Appendix

A. Training dataset

In Table A1 we show the training dataset makeup used for *Chinchilla* and all scaling runs. Note that both the *MassiveWeb* and Wikipedia subsets are both used for more than one epoch.

	Disk Size	Documents	Sampling proportion	Epochs in 1.4T tokens
MassiveWeb	1.9 TB	604M	45% (48%)	1.24
Books	2.1 TB	4M	30% (27%)	0.75
C4	0.75 TB	361M	10% (10%)	0.77
News	2.7 TB	1.1B	10% (10%)	0.21
GitHub	3.1 TB	142M	4% (3%)	0.13
Wikipedia	0.001 TB	6M	1% (2%)	3.40

Table A1 | ***MassiveText* data makeup.** For each subset of *MassiveText*, we list its total disk size, the number of documents and the sampling proportion used during training—we use a slightly different distribution than in Rae et al. (2021) (shown in parenthesis). In the rightmost column show the number of epochs that are used in 1.4 trillion tokens.

B. Optimal cosine cycle length

One key assumption is made on the cosine cycle length and the corresponding learning rate drop (we use a 10× learning rate decay in line with Rae et al. (2021)).[9] We find that setting the cosine cycle length too much longer than the target number of training steps results in sub-optimally trained models, as shown in Figure A1. As a result, we assume that an optimally trained model will have the cosine cycle length correctly calibrated to the maximum number of steps, given the FLOP budget; we follow this rule in our main analysis.

C. Consistency of scaling results across datasets

We show scaling results from an IsoFLOP (Approach 2) analysis after training on two different datasets: C4 (Raffel et al., 2020b) and GitHub code (we show results with data from Rae et al. (2021)), results are shown in Table A2. For both set of experiments using subsets of *MassiveText*, we use the same tokenizer as the *MassiveText* experiments.

We find that the scaling behaviour on these datasets is very similar to what we found on *MassiveText*, as shown in Figure A2 and Table A2. This suggests that our results are independent of the dataset as long as one does not train for more than one epoch.

[9] We find the difference between decaying by 10× and decaying to 0.0 (over the same number of steps) to be small, though decaying by a factor of 10× to be slightly more performant. Decaying by less (5×) is clearly worse.

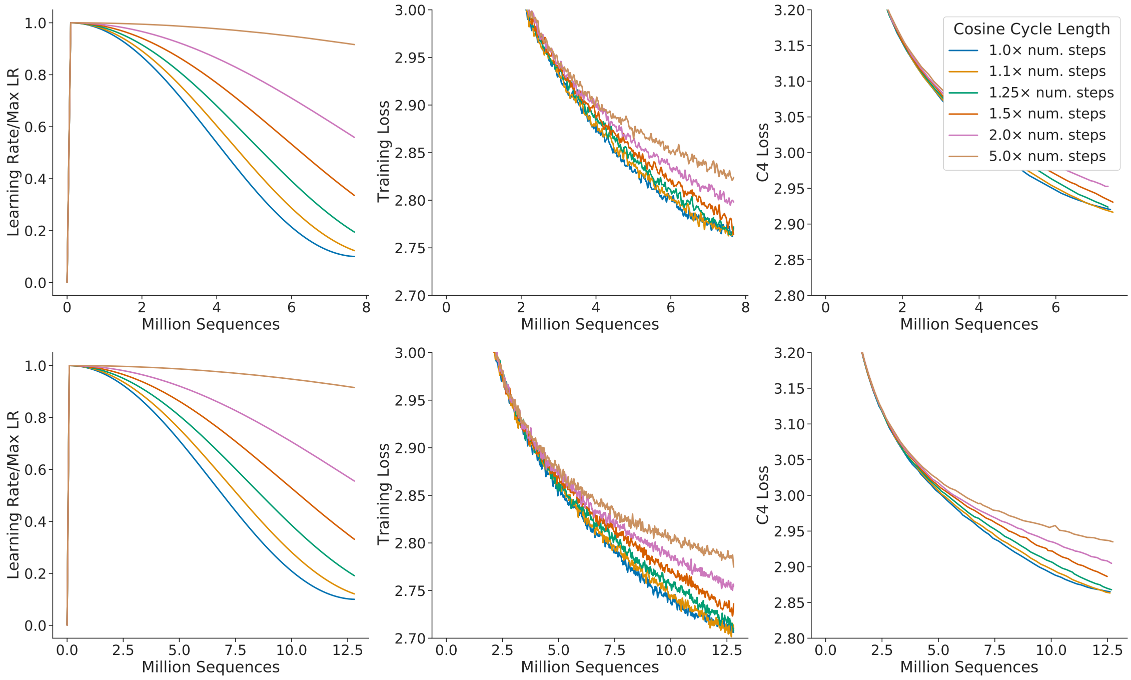

Figure A1 | **Grid over cosine cycle length.** We show 6 curves with the cosine cycle length set to 1, 1.1, 1.25, 1.5, 2, and 5× longer than the target number of training steps. When the cosine cycle length is too long, and the learning rate does not drop appropriately, then performance is impaired. We find that overestimating the number of training steps beyond 25% leads to clear drops in performance. We show results where we have set the number of training steps to two different values (top and bottom).

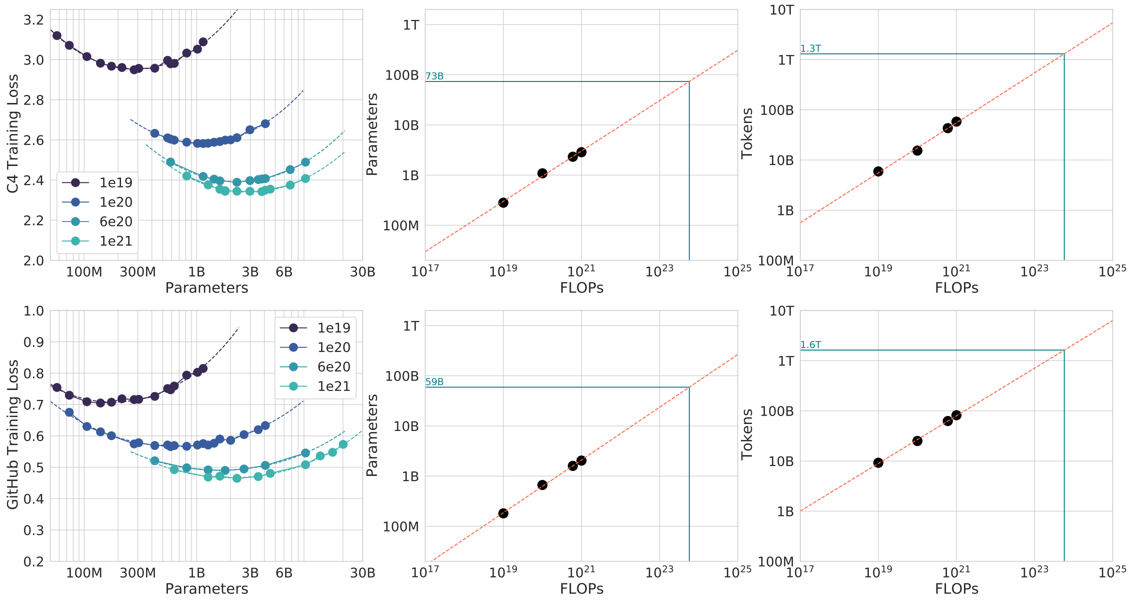

Figure A2 | **C4 and GitHub IsoFLOP curves.** Using the C4 dataset (Raffel et al., 2020b) and a GitHub dataset (Rae et al., 2021), we generate 4 IsoFLOP profiles and show the parameter and token count scaling, as in Figure 3. Scaling coefficients are shown in Table A2.

Approach	Coef. a where $N_{opt} \propto C^a$	Coef. b where $D_{opt} \propto C^b$
C4	0.50	0.50
GitHub	0.53	0.47
Kaplan et al. (2020)	0.73	0.27

Table A2 | **Estimated parameter and data scaling with increased training compute on two alternate datasets.** The listed values are the exponents, a and b, on the relationship $N_{opt} \propto C^a$ and $D_{opt} \propto C^b$. Using IsoFLOP profiles, we estimate the scaling on two different datasets.

D. Details on the scaling analyses

D.1. Approach 1: Fixing model sizes and varying training sequences

We use a maximum learning rate of 2×10^{-4} for the smallest models and 1.25×10^{-4} for the largest models. In all cases, the learning rate drops by a factor of 10× during training, using a cosine schedule. We make the assumption that the cosine cycle length should be approximately matched to the number of training steps. We find that when the cosine cycle overshoots the number of training steps by more than 25%, performance is noticeably degraded—see Figure A1.[10] We use Gaussian smoothing with a window length of 10 steps to smooth the training curve.

D.2. Approach 3: Parametric fitting of the loss

In this section, we first show how Equation (2) can be derived. We repeat the equation below for clarity,

$$\hat{L}(N, D) \triangleq E + \frac{A}{N^\alpha} + \frac{B}{D^\beta}, \tag{5}$$

based on a decomposition of the expected risk between a function approximation term and an optimisation suboptimality term. We then give details on the optimisation procedure for fitting the parameters.

Loss decomposition. Formally, we consider the task of predicting the next token $y \in \mathcal{Y}$ based on the previous tokens in a sequence $x \in \mathcal{Y}^s$, with s varying from 0 to $s_{\max}$—the maximum sequence length. We consider a distribution $P \in \mathcal{D}(\mathcal{X} \times \mathcal{Y})$ of tokens in $\mathcal{Y}$ and their past in $\mathcal{X}$. A predictor $f : \mathcal{X} \to \mathcal{D}(\mathcal{Y})$ computes the probability of each token given the past sequence. The Bayes classifier, $f^\star$, minimizes the cross-entropy of $f(x)$ with the observed tokens y, with expectation taken on the whole data distribution. We let L be the expected risk

$$L(f) \triangleq \mathbb{E}[\log f(x)_y], \qquad \text{and set} \qquad f^\star \triangleq \underset{f \in \mathcal{F}(\mathcal{X}, \mathcal{D}(\mathcal{Y}))}{\operatorname{argmin}} L(f). \tag{6}$$

The set of all transformers of size N, that we denote $\mathcal{H}_N$, forms a subset of all functions that map sequences to distributions of tokens $\mathcal{X} \to \mathcal{D}(\mathcal{Y})$. Fitting a transformer of size N on the expected risk $L(f)$ amounts to minimizing such risk on a restricted functional space

$$f_N \triangleq \underset{f \in \mathcal{H}_N}{\operatorname{argmin}} L(f). \tag{7}$$

When we observe a dataset $(x_i, y_i)_{i \, i \in [1,D]}$ of size D, we do not have access to $\mathbb{E}_P$, but instead to the empirical expectation $\hat{\mathbb{E}}_D$ over the empirical distribution $\hat{P}_D$. What happens when we are given D

[10] This further emphasises the point of not only determining model size, but also training length before training begins.