datapoints that we can only see once, and when we constrain the size of the hypothesis space to be N-dimensional ? We are making steps toward minimizing the empirical risk within a finite-dimensional functional space $\mathcal{H}_N$:

$$\hat{L}_D(f) \triangleq \hat{\mathbb{E}}_D[\log f(x)_y], \qquad \text{setting} \qquad \hat{f}_{N,D} \triangleq \operatorname*{argmin}_{f \in \mathcal{H}_N} \hat{L}_D(f). \tag{8}$$

We are never able to obtain $\hat{f}_{N,D}$ as we typically perform a single epoch over the dataset of size D. Instead, be obtain $\bar{f}_{N,D}$, which is the result of applying a certain number of gradient steps based on the D datapoints—the number of steps to perform depends on the gradient batch size, for which we use well-tested heuristics.

Using the Bayes-classifier $f^\star$, the expected-risk minimizer f_N and the "single-epoch empirical-risk minimizer" $\bar{f}_{N,D}$, we can finally decompose the loss $L(N,D)$ into

$$L(N,D) \triangleq L(\bar{f}_{N,D}) = L(f^\star) + \big(L(f_N) - L(f^\star)\big) + \big(L(\bar{f}_{N,D}) - L(f_N)\big)\,. \tag{9}$$

The loss comprises three terms: the Bayes risk, i.e. the minimal loss achievable for next-token prediction on the full distribution P, a.k.a the "entropy of natural text."; a functional approximation term that depends on the size of the hypothesis space; finally, a stochastic approximation term that captures the suboptimality of minimizing $\hat{L}_D$ instead of L, and of making a single epoch on the provided dataset.

Expected forms of the loss terms. In the decomposition (9), the second term depends entirely on the number of parameters N that defines the size of the functional approximation space. *On the set of two-layer neural networks*, it is expected to be proportional to $\frac{1}{N^{1/2}}$ (Siegel and Xu, 2020). Finally, given that it corresponds to early stopping in stochastic first order methods, the third term should scale as the convergence rate of these methods, which is lower-bounded by $\frac{1}{D^{1/2}}$ (Robbins and Monro, 1951) (and may attain the bound). This convergence rate is expected to be dimension free (see e.g. Bubeck, 2015, for a review) and depends only on the loss smoothness; hence we assume that the second term only depends on D in (2). Empirically, we find after fitting (2) that

$$L(N,D) = E + \frac{A}{N^{0.34}} + \frac{B}{D^{0.28}}, \tag{10}$$

with $E = 1.69$, $A = 406.4$, $B = 410.7$. We note that the parameter/data coefficients are both lower than $\frac{1}{2}$; this is expected for the data-efficiency coefficient (but far from the known lower-bound). Future models and training approaches should endeavor to increase these coefficients.

Fitting the decomposition to data. We effectively minimize the following problem

$$\min_{a,b,e,\alpha,\beta} \sum_{\text{Run } i} \text{Huber}_\delta\Big(\text{LSE}\big(a - \alpha \log N_i, b - \beta \log D_i, e\big) - \log L_i\Big), \tag{11}$$

where LSE is the log-sum-exp operator. We then set $A, B, E = \exp(a), \exp(b), \exp(e)$.

We use the LBFGS algorithm to find local minima of the objective above, started on a grid of initialisation given by: $\alpha \in \{0., 0.5, \ldots, 2.\}$, $\beta \in \{0., 0.5, \ldots, 2.\}$, $e \in \{-1., -.5, \ldots, 1.\}$, $a \in \{0, 5, \ldots, 25\}$, and $b \in \{0, 5, \ldots, 25\}$. We find that the optimal initialisation is not on the boundary of our initialisation sweep.

We use $\delta = 10^{-3}$ for the Huber loss. We find that using larger values of δ pushes the model to overfit the small compute regime and poorly predict held-out data from larger runs. We find that using a δ smaller than 10^{-3} does not impact the resulting predictions.

D.3. Predicted compute optimal frontier for all three methods

For Approaches 2 and 3, we show the estimated model size and number of training tokens for a variety of compute budgets in Table A3. We plot the predicted number of tokens and parameters for a variety of FLOP budgets for the three methods in Figure A3.

	Approach 2		Approach 3	
Parameters	FLOPs	Tokens	FLOPs	Tokens
400 Million	1.84e+19	7.7 Billion	2.21e+19	9.2 Billion
1 Billion	1.20e+20	20.0 Billion	1.62e+20	27.1 Billion
10 Billion	1.32e+22	219.5 Billion	2.46e+22	410.1 Billion
67 Billion	6.88e+23	1.7 Trillion	1.71e+24	4.1 Trillion
175 Billion	4.54e+24	4.3 Trillion	1.26e+24	12.0 Trillion
280 Billion	1.18e+25	7.1 Trillion	3.52e+25	20.1 Trillion
520 Billion	4.19e+25	13.4 Trillion	1.36e+26	43.5 Trillion
1 Trillion	1.59e+26	26.5 Trillion	5.65e+26	94.1 Trillion
10 Trillion	1.75e+28	292.0 Trillion	8.55e+28	1425.5 Trillion

Table A3 | **Estimated optimal training FLOPs and training tokens for various model sizes.** Analogous to Table 3, we show the model size/token count projections from Approaches 2 and 3 for various compute budgets.

.

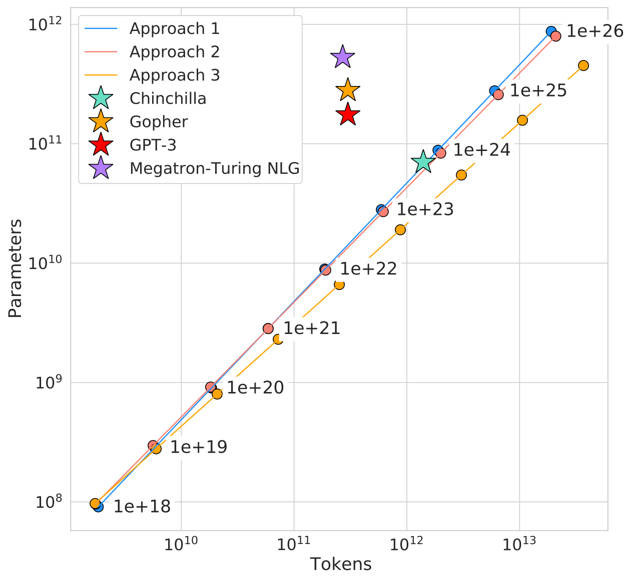

Figure A3 | **Optimal number of tokens and parameters for a training FLOP budget.** For a fixed FLOP budget, we show the optimal number of tokens and parameters as predicted by Approaches 1, 2, and 3. For an alternate representation, see Figure 1.

D.4. Small-scale comparison to Kaplan *et al.* (2020)

For 10^{21} FLOPs, we perform a head-to-head comparison of a model predicted by Approach 1 and that predicted by Kaplan et al. (2020). For both models, we use a batch size of 0.5M tokens and a maximum learning rate of 1.5×10^{-4} that decays by 10×. From Kaplan et al. (2020), we find that the optimal model size should be 4.68 billion parameters. From our approach 1, we estimate a 2.86 billion parameter model should be optimal. We train a 4.74 billion parameter and a 2.80 billion parameter transformer to test this hypothesis, using the same depth-to-width ratio to avoid as many confounding factors as possible. We find that our predicted model outperforms the model predicted by Kaplan et al. (2020) as shown in Figure A4.

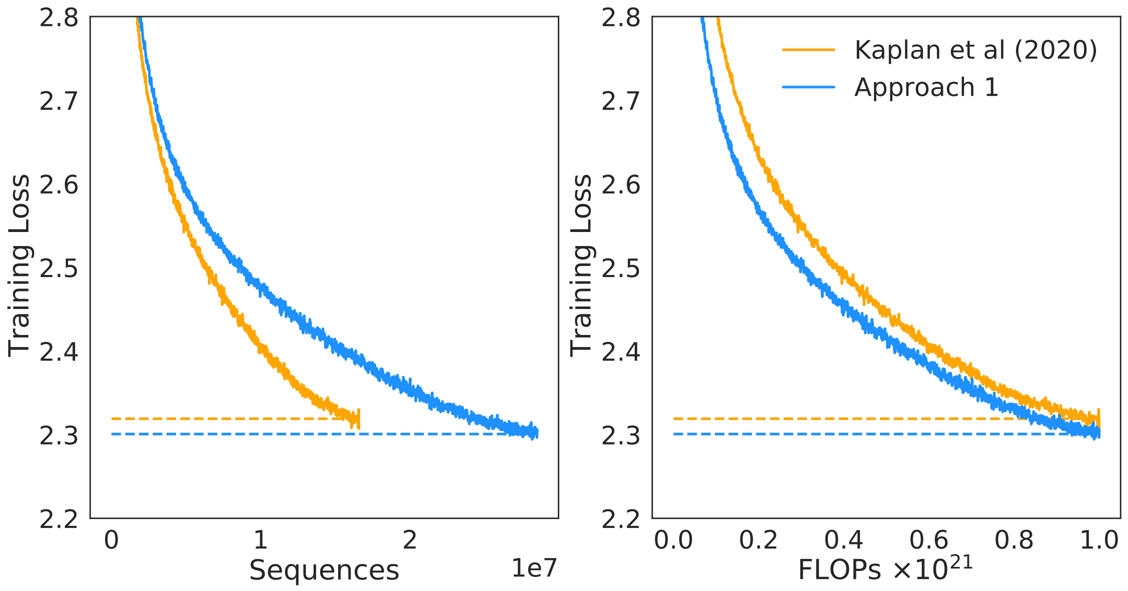

Figure A4 | **Comparison to Kaplan et al. (2020) at 10^{21} FLOPs.** We train 2.80 and 4.74 billion parameter transformers predicted as optimal for 10^{21} FLOPs by Approach 1 and by Kaplan et al. (2020). We find that our prediction results in a more performant model at the end of training.

E. Curvature of the FLOP-loss frontier

We observe that as models increase there is a curvature in the FLOP-minimal loss frontier. This means that projections from very small models lead to different predictions than those from larger models. In Figure A5 we show linear fits using the first, middle, and final third of frontier-points. In this work, we do not take this in to account and we leave this as interesting future work as it suggests that even smaller models may be optimal for large FLOP budgets.

F. FLOPs computation

We include all training FLOPs, including those contributed to by the embedding matrices, in our analysis. Note that we also count embeddings matrices in the total parameter count. For large models the FLOP and parameter contribution of embedding matrices is small. We use a factor of 2 to describe the multiply accumulate cost. For the forward pass, we consider contributions from:

- Embeddings
  - $2 \times \text{seq\_len} \times \text{vocab\_size} \times \text{d\_model}$
- Attention (Single Layer)
  - **Key, query and value projections**: $2 \times 3 \times \text{seq\_len} \times \text{d\_model} \times (\text{key\_size} \times \text{num\_heads})$