Scaling Laws for Neural Language Models (2020)

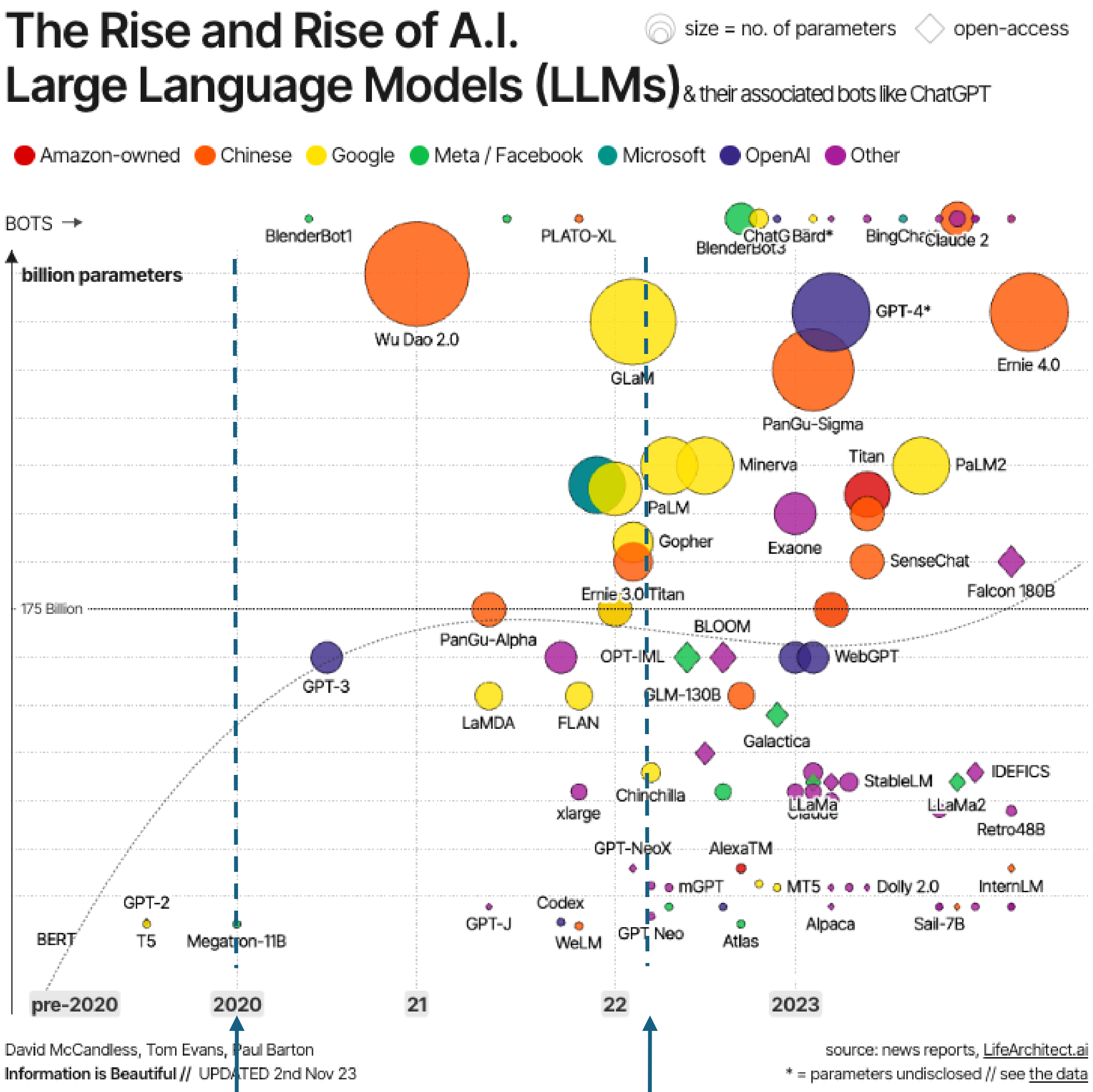
Jared Kaplan, Sam McCandlish et Al.

from OpenAI

Context

- **Language as a Natural Domain for AI:**
  - Most **reasoning tasks** can be resolved with language.
  - The abundance of **text data worldwide** permits powerful learning.

- **Rapid Progress in Deep Learning for LLM:**
  - Models closer to **human-level performance** on many tasks (Transformers, BERT …)
  - Models now excel in tasks like composing **coherent, multi-paragraph prompted text samples**.



Kaplan et al.

Hoffmann et al.

Art1 - Scaling Laws for Neural Language Models
Why this article?

Goal:

- Empirically investigate how **language modeling performance** (ie the Loss function) is influenced by **various factors**
- Based on this analysis, determine the optimal **trade-off** between **model size** and **dataset size** for a given **training compute budget**.

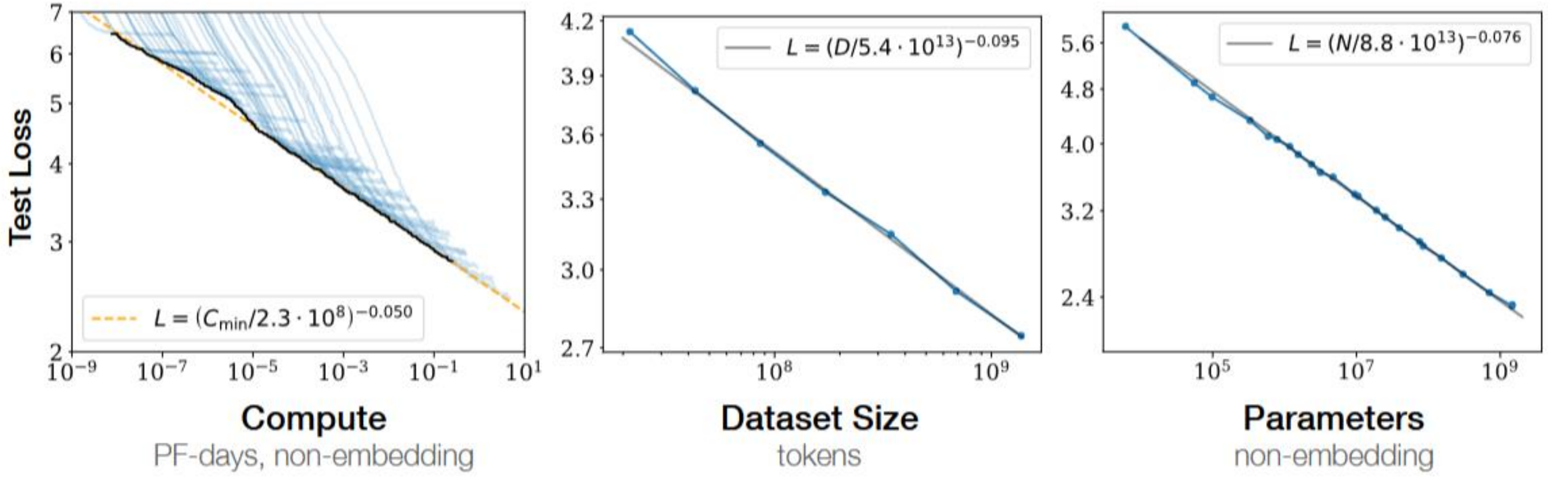
Results:

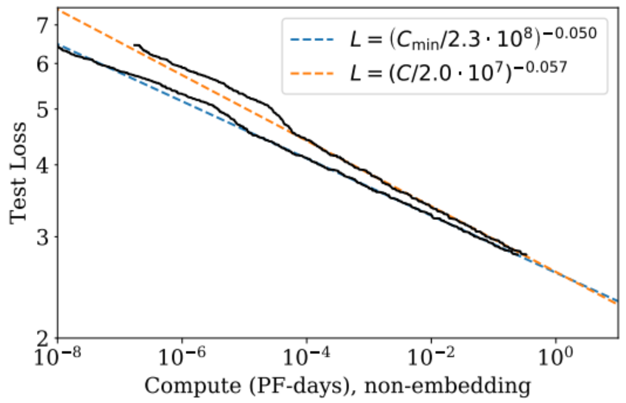
- Influence of **factors**:
  - Training compute (C)
  - Dataset size (D)
  - Model size (N)
  - Model shapes (Width & Depth)
  - Batch size (B)
  - Number of steps (S)

- Loss follows power laws
- On very large models, it is more efficient to stop training early
- $D \propto N^{0.74}$ (False: Chinchilla) should evolve in tandem
- Existence of optimal batch size
- Transfer incurs only a constant penalty
- Sample efficiency

Art1 - Scaling Laws for Neural Language Models

Main results – Empirical laws





With **X** as the factor, **X_c** as the constant (meaningless) and **α** as the scale factor :

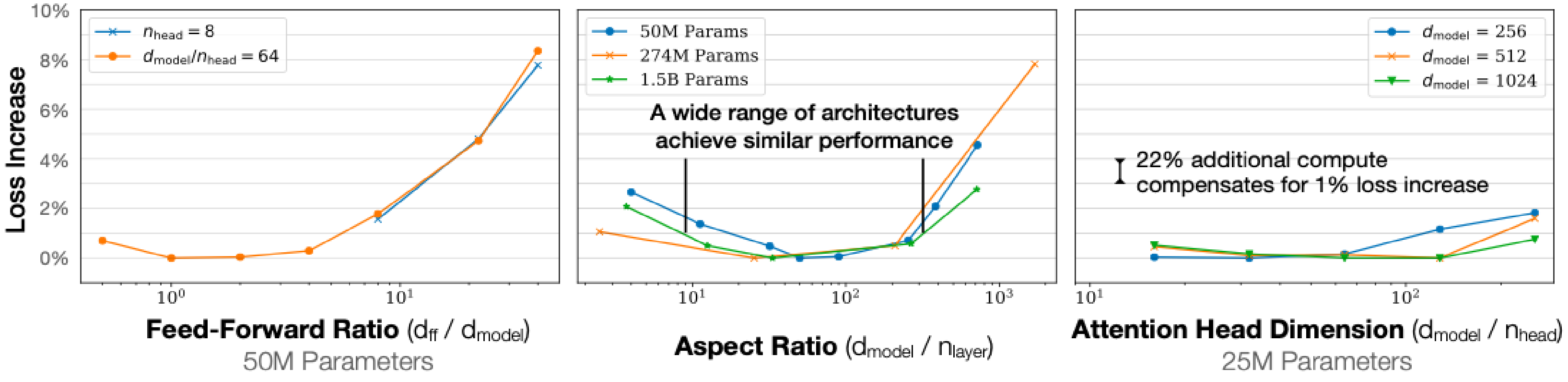
$$\boxed{L(X) = \left(\frac{X_c}{X}\right)^{\alpha x}}$$

Power Law	**Scale (tokenization-dependent)**
$\alpha_N = 0.076$	$N_c = 8.8 \times 10^{13}$ params (non-embed)
$\alpha_D = 0.095$	$D_c = 5.4 \times 10^{13}$ tokens
$\alpha_C = 0.057$	$C_c = 1.6 \times 10^7$ PF-days
$\alpha_C^{min} = 0.050$	$C_c^{min} = 3.1 \times 10^8$ PF-days
$\alpha_B = 0.21$	$B_* = 2.1 \times 10^8$ tokens
$\alpha_S = 0.76$	$S_c = 2.1 \times 10^3$ steps

Art1 - Scaling Laws for Neural Language Models

Main results - Model shape dependency & Sample Efficiency

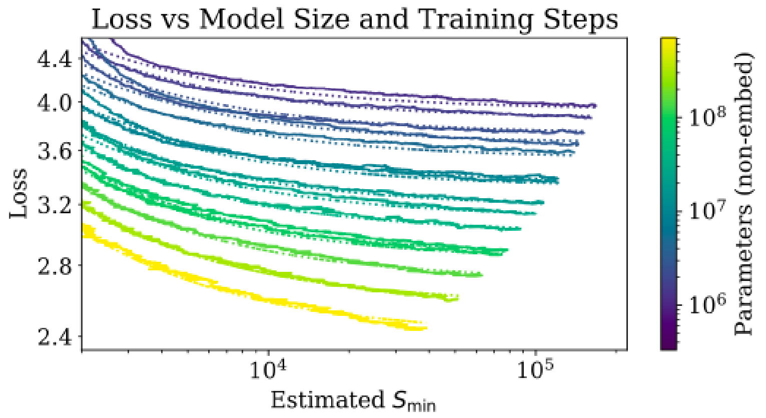
Performances are **weakly** affected by **model shape** for a Transformer.



Feed-Forward Width **Depth** **Number of Attention heads**

dmodel is the dimensionality of the model's hidden states

Smin: Estimated **steps** to obtain a **given compute**



The more the **size grows**, the more the model is **sample efficient**

For a **given compute**, you can go with a **big model** with **early stopping**

Art1 - Scaling Laws for Neural Language Models
Experiment

- **Training on WebText2 (dataset)**
  - $n_{vocab} = 50257$
  - Loss : cross-entropy over 1024 token context

- **Training factors:**
  - Variation of non-embedding **parameters number**
    768M → 1500M parameters
  - Adam optimizer with a **fixed number of training tokens** (most of runs)
    2.5x10^5 steps with batches of 512 sequences of 1024 tokens
  - Learning rate schedule : linear warmup followed by a cosine decay

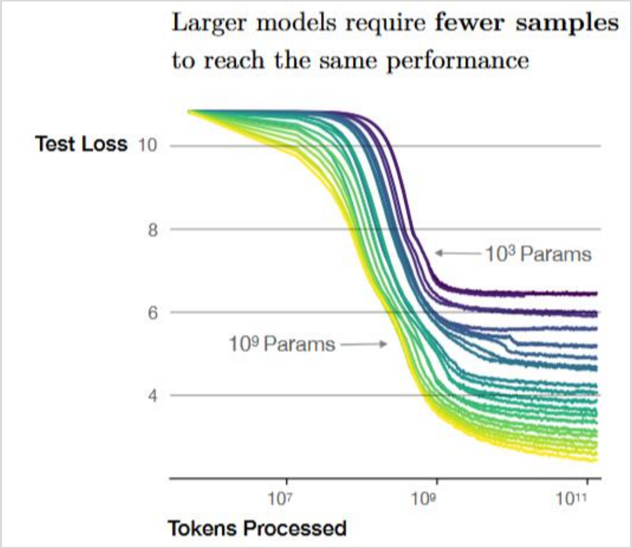
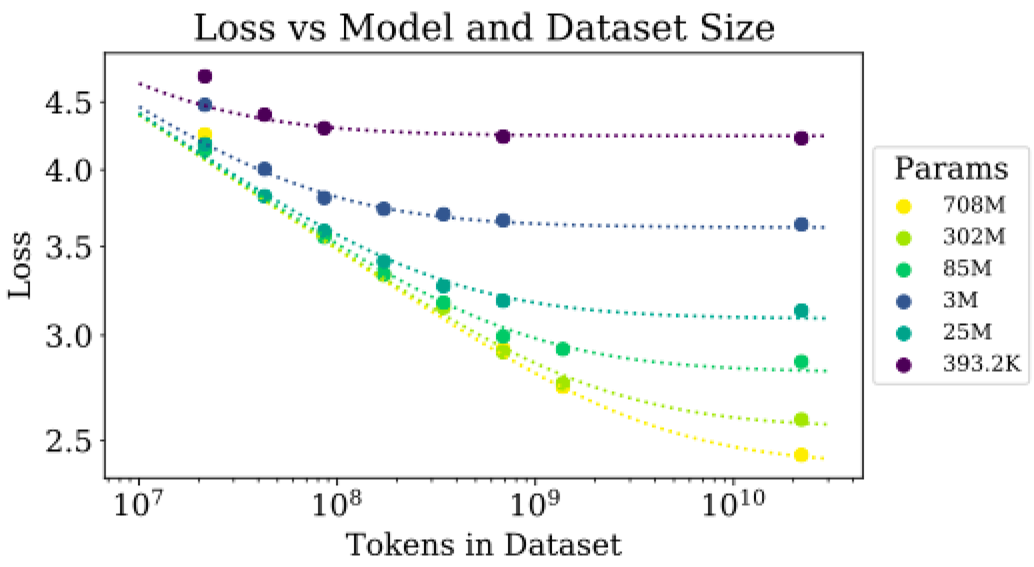


Figure 2 – some of the training runs

Art1 - Scaling Laws for Neural Language Models

Experiment

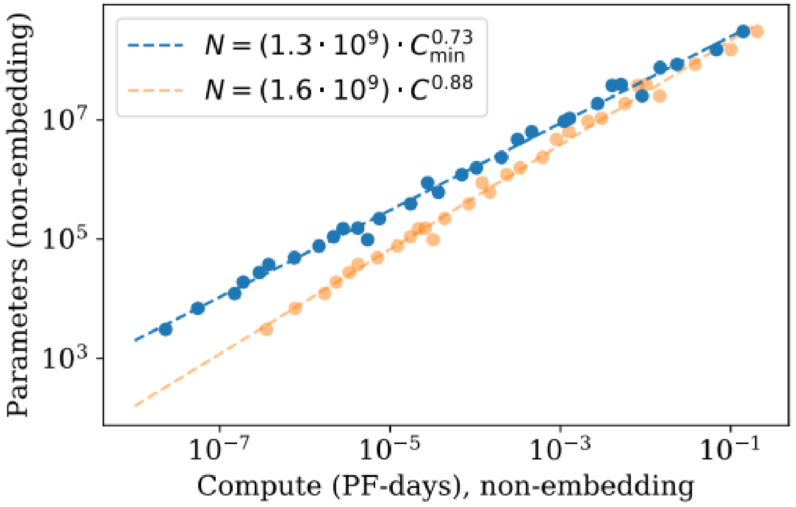
$$L(N,D) = \left[\left(\frac{N_c}{N}\right)^{\frac{\alpha_N}{\alpha_D}} + \frac{D_c}{D}\right]^{\alpha_D}$$



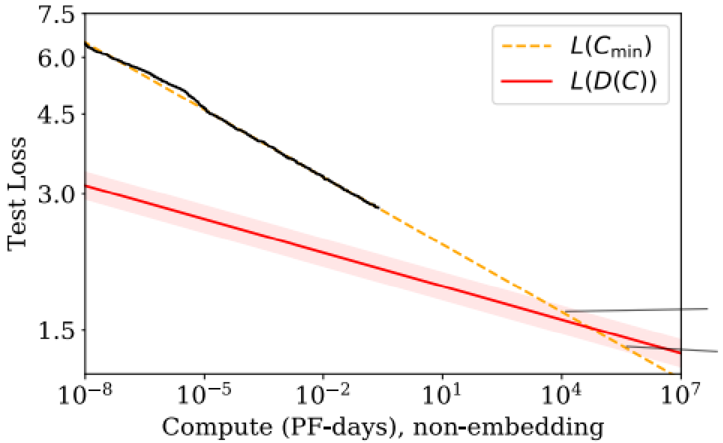
- **<u>Vocabulary size changes :</u>**
  - Changes in vocabulary size or tokenization should scale the loss proportionally, so L(N,D) must support this scaling naturally
- **<u>Asymptotic behavior of loss :</u>**
  - As N→∞ (with fixed D), the loss should approach L(D); as D→∞ (with fixed N), it should approach L(N)
- **<u>Series expansion at large D:</u>**
  - L(N,D) should allow a series expansion in 1/D, though this principle has weaker theoretical backing.

Art1 - Scaling Laws for Neural Language Models

Contradiction



Curve of optimal parameters for optimal allocation



To keep underfitting under control :

$$D \propto N^{0.74} \propto C_{\rm min}^{0.54}$$

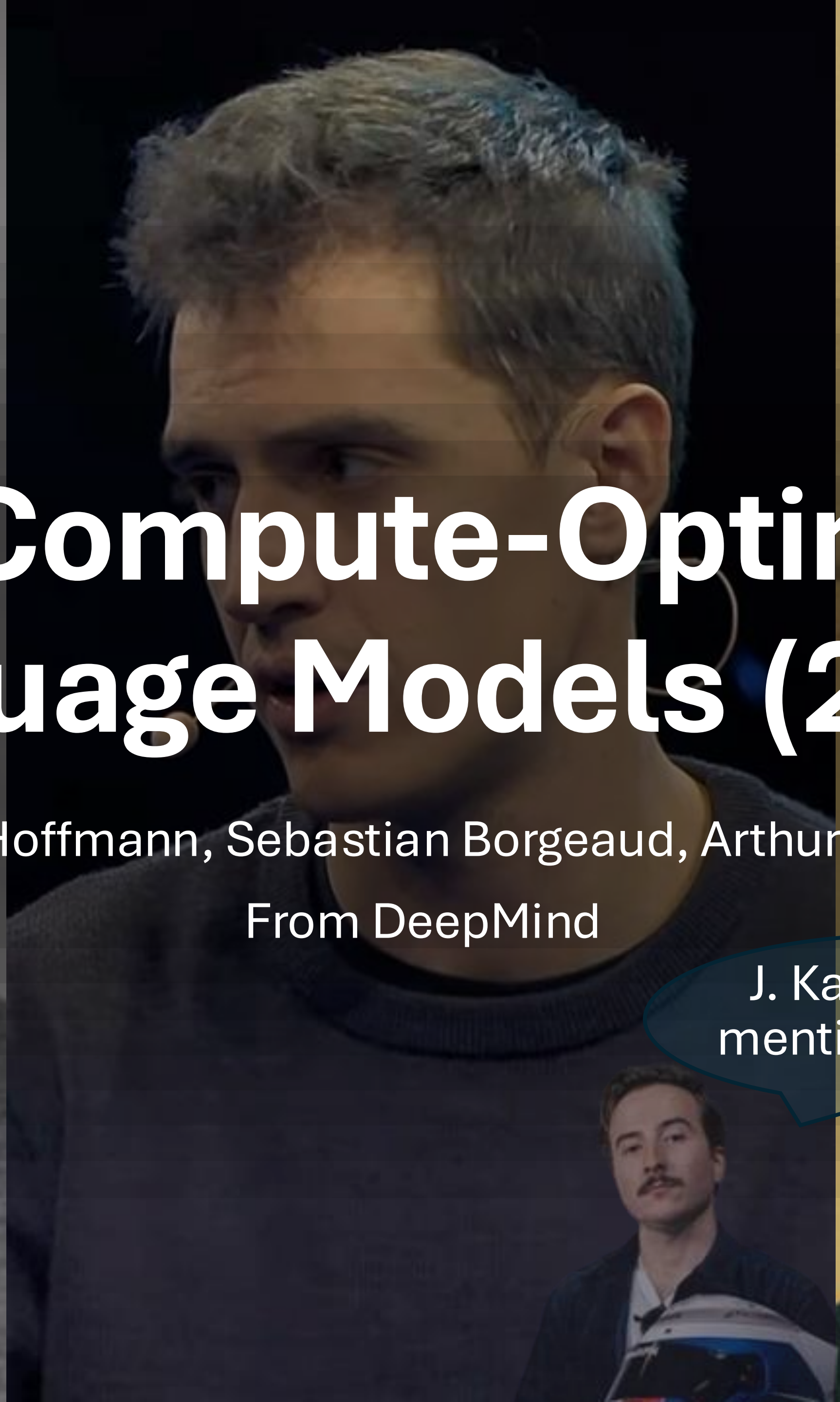
While at compute-efficient training : (for C = 2*C_min) :

$$D(C_{\rm min}) = \frac{2C_{\rm min}}{6N(C_{\rm min})} \propto C_{min}^{0,26}$$

Optimal values for the **minimal loss** theorically reachable

$$C^* \sim 10^4 \text{ PF-Days} \quad N^* \sim 10^{12} \text{ parameters},$$

$$D^* \sim 10^{12} \text{ tokens}, \quad L^* \sim 1.7 \text{ nats/token}$$


Training Compute-Optimal Large Language Models (2022)

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch

From DeepMind

J. Kaplan vous a menti. Ma réponse

Art2 - Training Compute-Optimal Large Language Models (2022)
Goal

- **Goal:** same as in the previous article:

best model size / dataset size trade-off

- **Differences:**
  - More model parameters than before:
    - art1: **<100M parameters**
    - art2: **>500M parameters**
  - Different dataset size and learning rate

  - Approach as an optimization problem: ***Minimizing L(N,D)*** under the constraint ***FLOPs(N,D) = C***

$$N_{opt}(C), D_{opt}(C) = \underset{N,D \text{ s.t. } \mathrm{FLOPs}(N,D)=C}{\operatorname{argmin}} L(N, D).$$

Art2 - Training Compute-Optimal Large Language Models (2022)

Main results – new scaling laws

- A new scaling law:

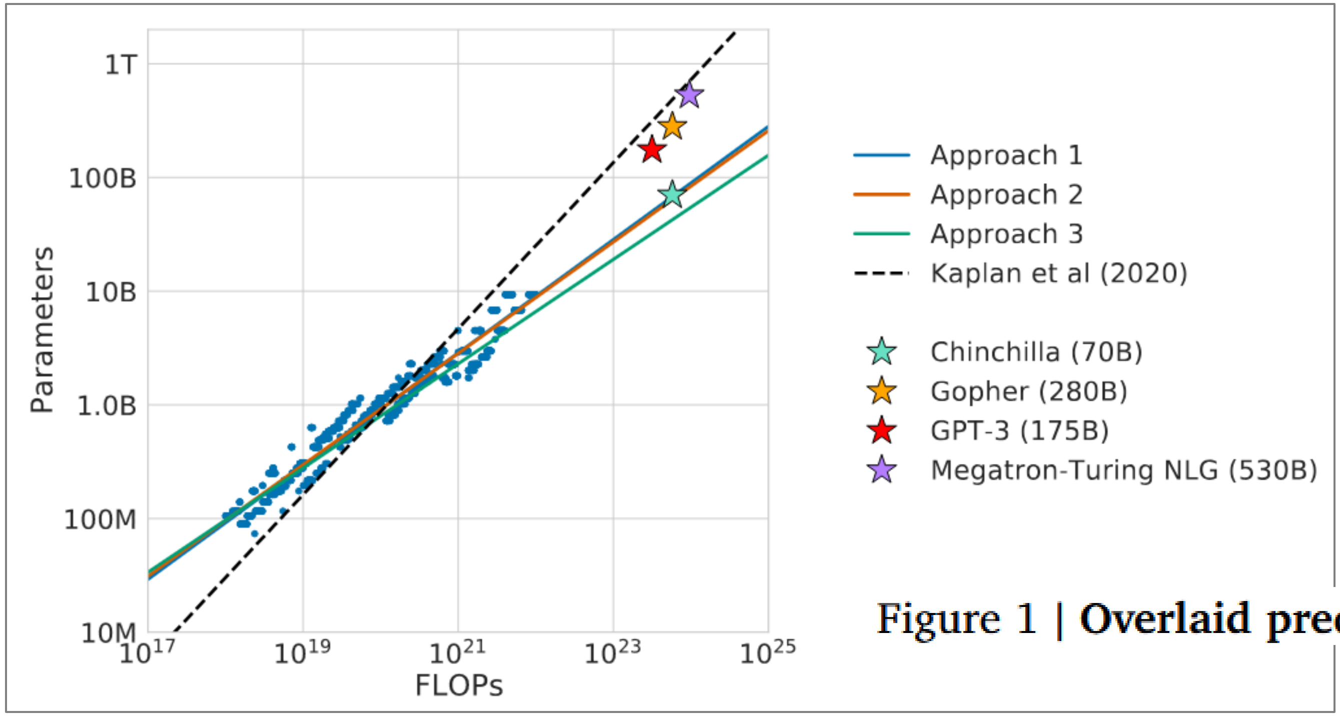


Figure 1 | Overlaid predictions.

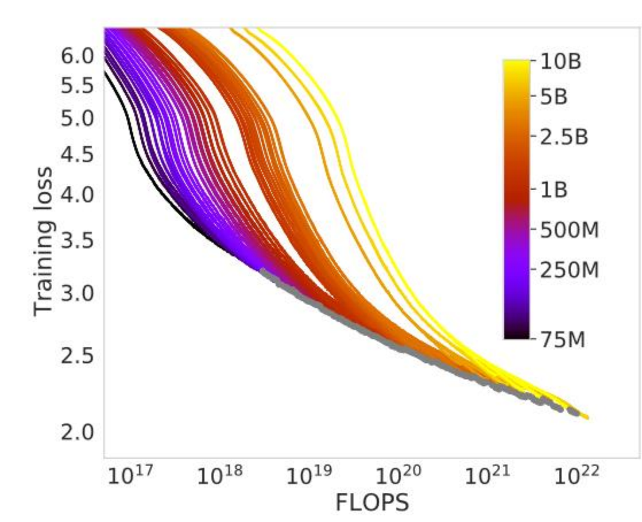
- Most of the models on market have **too much parameters compared** to **the size of there dataset**
- Disagreement with art1 : **Model size x2 → Dataset x2**

Art2 - Training Compute-Optimal Large Language Models (2022)

3 approaches

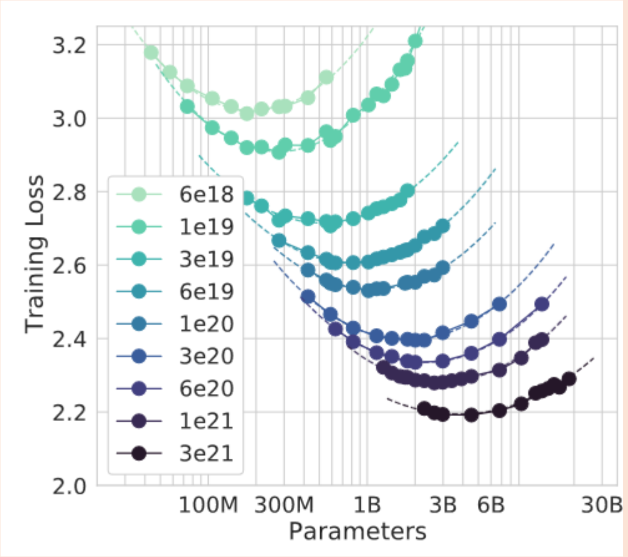
Approach 1

- Varying training sequences (4 différents sizes for each models)
- Fixed model sizes (from 70M to 10B)



Approach 2 - IsoFLOPS

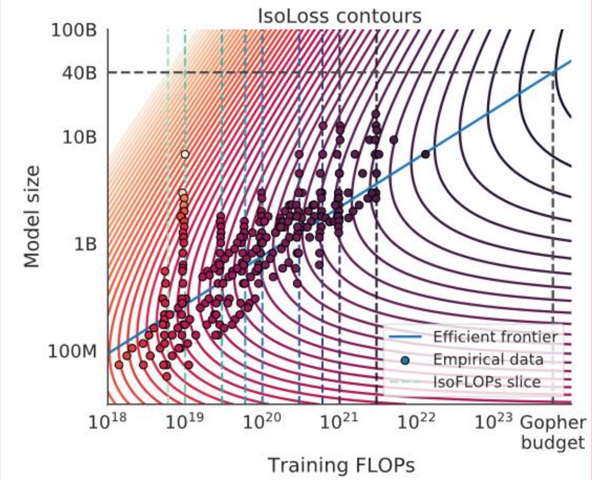
- Varying model size (up to 16B)
- Fixed FLOPS count (9 différents)



Approach 3 – Fitting law

- Fitting the law: $\hat{L}(N, D) \triangleq E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}$
- Using the loss :

$$\min_{A,B,E,\alpha,\beta} \sum_{\text{Runs } i} \text{Huber}_{\delta}\left(\log \hat{L}(N_i, D_i) - \log L_i\right)$$



Art2 - Training Compute-Optimal Large Language Models (2022)

Result : **Chinchilla**

- A small model compare to others (70B)

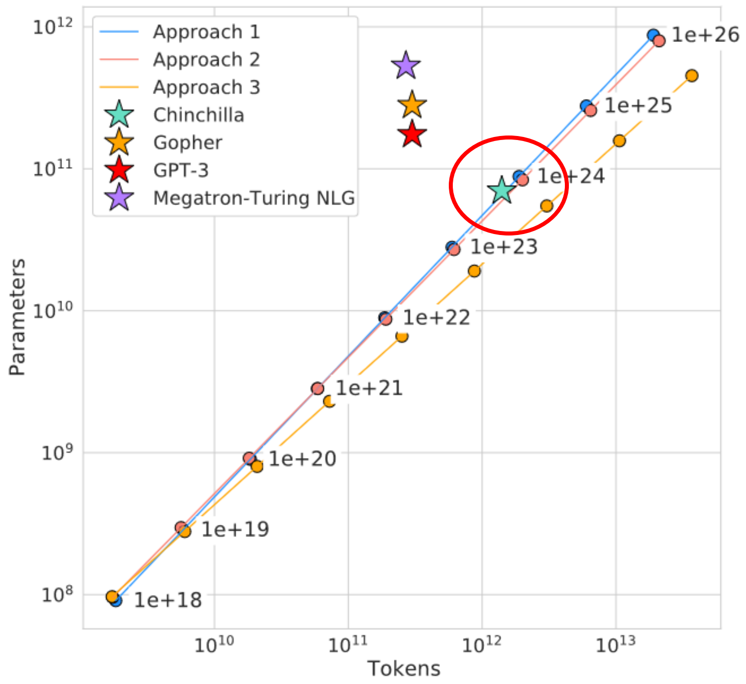


Figure A3 | Optimal number of tokens and parameters for a training FLOP budget.

- Outperform Gopher on most tasks

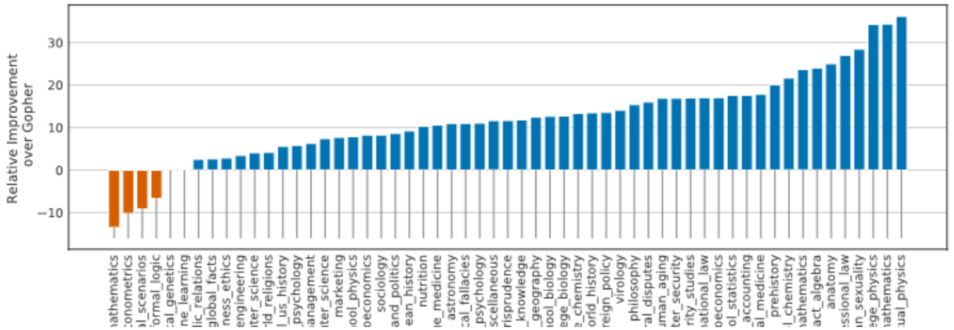


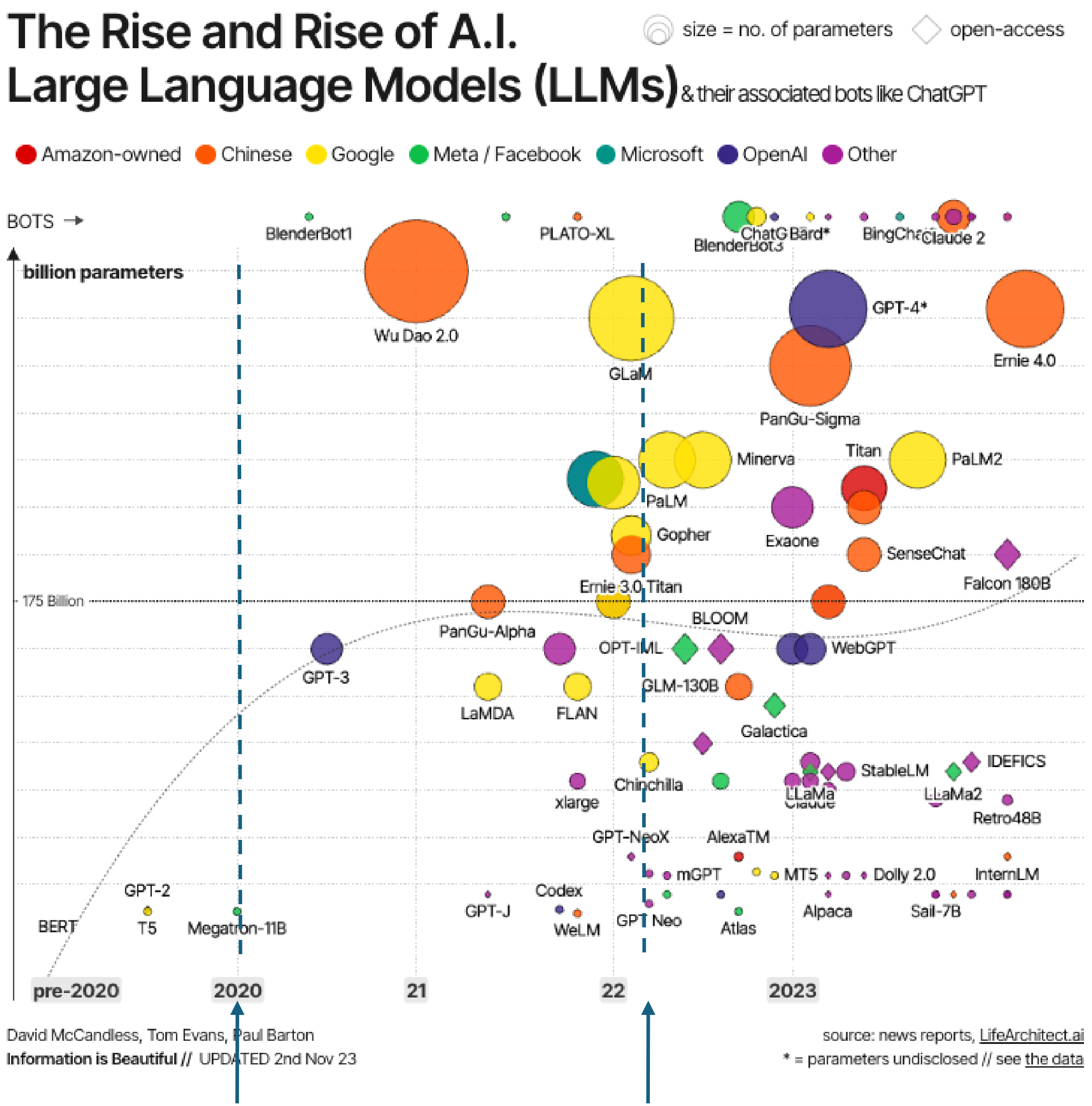
Figure 6 | MMLU results compared to *Gopher* We find that *Chinchilla* outperforms *Gopher* by 7.6% on average (see Table 6) in addition to performing better on 51/57 individual tasks, the same on

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%

Table 6 | Massive Multitask Language Understanding (MMLU).

Conclusion

- After Kaplan:
  - Tendency to increase more the model size than the dataset size
- After Chinchilla:
  - Slowed the model increase pace
  - Allocate more computation on dataset size
  - Tendency to increase the model size as much as the dataset size



Kaplan et al.

Hoffmann et al.