedding parameters $N_{\text{guess}}(C_i)$.
  - Query training runs one at a time, as follows:
    - Start with a model that has $N_{\text{guess}}(C_i)$ non-embedding parameters.
    - Expand the search on either side (i.e. models with more or fewer non-embedding parameters) based on the results, until finding points on either side of the minimum.
  - Possibly refine the search by querying close to the initial minimum.
  - Fit a quadratic mapping log-non-embedding parameters to loss.
  - Find $N_{\text{opt}}(C_i)$ and $L_{\text{opt}}(C_i)$ (the loss achieved by a model with the optimal number of non-embedding parameters) as the minimum of that quadratic.
- Fit scaling laws using the minima of the quadratics at each scale from `3e16` to `3e17` (4 total points):
  - $N_{\text{opt}}(C) = a \cdot C^b$
  - $L_{\text{opt}}(C) = 1.69 + c \cdot C^d$
    - A floored power law, where the floor is intended to represent the entropy of the training data, and the value 1.69 comes from J. Hoffmann *et al.* [1]. The actual entropy of SlimPajama is likely not identical to the entropy of the Chinchilla dataset. Per J. Kaplan *et al.* [2], though, the range of possible values is narrow, and the impact on the losses predicted by the scaling law is minimal.
- Plug $C = 1 \cdot 10^{19}$ into the scaling laws to predict the optimal number of non-embedding parameters and the associated loss.
- Sweep over model configurations to recover the d_{model}, n_{layers}, and n_{heads} values that most closely approximate $N_{\text{opt}}(1 \cdot 10^{19})$ non-embedding parameters, while adhering to whatever aspect ratio and head dimension constraints were pointed to by the small-scale experiments.

Small-scale Experiments (`3e16` and below)

Aspect ratio and head dimension

Grounded in the findings from J. Kaplan *et al.* [2] on the insignificance of model shape within fairly broad bounds, I intended to clamp aspect ratio and head dimension within a sensible range at each compute scale (e.g. $(d_{\text{model}}/n_{\text{layers}}) \in [32, 256]$ and $d_{\text{head}} \in [32, 128]$).

I instantiated that logic in a helper that would generate the candidate model configuration that best approximates a target non-embedding parameter count, within provided bounds.

Empirically, up to the `3e16` scale, this was producing noisy data (see appendix) that looked unlikely to be amenable to fitting a useful scaling law.

I decided to fix the aspect ratio and head dimension. For the small-scale experiments (`1e16` and below) at which an aspect ratio of 64 would yield a 1-layer model or a 1-head model, I used 32. For `3e16` and above, I used 64 for both — well within the bounds suggested by J. Kaplan *et al.* [2].

Unfortunately, the budget spent prior to discovering that the range-based approach didn't look promising necessitated some game-time adjustments to the final run plan (more on this later).

Batch size and learning rate

I started with a batch size of 128, and found that a learning rate of `1e-3` was optimal in every small-scale run.

I decided to provisionally fix learning rate to `1e-3` and batch size to 128 (pending any indication of divergence at larger scales), on the intuition that using a larger batch size would be unlikely to be optimal given that I could not push the learning rate any higher.

Fixed hyperparameters

Heading into the larger-scale experiments that I hoped to use to fit the scaling laws, I provisionally fixed the following hyperparameters for all subsequent runs:

- Aspect ratio: 64
- Head dimension: 64
- Batch size: 128
- Learning rate: `1e-3`

Budgeting

With batch size, learning rate, head dimension, and aspect ratio fixed, what remained was to sweep over models with varying non-embedding parameter counts at each of the scales that I intended to use to fit the scaling laws. Prior to the small-scale experiments, I had the following run plan:

- 5 runs at `3e16` FLOPs
- 4 runs at `6e16` FLOPs
- 4 runs at `1e17` FLOPs
- 3 runs at `3e17` FLOPs

Budget: `1.69e18` FLOPs

As mentioned, it took more budget than I'd hoped to develop confidence that I had a setup that might yield a clean scaling law. I used `0.6e18` FLOPs for the small-scale experiments, leaving `1.4e18` FLOPs for the larger-scale runs, and had to cut down to 2 runs at `3e17` FLOPs. More on how this affected the final scaling law fit below.

Larger-scale Experiments

To help determine which model configurations (d_{model}, n_{layers}, and n_{heads}) to query at each scale, I generated all model configurations that (i) would be accepted by the training API's (ii) used a provided aspect ratio and head dimension. I also computed the number of tokens per parameter that each model would be trained for at each compute scale, using $C = 6ND \Rightarrow D = C/(6N)$. The results with $\left(d_{\text{model}}/n_{\text{layers}}\right) = 64$ and $d_{\text{head}} = 64$ are shown below.

D	L	H	$N_{\text{non-embed}}$	N_{total}	$(D/N)_{3e16}$	$(D/N)_{6e16}$	$(D/N)_{1e17}$	$(D/N)_{3e17}$
128	2	2	393,216	8,585,216	67.8	135.7	226.1	678.4
192	3	3	1,327,104	13,615,104	27.0	53.9	89.9	269.7
256	4	4	3,145,728	19,529,728	13.1	26.2	43.7	131.1
320	5	5	6,144,000	26,624,000	7.1	14.1	23.5	70.5
384	6	6	10,616,832	35,192,832	4.0	8.1	13.5	40.4
448	7	7	16,859,136	45,531,136	2.4	4.8	8.0	24.1
512	8	8	25,165,824	57,933,824	1.5	3.0	5.0	14.9
576	9	9	35,831,808	72,695,808	0.9	1.9	3.2	9.5
640	10	10	49,152,000	90,112,000	0.6	1.2	2.1	6.2
704	11	11	65,421,312	110,477,312	0.4	0.8	1.4	4.1
768	12	12	84,934,656	134,086,656	0.3	0.6	0.9	2.8
832	13	13	107,986,944	161,234,944	0.2	0.4	0.6	1.9
896	14	14	134,873,088	192,217,088	0.1	0.3	0.5	1.4
960	15	15	165,888,000	227,328,000	0.1	0.2	0.3	1.0
1024	16	16	201,326,592	266,862,592	0.1	0.1	0.2	0.7

Table 3: Model configurations and token/parameter ratios at each compute scale

At each compute scale (`3e16`, `6e16`, `1e17`, and `3e17`), I used the results from the previous scale to guess at the optimal configuration. I ran queries one at a time, attempting to use as little compute as possible to find points on close to the minimum, and on both sides of th minimum. I think this can be said to have worked moderately well at best. The results are below.

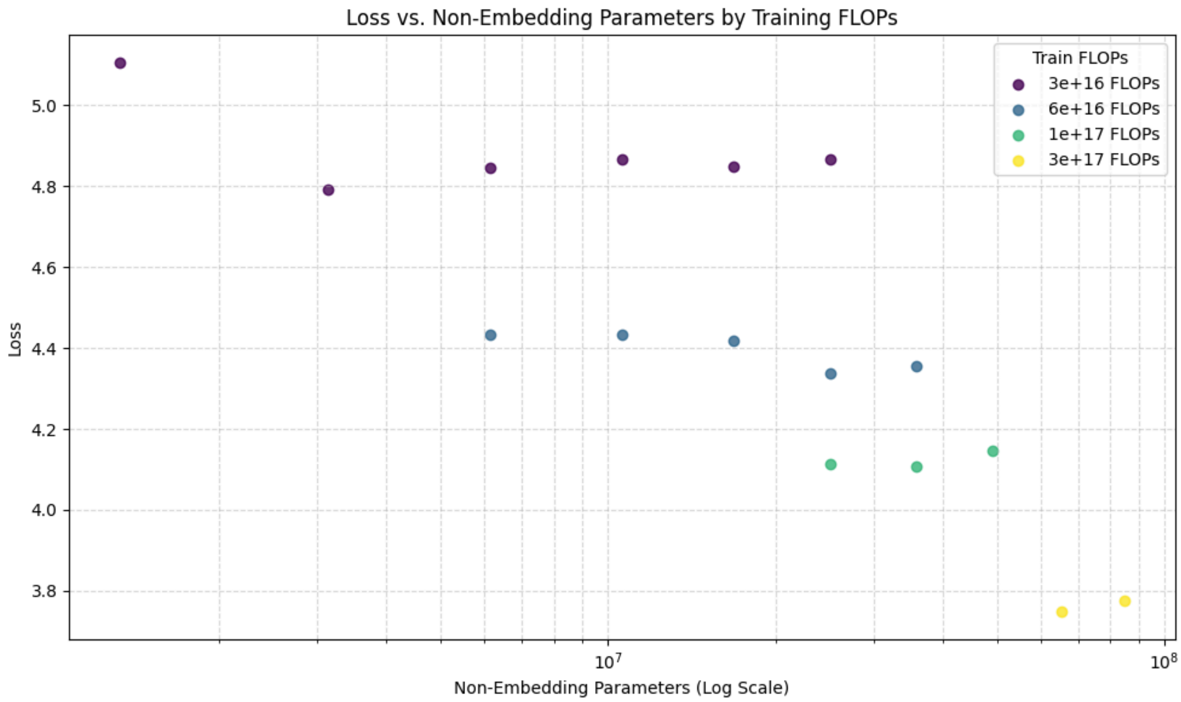


Figure 3: Non-embedding parameter count vs. loss at scales used to fit scaling laws

The point that produced the lowest loss at the `3e16` scale appeared anomalous, so I excluded it from the analysis. As mentioned, I also had to perform one fewer run than I would have liked at `3e17`, so didn't truly have enough data to fit a scaling law.

Based on the trend observed across `3e16` through `1e17`, the best-performing model saw more tokens per parameter as the compute budget increased. I guessed that the optimal configuration at `3e17` would be one size smaller (in the table above) than the smallest size I tested, and added that point to the set that I would use to fit the scaling law.

The final set of points in consideration for the scaling law fit is shown below.

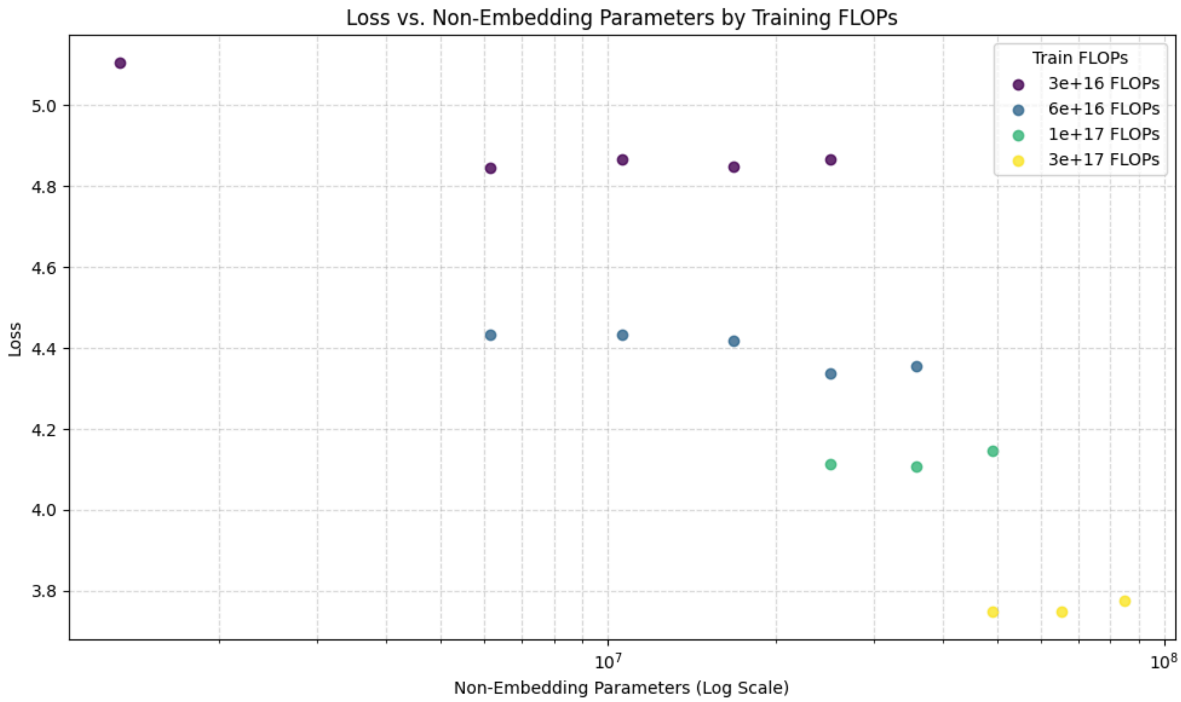


Figure 4: Non-embedding parameter count vs. loss at scales used to fit scaling laws

Fitting Scaling Laws

Fitting a quadratic to the data didn't look hopeful, so I used the empirical minimum at each scale. This gave a set of four $(C_i, N_{\text{opt}}(C_i), L_{\text{opt}}(C_i))$ triples to which to fit power laws using `scipy.optimize.curve_fit`. The resulting power laws are:

$$N_{\text{opt}}(C) = 2.77 \cdot C^{0.42} \tag{1}$$

$$L_{\text{opt}}(C) = 1.69 + [(4.14 \cdot 10^3) \cdot C^{-0.19}] \tag{2}$$

The fits are shown below.

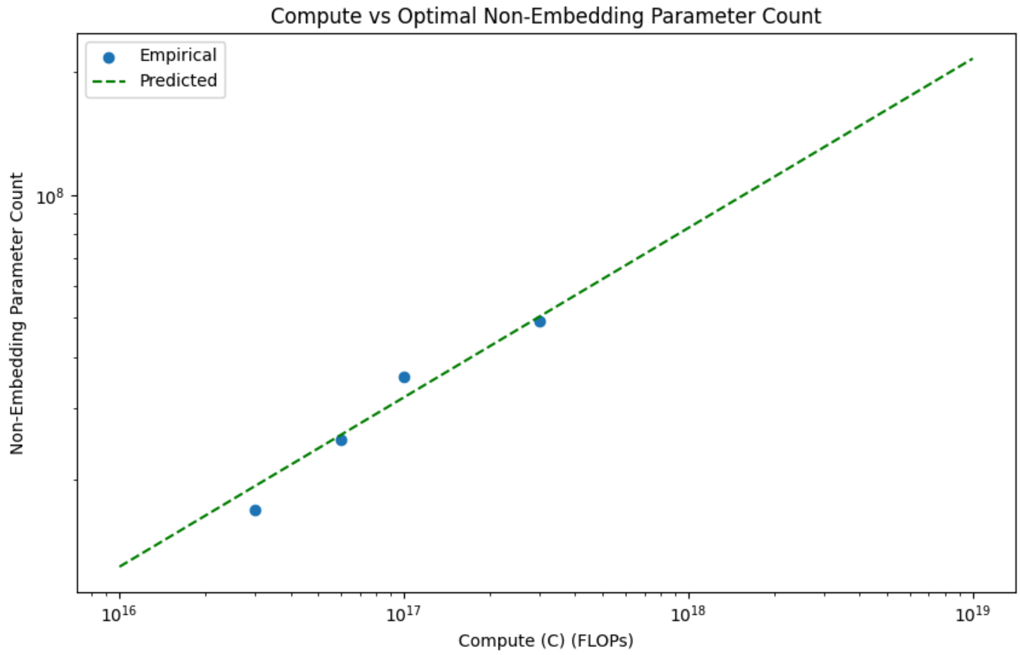


Figure 5: Compute vs. non-embedding parameter count fit

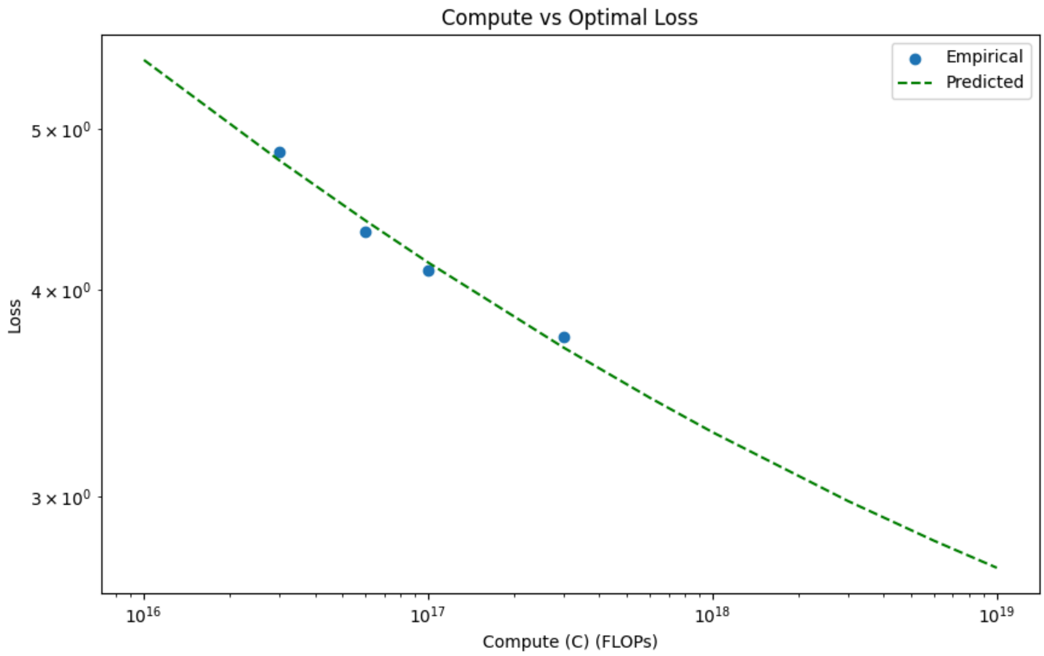


Figure 6: Compute vs. loss fit

Making Predictions

Plugging $C = 1 \cdot 10^{19}$ into the scaling laws, we get:

$$N_{\text{opt}}(1 \cdot 10^{19}) = 2.77 \cdot (1 \cdot 10^{19})^{0.42} = 216,042,918 \tag{3}$$

$$L_{\text{opt}}(1 \cdot 10^{19}) = 1.69 + \left[(4.14 \cdot 10^{3}) \cdot (1 \cdot 10^{19})^{-0.19}\right] = 2.72 \tag{4}$$

To recover the optimal model configuration, I simply extended the table of model sizes above to go beyong the training API's constraints, and picked the model with the closest number of non-embedding parameters (though it turns out that the predicted optimal model would be accepted by the training API).

The predicted optimal configuration is:

$$\begin{aligned} d_{\text{model}} &= 1024 \\ n_{\text{layers}} &= 16 \\ n_{\text{heads}} &= 16 \end{aligned} \tag{5}$$

Estimated number of parameters (in the realized model configuration, as opposed to the idealized number predicted by the scaling law):

$$\begin{aligned} N_{\text{non-embed}} &\approx 12 n_{\text{layers}} d_{\text{model}}^2 \\ &= 12 * 16 * 1024^2 \\ &= 201,326,592 \end{aligned} \tag{6}$$

$$\begin{aligned} N_{\text{total}} &\approx N_{\text{non-embed}} + (2 \cdot \text{vocab size} \cdot d_{\text{model}}) \\ &= 201,326,592 + (2 * 32,000 * 1024) \\ &= 266,862,592 \end{aligned} \tag{7}$$

Estimated number of tokens per parameter:

$$\begin{aligned} D &\approx C/(6 N_{\text{total}}) \\ &= 1 \cdot 10^{19}/(6 * 266,862,592) \\ &\approx 6,245,411,371 \text{ tokens} \end{aligned} \tag{8}$$

$$\begin{aligned} D/N_{\text{total}} &\approx 6,245,411,371/266,862,592 \\ &\approx 23.4 \text{ tokens/parameter} \end{aligned} \tag{9}$$

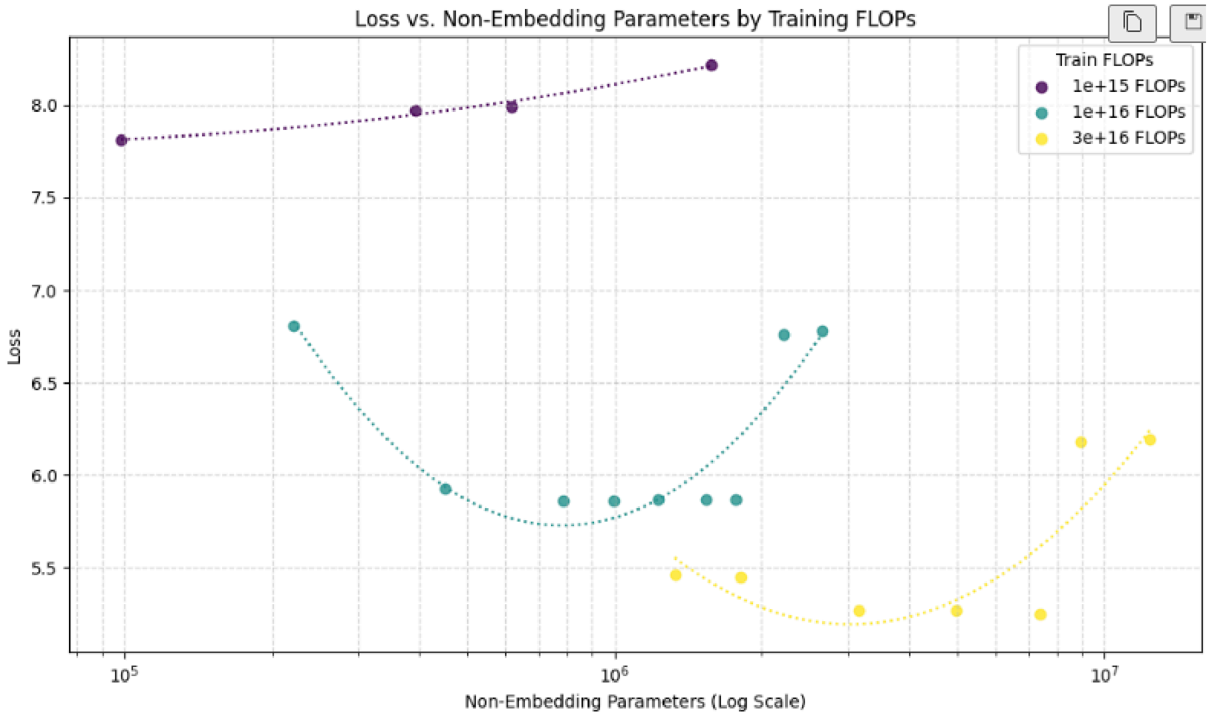
Batch size and learning rate:

Given that no divergence was observed in any of the queried runs, I would also use a batch size of 128 and a learning rate of `1e-3` to train this model.

Appendix

Range-based model configuration search

See below for the relationship between non-embedding parameter count vs. loss up to the `3e16` scale when allowing aspect ratio and head dimension to vary. After observing this, I decided to fix these parameters.



Bibliography

[1] J. Hoffmann *et al.*, "Training Compute-Optimal Large Language Models." [Online]. Available: https://arxiv.org/abs/2203.15556

[2] J. Kaplan *et al.*, "Scaling Laws for Neural Language Models." [Online]. Available: https://arxiv.org/abs/2001.08361

Index of Figures

Index of Tables