Training Large Language Models

Scaling Laws and Chinchilla

Learning goals

- Understand Chinchilla
- Understand the various scaling laws

NUMBER OF PARAMETERS: NOTATION

- Up to now in this chapter: number of parameters = P
- From now on: number of parameters = N

SCALING LAWS

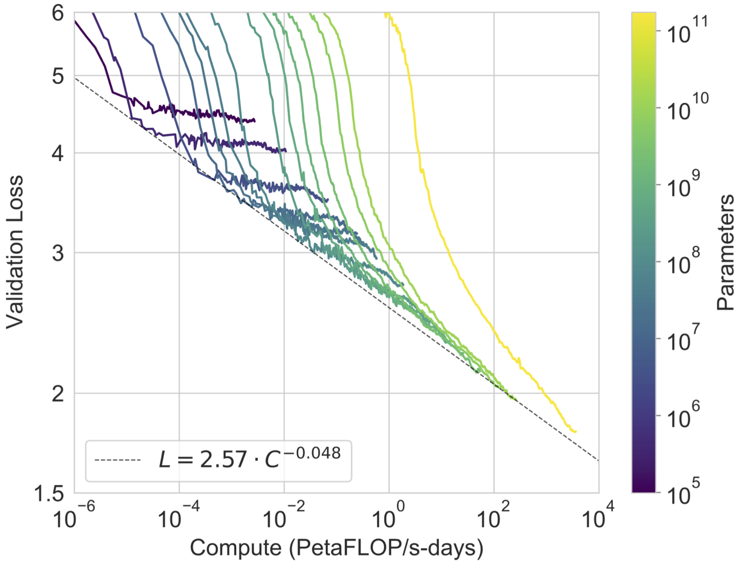
Kaplan et al. (2020)

- Performance depends strongly on scale, weakly on model shape
  - Scale means: parameters N, data D, and compute C
  - Shape means: depth and width
- Smooth power laws
  - Performance has power-law relation with each factor N, D, C
  - When not bottlenecked by the other two
  - Trend spanning more than six orders of magnitude
- Universality of overfitting
  - Performance enters regime of diminishing returns if N or D held fixed while the other increases

SCALING LAWS

- Universality of training
  - Training curves follow predictable power-laws
  - Their parameters are roughly independent of model size
  - It is possible to predict by extrapolating the early part of the training curve
- Transfer improves with test performance
  - When evaluating on text with different distribution from training text, results are strongly correlated to those on the validation set
  - Transfer to different distribution incurs a constant penalty but improves in line with performance on training set
- Sample efficiency
  - Large models are more sample-efficient than small models
  - They reach same performance with fewer optimization steps

POWER LAW (GPT3 PAPER)



SCALING LAWS

- Convergence is inefficient
  - When C is fixed but N and D are not, optimal performance is achived by training very large models and stopping significantly short of convergence
- Optimal batch size
  - "Optimal batch size: The ideal batch size for training these models is roughly a power of the loss only, and continues to be determinable by measuring the gradient noise scale [MKAT18]; it is roughly 1-2 million tokens at convergence for the largest models we can train."
  - gradient noise scale = a measure of the signal-to-noise ratio of gradient across training examples

Larger language models will perform better and be more sample efficient than current models.

OPTIMAL BATCH SIZE

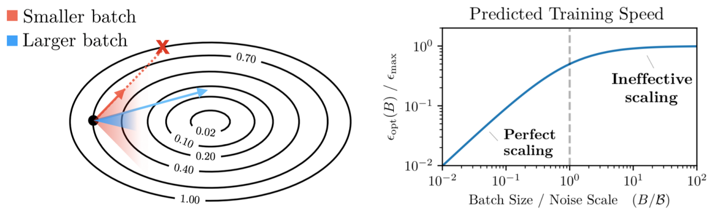


Figure 3: Larger batch sizes yield estimated gradients that are closer to the true gradient, on average. Larger step sizes can be used when the estimated gradient is closer to the true gradient, so more progress can be made per step. **Left:** A large step size used with a small batch size can lead to instability, as illustrated for a quadratic loss. **Right:** Equation 2.6 predicts that the 'turning point' after which larger batch sizes become less helpful is the noise scale $\mathcal{B}$, where the training speed drops to 50% of the maximum possible.

- Estimate gradient as accurately as possible $\rightarrow$ large batch
- Increase training speed as much as possible $\rightarrow$ large step size
- Based on the estimated gradient, choose a step size such that the cost of the landing position does not deviate too much from the cost of the ideal landing position $\rightarrow$ small step size

OPTIMAL BATCH SIZE

- Exploit stochasticity (epoch batches would not be a good thing even if we could compute them efficiently) $\rightarrow$ small step size

SCALING LAW FOR NEXT WORD PREDICTION

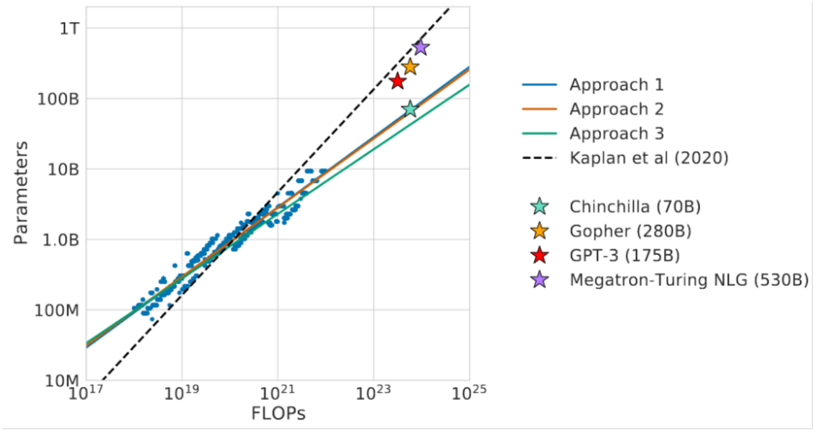
- $L(N, D) = 1.61 + \frac{406.4}{N^{0.34}} + \frac{410.7}{D^{0.28}}$
- $L(N, D)$ is cross entropy on new text

COMPUTE-OPTIMAL LLMs

Given a fixed FLOPs budget, how should we trade-off model size and text size to optimize performance? ▸ Hoffmann et al., 2022

- Find N and D so that $FLOPs(N, D) = C$ and $L(N, D)$ is minimal
- Empirically estimated N and D based on 400 models.
  - Ranging from 70 M to 16 B parameters
  - Trained on 5 B to 400 B tokens
- Different results from those of ▸ Kaplan et al., 2020
- Results verified using Chinchilla
  - Chinchilla has 70 B parameters and is trained on 1.4 T tokens
  - 4x less parameters and 4x more tokens than Gopher
  - Chinchilla outruns Gopher and has reduced memory footprint and inference cost

COMPUTE-OPTIMAL LLMs: ARE GPT3 ETC TOO LARGE?

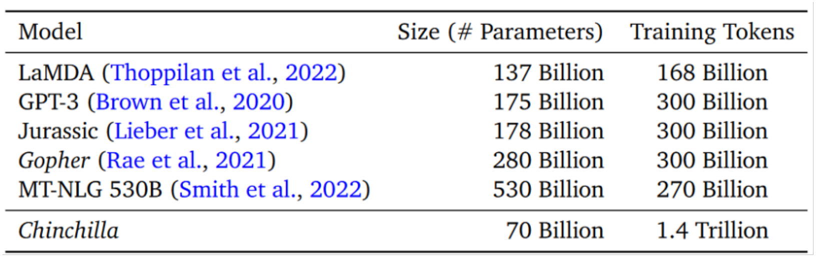


▸ Source: Hoffmann et al., 2022

COMPUTE-OPTIMAL LLMs (2)

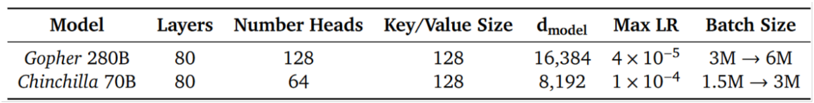
Given a fixed FLOPs budget, how should one trade off model size and the number of training tokens? We find that all three methods predict that current large models should be substantially smaller and therefore trained much longer than is currently done. Based on our estimated compute-optimal frontier, we predict that for the compute budget used to train Gopher, an optimal model should be 4 times smaller, while being training on 4 times more tokens. We verify this by training a more compute-optimal 70B model, called Chinchilla, on 1.4 trillion tokens. Not only does Chinchilla outperform its much larger counterpart, Gopher, but its reduced model size reduces inference cost considerably and greatly facilitates downstream uses on smaller hardware. The energy cost of a large language model is amortized through its usage for inference and fine-tuning. The benefits of a more optimally trained smaller model, therefore, extend beyond the immediate benefits of its improved performance.

CHINCHILLA AND THE OTHER LLMs



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

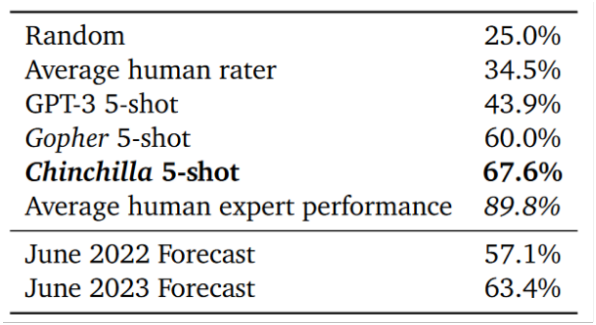
▸ Source: Hoffmann et al., 2022



Model	Layers	Number Heads	Key/Value Size	d_{model}	Max LR	Batch Size
Gopher 280B	80	128	128	16,384	4×10^{-5}	3M $\rightarrow$ 6M
Chinchilla 70B	80	64	128	8,192	1×10^{-4}	1.5M $\rightarrow$ 3M

▸ Source: Hoffmann et al., 2022

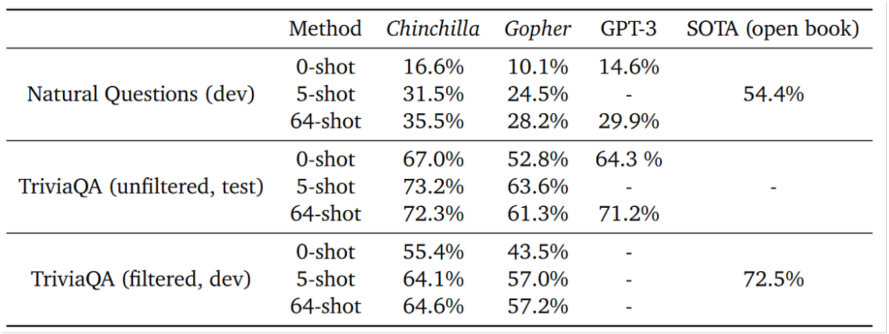
CHINCHILLA OUTPERFORMS OTHER LLMs: MMLU



Random	25.0%
Average human rater	34.5%
GPT-3 5-shot	43.9%
Gopher 5-shot	60.0%
***Chinchilla* 5-shot**	**67.6%**
Average human expert performance	*89.8%*
June 2022 Forecast	57.1%
June 2023 Forecast	63.4%

▸ Source: Hoffmann et al., 2022

CHINCHILLA OUTPERFORMS OTHER LLMs: QA



	Method	*Chinchilla*	*Gopher*	GPT-3	SOTA (open book)
	0-shot	16.6%	10.1%	14.6%	
Natural Questions (dev)	5-shot	31.5%	24.5%	-	54.4%
	64-shot	35.5%	28.2%	29.9%	
	0-shot	67.0%	52.8%	64.3 %	
TriviaQA (unfiltered, test)	5-shot	73.2%	63.6%	-	-
	64-shot	72.3%	61.3%	71.2%	
	0-shot	55.4%	43.5%	-	
TriviaQA (filtered, dev)	5-shot	64.1%	57.0%	-	72.5%
	64-shot	64.6%	57.2%	-	

‣ Source: Hoffmann et al., 2022