NYU

Section 06

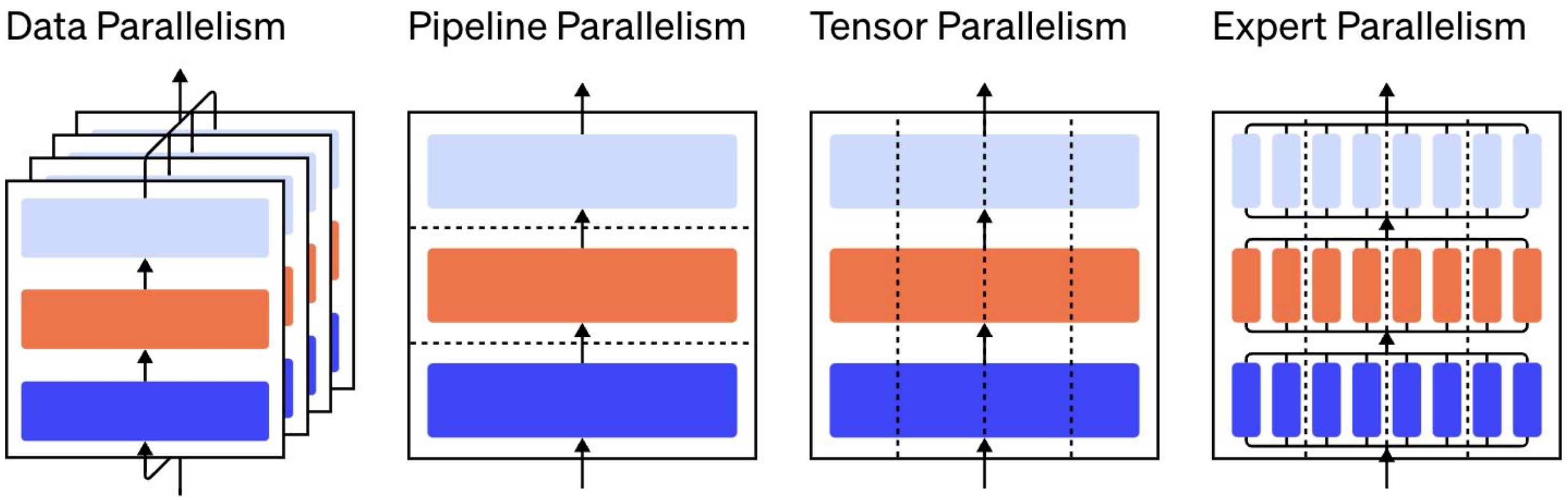
Xiang Pan

11/06/2023

PART 01

LLM Training

LLM Training



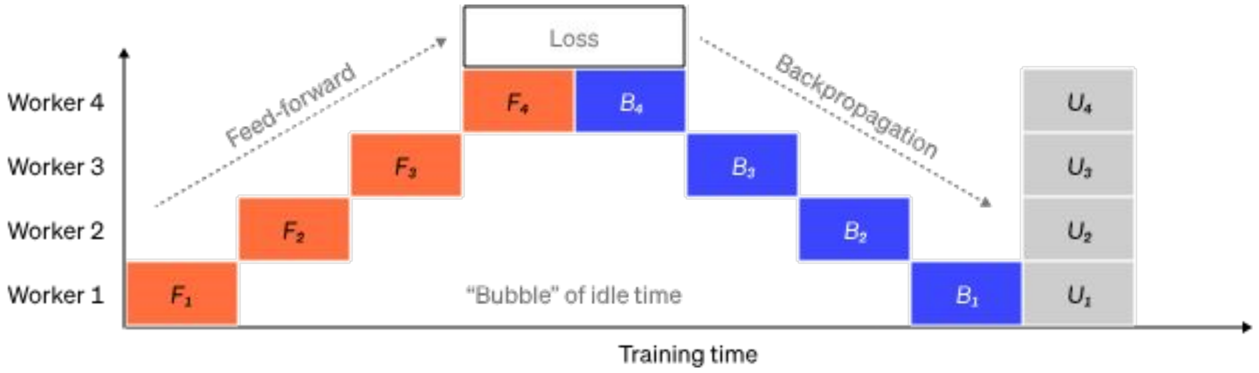
An illustration of various parallelism strategies on a three-layer model. Each color refers to one layer and dashed lines separate different GPUs.

Data Parallelism

- **Model Fits into a single GPU memory**
  - Keep multiple copies of parameters
- **Steps**:
  - independently compute the gradient on each worker;
  - average the gradients across workers;
    - **synchronous overhead!**
    - **asynchronous synchronization schemes?**
  - independently compute the same new parameters on each worker.

Pipeline Parallelism

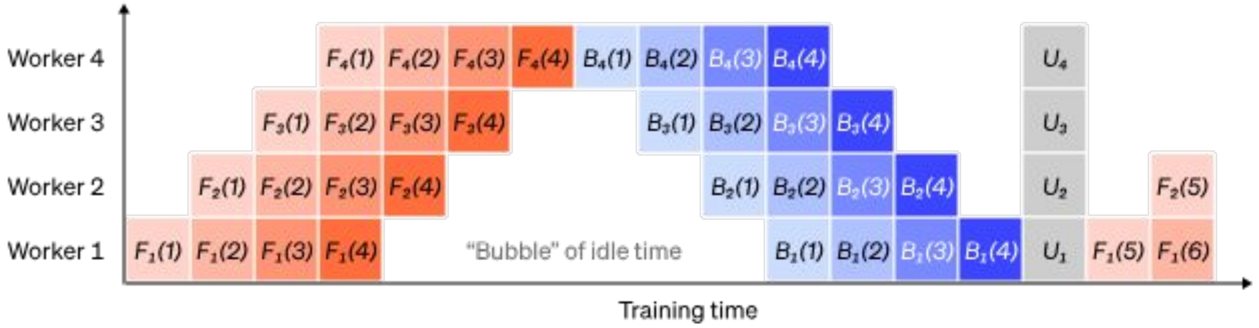
- ❑ waiting for outputs from the previous machine



NYU

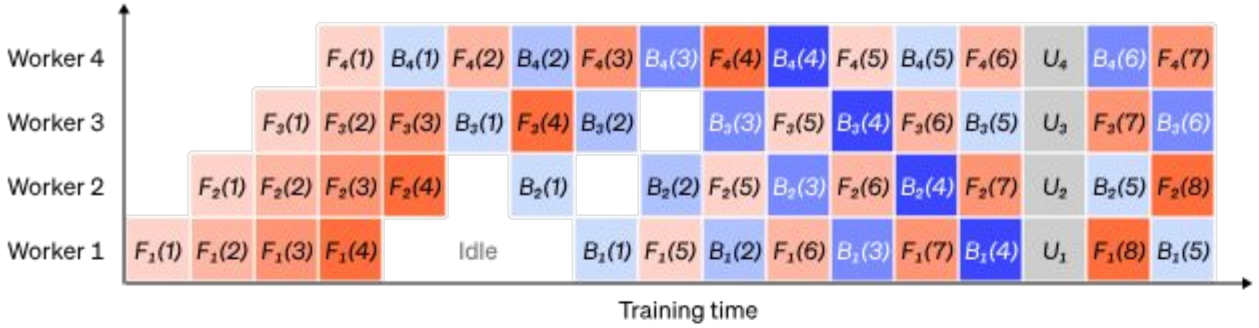
Pipeline Parallelism

- GPipe


Worker 4
Worker 3
Worker 2
Worker 1
$F_4(1)$ $F_4(2)$ $F_4(3)$ $F_4(4)$ $B_4(1)$ $B_4(2)$ $B_4(3)$ $B_4(4)$ U_4
$F_3(1)$ $F_3(2)$ $F_3(3)$ $F_3(4)$ $B_3(1)$ $B_3(2)$ $B_3(3)$ $B_3(4)$ U_3
$F_2(1)$ $F_2(2)$ $F_2(3)$ $F_2(4)$ $B_2(1)$ $B_2(2)$ $B_2(3)$ $B_2(4)$ U_2 $F_2(5)$
$F_1(1)$ $F_1(2)$ $F_1(3)$ $F_1(4)$ $B_1(1)$ $B_1(2)$ $B_1(3)$ $B_1(4)$ U_1 $F_1(5)$ $F_1(6)$
"Bubble" of idle time
Training time

Pipeline Parallelism

- PipeDream


Worker 4
Worker 3
Worker 2
Worker 1
$F_4(1)$ $B_4(1)$ $F_4(2)$ $B_4(2)$ $F_4(3)$ $B_4(3)$ $F_4(4)$ $B_4(4)$ $F_4(5)$ $B_4(5)$ $F_4(6)$ U_4 $B_4(6)$ $F_4(7)$
$F_3(1)$ $F_3(2)$ $F_3(3)$ $B_3(1)$ $F_3(4)$ $B_3(2)$ $B_3(3)$ $F_3(5)$ $B_3(4)$ $F_3(6)$ $B_3(5)$ U_3 $F_3(7)$ $B_3(6)$
$F_2(1)$ $F_2(2)$ $F_2(3)$ $F_2(4)$ $B_2(1)$ $B_2(2)$ $F_2(5)$ $B_2(3)$ $F_2(6)$ $B_2(4)$ $F_2(7)$ U_2 $B_2(5)$ $F_2(8)$
$F_1(1)$ $F_1(2)$ $F_1(3)$ $F_1(4)$ Idle $B_1(1)$ $F_1(5)$ $B_1(2)$ $F_1(6)$ $B_1(3)$ $F_1(7)$ $B_1(4)$ U_1 $F_1(8)$ $B_1(5)$
Training time

Tensor Parallelism

- ❑ Transformer/CNN -> Matrix Multiplication
- ❑ Sequence parallelism:

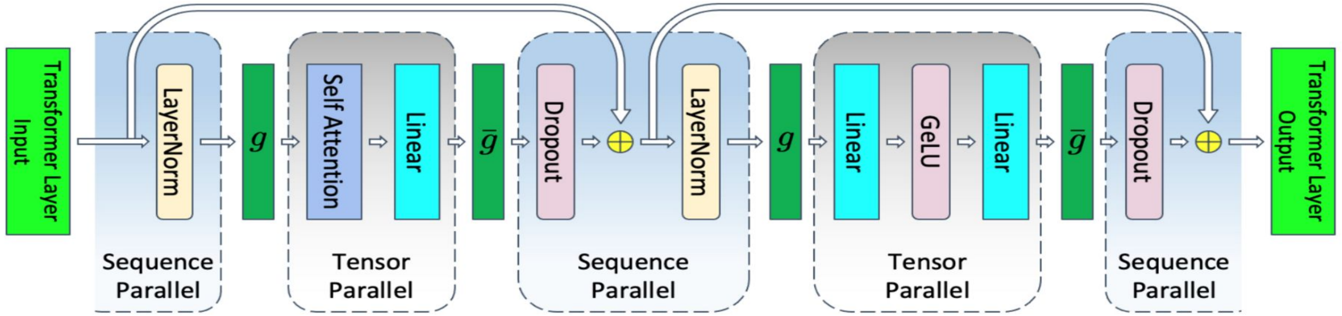
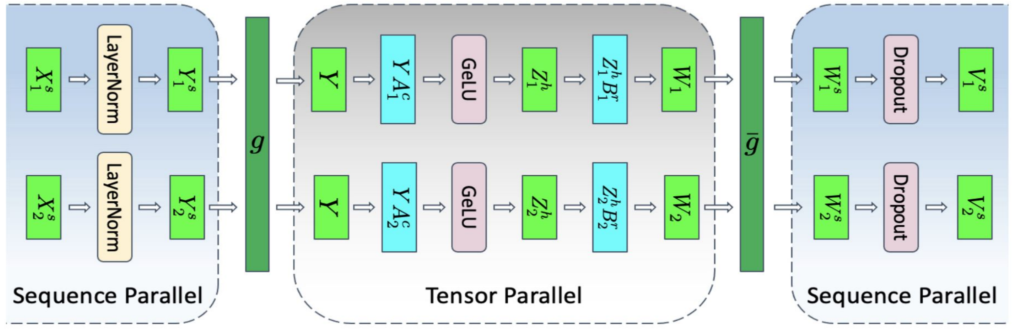


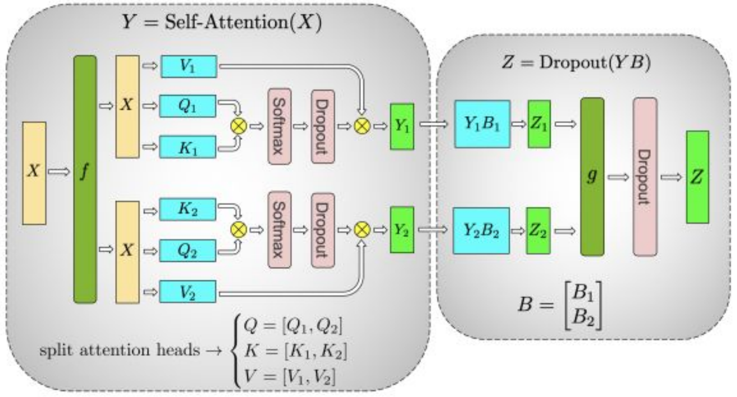
Figure 5: Transformer layer with tensor and sequence parallelism. g and $\bar{g}$ are conjugate. g is all-gather in the forward pass and reduce-scatter in the backward pass. $\bar{g}$ is reduce-scatter in forward pass and all-gather in backward pass.

PART 01

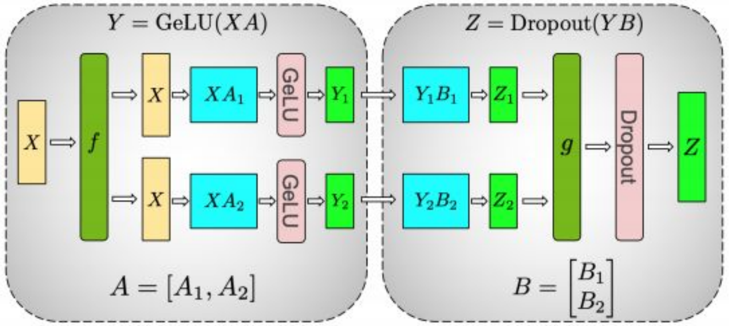
Tensor Parallelism



$$
\begin{aligned}
[Y_1^s, Y_2^s] &= \text{LayerNorm}([X_1^s, X_2^s]),\\
Y &= g(Y_1^s, Y_2^s),\\
[Z_1^h, Z_2^h] &= [\text{GeLU}(YA_1^c),\ \text{GeLU}(YA_2^c)],\\
W_1 &= Z_1^hB_1^r \quad \text{and} \quad W_2 = Z_2^hB_2^r,\\
[W_1^s, W_2^s] &= \bar{g}(W_1, W_2),\\
[V_1^s, V_2^s] &= [\text{Dropout}(W_1^s),\ \text{Dropout}(W_2^s)].
\end{aligned}
$$



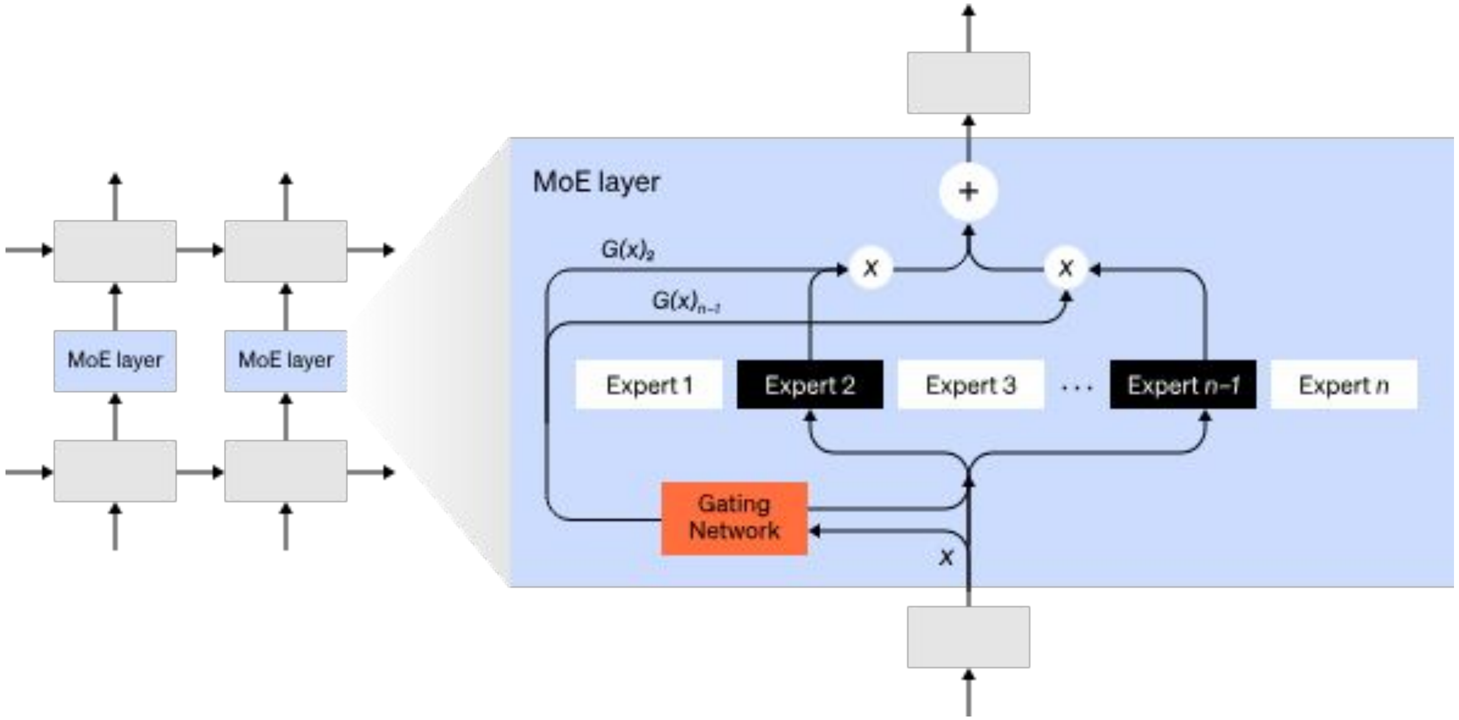
(b) Self-Attention



(a) MLP

NYU

MOE

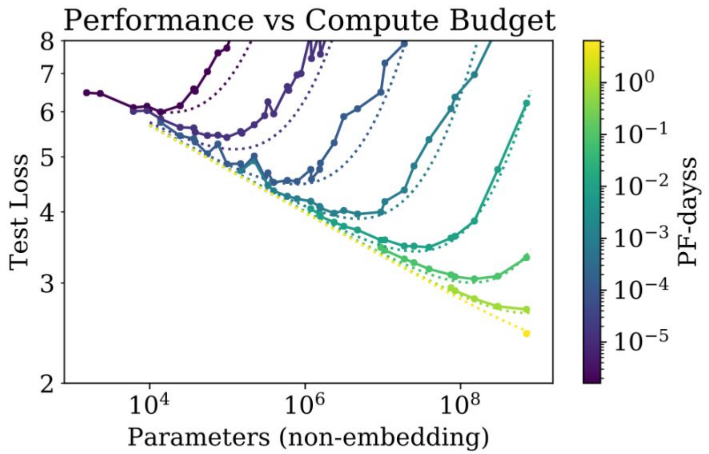

MoE layer
MoE layer
MoE layer
G(x)₂
G(x)ₙ₋₁
x
x
+
Expert 1
Expert 2
Expert 3
...
Expert n-1
Expert n
Gating Network
x

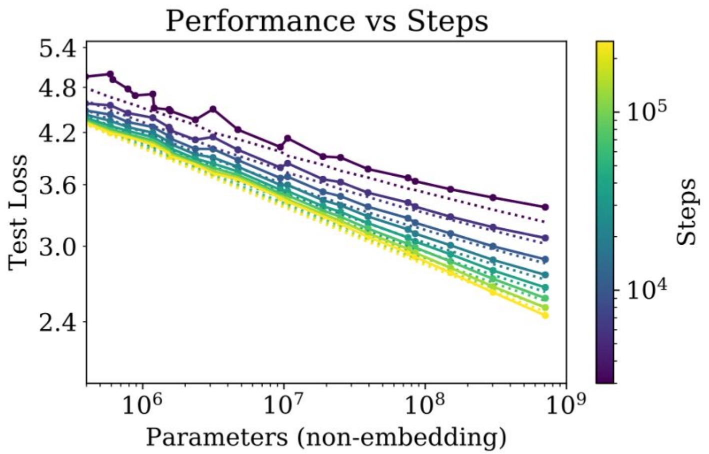
PART 02

Scaling Law for LLM Training

Scaling Law

Why do we care about scaling law?





NYU

Optimal Configuration for LLM

Given a fixed FLOPs budget, how should one trade-off model size and the number of training tokens?

FLOPs (Token, Parameter)

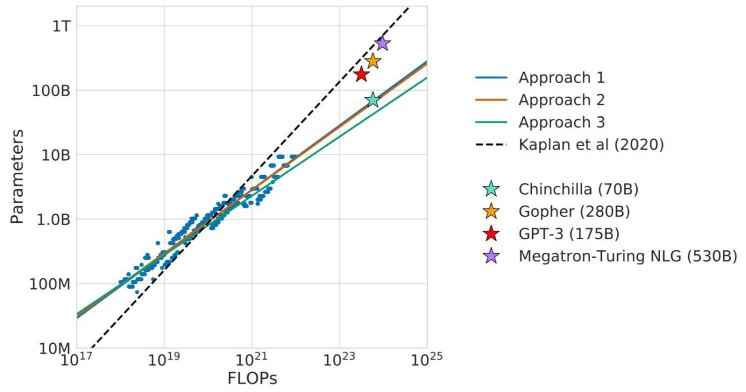


Figure 1 | **Overlaid predictions.** We overlay the predictions from our three different approaches, along with projections from Kaplan et al. (2020). We find that all three methods predict that current large models should be substantially smaller and therefore trained much longer than is currently done. 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).**

- Approach 1: Fix model sizes and vary number of training tokens
- Approach 2: IsoFLOP profiles
- Approach 3: Fitting a parametric loss function

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

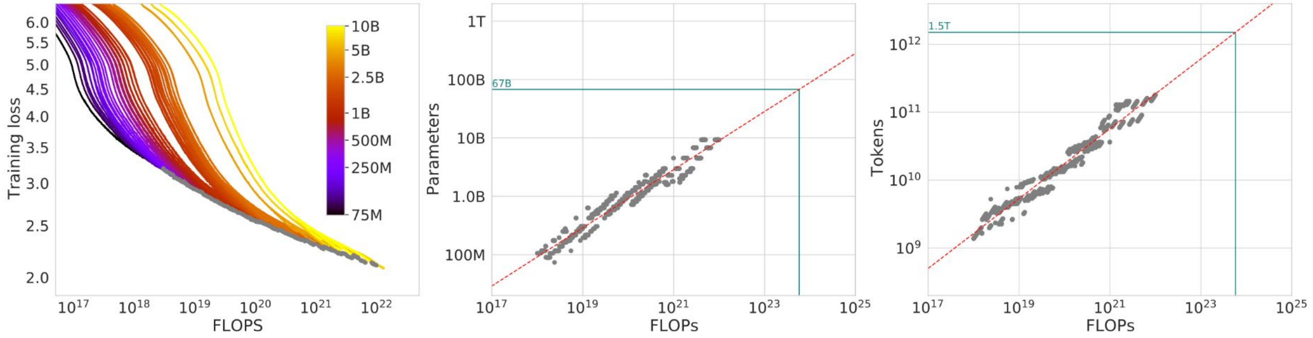


Figure 2 | **Training curve envelope.** On the **left** we show all of our different runs. We launched a range of model sizes going from 70M to 10B, each for four different cosine cycle lengths. From these curves, we extracted the envelope of minimal loss per FLOP, and we used these points to estimate 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}).

Approach 2: IsoFLOP profiles

For a given FLOP budget, what is the optimal parameter count?

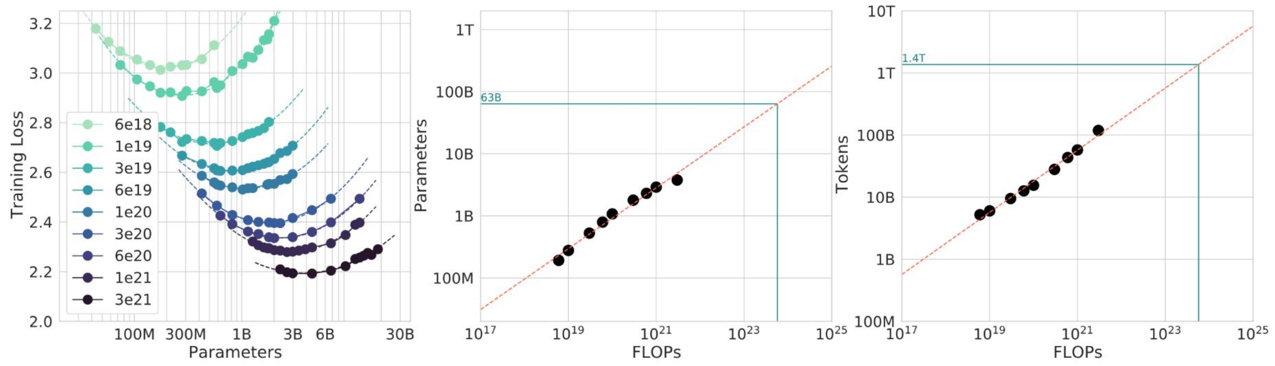
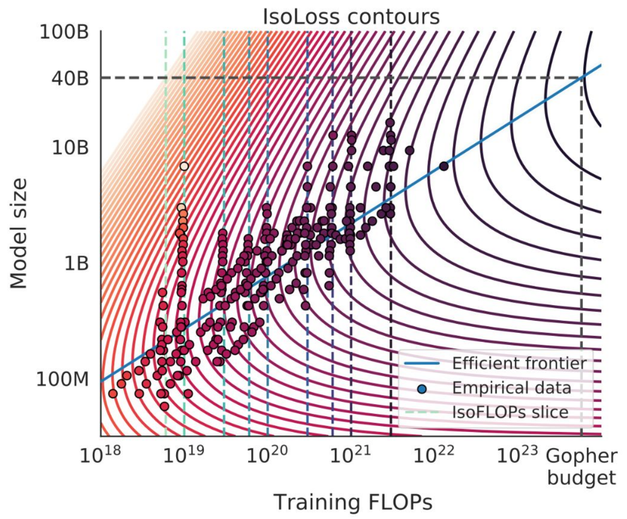


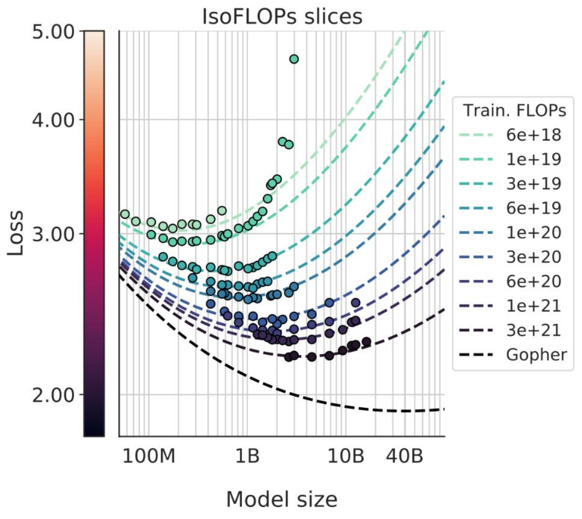
Figure 3 | **IsoFLOP curves.** For various model sizes, we choose the number of training tokens such that the final FLOPs is a constant. The cosine cycle length is set to match the target FLOP count. We find a clear valley in loss, meaning that for a given FLOP budget there is an optimal model to train (**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*.

Approach 3: Fitting a parametric loss function

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

N: Model Size
D: Token





FLOPs (Token, Parameter)

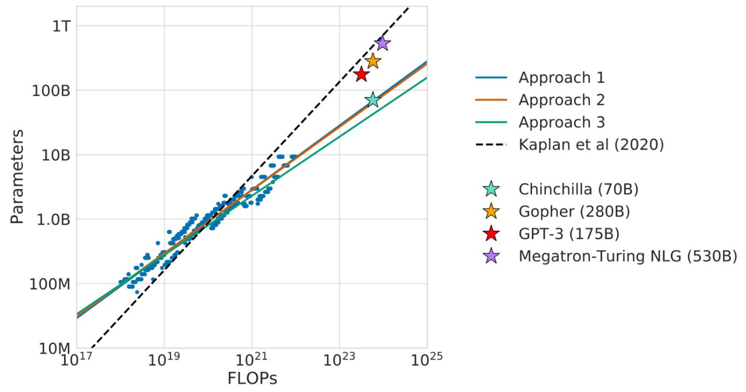
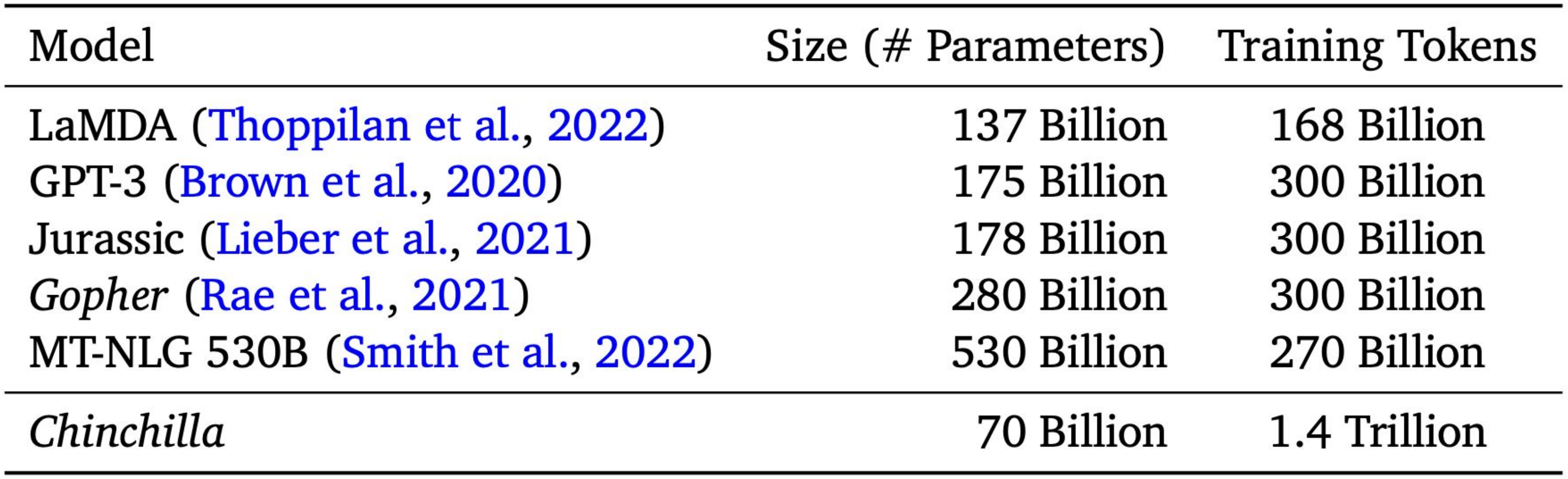


Figure 1 | **Overlaid predictions.** We overlay the predictions from our three different approaches, along with projections from Kaplan et al. (2020). We find that all three methods predict that current large models should be substantially smaller and therefore trained much longer than is currently done. 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).**

- Approach 1: Fix model sizes and vary number of training tokens
- Approach 2: IsoFLOP profiles
- Approach 3: Fitting a parametric loss function

FLOPs (Token, Parameter)



Model	Size (# Parameters)	Training Tokens
LaMDA (Thoppilan et al., 2022)	137 Billion	168 Billion
GPT-3 (Brown et al., 2020)	175 Billion	300 Billion
Jurassic (Lieber et al., 2021)	178 Billion	300 Billion
Gopher (Rae et al., 2021)	280 Billion	300 Billion
MT-NLG 530B (Smith et al., 2022)	530 Billion	270 Billion
Chinchilla	70 Billion	1.4 Trillion