CSCE 689-609
Scaling Laws

Jeff Huang
jeff@cse.tamu.edu
o2lab.github.io

Given a fixed compute budget, what is the optimal model size and training dataset size for training a Transformer LM?

Let's say you can use one GPU for one day

- Would you train a 5 million parameter LM on 100 books?
- What about a 500 million parameter LM on one book?
- Or a 100k parameter LM on 5k books?

https://arxiv.org/pdf/2001.08361 2020

Scaling Laws for Neural Language Models

Jared Kaplan *
Johns Hopkins University, OpenAI
jaredk@jhu.edu

Sam McCandlish*
OpenAI
sam@openai.com

Tom Henighan
OpenAI
henighan@openai.com

Tom B. Brown
OpenAI
tom@openai.com

Benjamin Chess
OpenAI
bchess@openai.com

Rewon Child
OpenAI
rewon@openai.com

Scott Gray
OpenAI
scott@openai.com

Alec Radford
OpenAI
alec@openai.com

Jeffrey Wu
OpenAI
jeffwu@openai.com

Dario Amodei
OpenAI
damodei@openai.com

What is “Scale”?

- Compute?
- Data?
- Information?
- Effective compression of information?

BERT 2018 https://arxiv.org/pdf/1810.04805

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

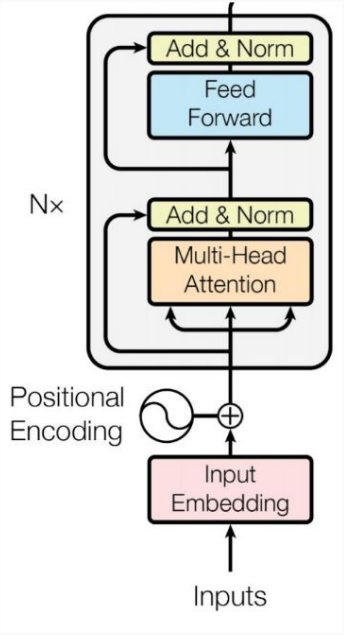
Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

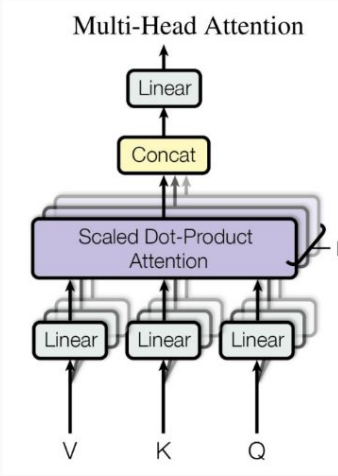
https://nlp.stanford.edu/seminar/details/jdevlin.pdf

Model Architecture

Transformer encoder

- Multi-headed self attention
  - Models context
- Feed-forward layers
  - Computes non-linear hierarchical features
- Layer norm and residuals
  - Makes training deep networks healthy
- Positional embeddings
  - Allows model to learn relative positioning





https://gluebenchmark.com/

GLUE General Language Understanding Evaluation (GLUE) benchmark

GLUE Results

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

GPT

Generative Pre-trained Transformer

GPT-2: A Big Language Model (2019)

Language Models are Unsupervised Multitask Learners

Alec Radford [*1] Jeffrey Wu [*1] Rewon Child [1] David Luan [1] Dario Amodei [**1] Ilya Sutskever [**1]

GPT: An Auto-Regressive LM (2018)

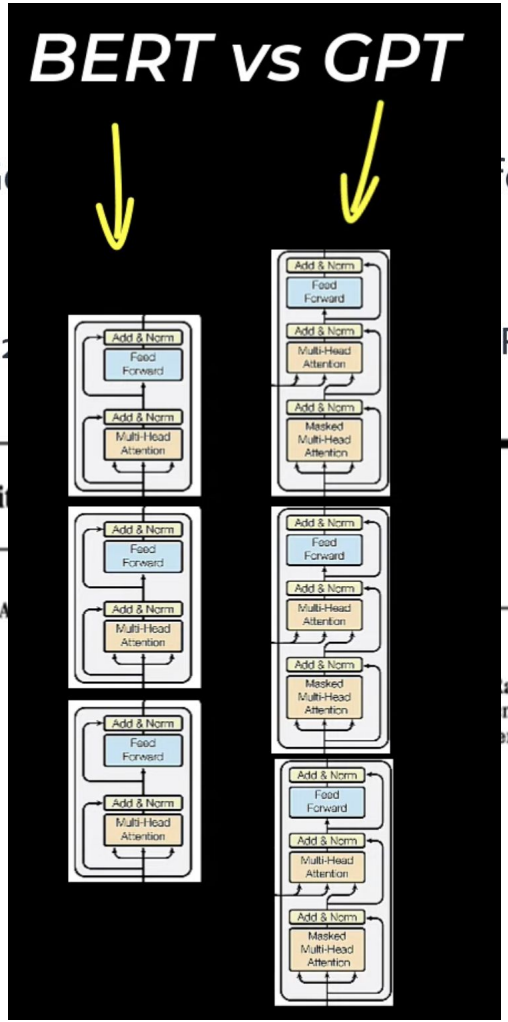
Improving Language Understanding by Generative Pre-Training

Alec Radford
OpenAI
alec@openai.com

Karthik Narasimhan
OpenAI
karthikn@openai.com

Tim Salimans
OpenAI
tim@openai.com

Ilya Sutskever
OpenAI
ilyasu@openai.com



G ormer

GPT-2: A Big Language Model (2

PT: An Auto-Regressive LM (2018)

Language Models are Unsupervised Multi

Alec Radford [*1] Jeffrey Wu [*1] Rewon Child [1] David Luan [1] Dario A

Improving Language Understanding by Generative Pre-Training

adford
nAI
enai.com

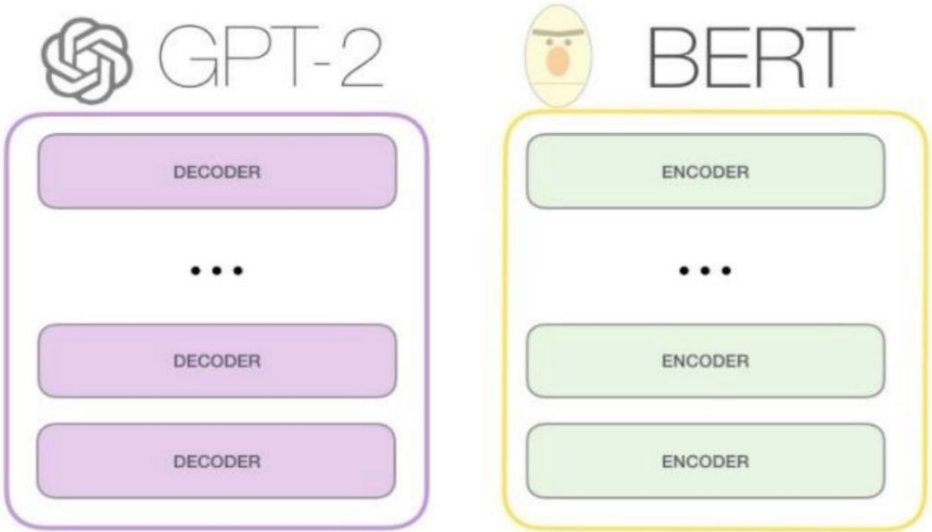
Karthik Narasimhan
OpenAI
karthikn@openai.com

Tim Salimans
OpenAI
tim@openai.com

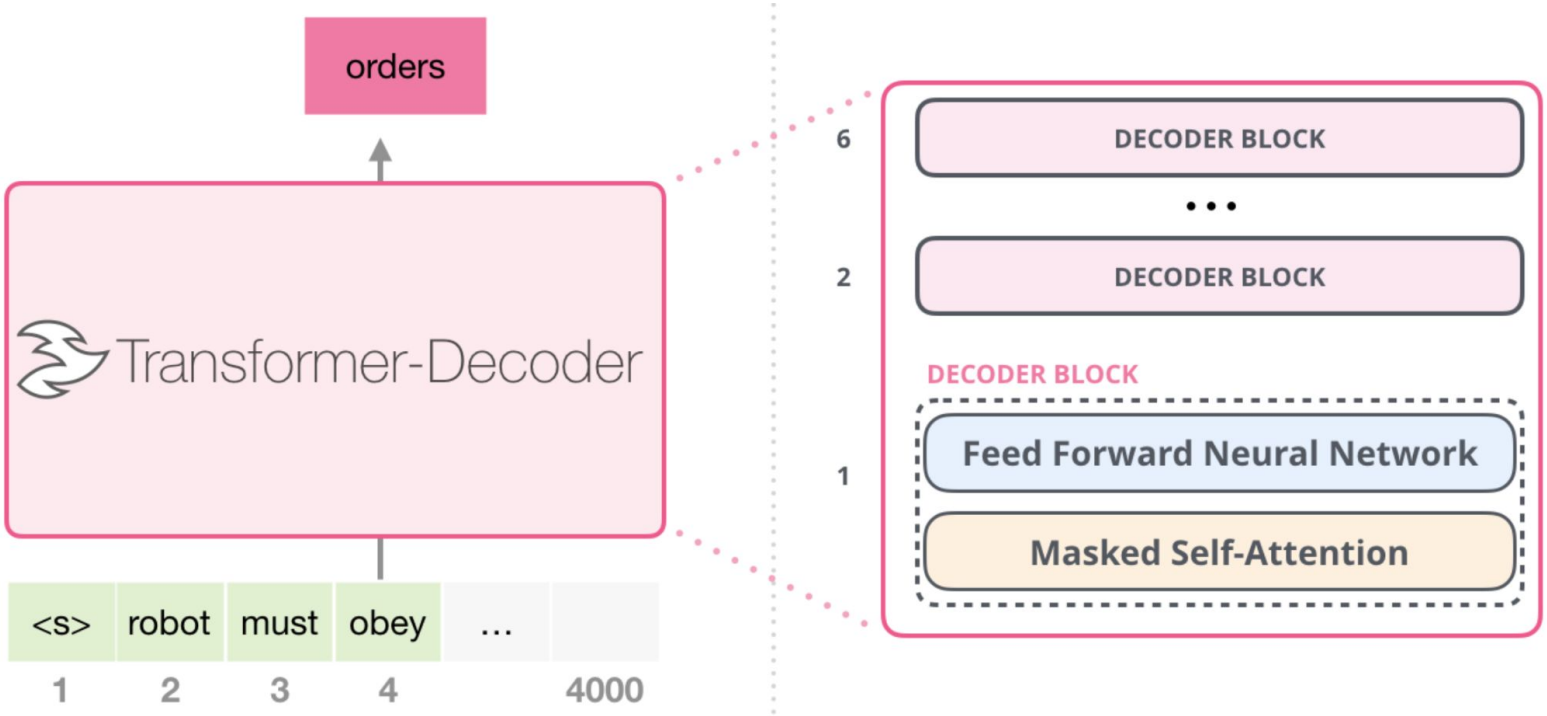
Ilya Sutskever
OpenAI
ilyasu@openai.com

GPT-2

- GPT-2 uses only Transformer Decoders (no Encoders) to generate new sequences from scratch or from a starting sequence



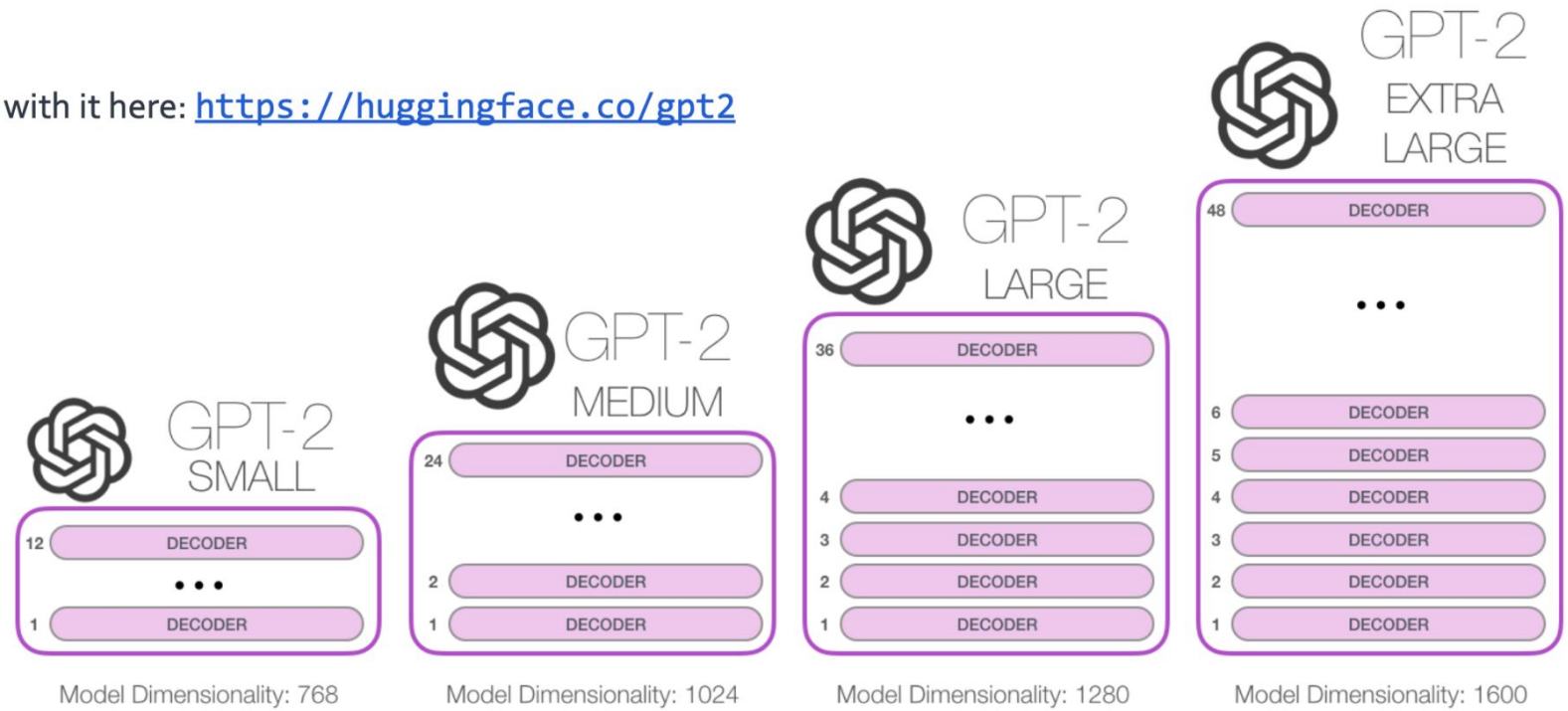
https://jalammar.github.io/illustrated-gpt2/



One key difference in the self-attention layer here, is that it masks future tokens – not by changing the word to [mask] like BERT, but by interfering in the self-attention calculation blocking information from tokens that are to the right of the position being calculated.

GPT2: Model Sizes

Play with it here: https://huggingface.co/gpt2



117M parameters 345M 762M 1542M

GPT2: Some Results

Language Models are Unsupervised Multitask Learners

	LAMBADA (PPL)	LAMBADA (ACC)	CBT-CN (ACC)	CBT-NE (ACC)	WikiText2 (PPL)	PTB (PPL)	enwik8 (BPB)	text8 (BPC)	WikiText103 (PPL)	1BW (PPL)
SOTA	99.8	56.25	85.7	82.3	39.14	46.54	0.99	1.08	18.3	**21.8**
117M	**35.13**	45.99	**87.65**	**83.4**	**29.41**	65.85	1.16	1.17	37.50	75.20
345M	**15.60**	55.48	**92.35**	**87.1**	**22.76**	47.33	1.01	**1.06**	26.37	55.72
762M	**10.87**	**60.12**	**93.45**	**88.0**	**19.93**	**40.31**	**0.97**	**1.02**	22.05	44.575
1542M	**8.63**	**63.24**	**93.30**	**89.05**	**18.34**	**35.76**	**0.93**	**0.98**	**17.48**	42.16

Table 3. Zero-shot results on many datasets. No training or fine-tuning was performed for any of these results. PTB and WikiText-2 results are from (Gong et al., 2018). Other language model results are from (Dai et al., 2019).

GPT-3: A Very Large Language Model (2020)

- More layers & parameters
- Bigger dataset
- Longer training
- Larger embedding/hidden dimension
- Larger context window

https://the-decoder.com/gpt-4-architecture-datasets-costs-and-more-leaked/

GPT 3.5, ChatGPT, GPT-4 …

- **GPT-4's Scale**: GPT-4 has ~1.8 trillion parameters across 120 layers, which is over 10 times larger than GPT-3.
- **Mixture Of Experts (MoE)**: OpenAI utilizes 16 experts within their model, each with ~111B parameters for MLP. Two of these experts are routed per forward pass, which contributes to keeping costs manageable.
- **Dataset**: GPT-4 is trained on ~13T tokens, including both text-based and code-based data, with some fine-tuning data from ScaleAI and internally.
- **Dataset Mixture**: The training data included CommonCrawl & RefinedWeb, totaling 13T tokens. Speculation suggests additional sources like Twitter, Reddit, YouTube, and a large collection of textbooks.
- **Training Cost**: The training costs for GPT-4 was around $63 million, taking into account the computational power required and the time of training.
- **Inference Cost**: GPT-4 costs 3 times more than the 175B parameter Davinci, due to the larger clusters required and lower utilization rates.
- **Inference Architecture**: The inference runs on a cluster of 128 GPUs, using 8-way tensor parallelism and 16-way pipeline parallelism.

https://arxiv.org/pdf/2302.13971

LLaMA: Open and Efficient Foundation Language Models

Hugo Touvron[*], **Thibaut Lavril**[*], **Gautier Izacard**[*], **Xavier Martinet**
Marie-Anne Lachaux, Timothee Lacroix, Baptiste Rozière, Naman Goyal
Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin
Edouard Grave[*], **Guillaume Lample**[*]

Meta AI

Abstract

We introduce LLaMA, a collection of foundation language models ranging from 7B to 65B parameters. We train our models on trillions of tokens, and show that it is possible to train state-of-the-art models using publicly available datasets exclusively, without resorting to proprietary and inaccessible datasets. In particular, LLaMA-13B outperforms GPT-3 (175B) on most benchmarks, and LLaMA-65B is competitive with the best models, Chinchilla-70B and PaLM-540B. We release all our models to the research community[1].

performance, a smaller one trained longer will ultimately be cheaper at inference. For instance, although Hoffmann et al. (2022) recommends training a 10B model on 200B tokens, we find that the performance of a 7B model continues to improve even after 1T tokens.

The focus of this work is to train a series of language models that achieve the best possible performance at various inference budgets, by training on more tokens than what is typically used. The resulting models, called *LLaMA*, ranges from 7B to 65B parameters with competitive performance

https://towardsdatascience.com/the-evolution-of-llama-from-llama-1-to-llama-3-1-13c4ebe96258

Original Transformer vs Llama 1 architecture

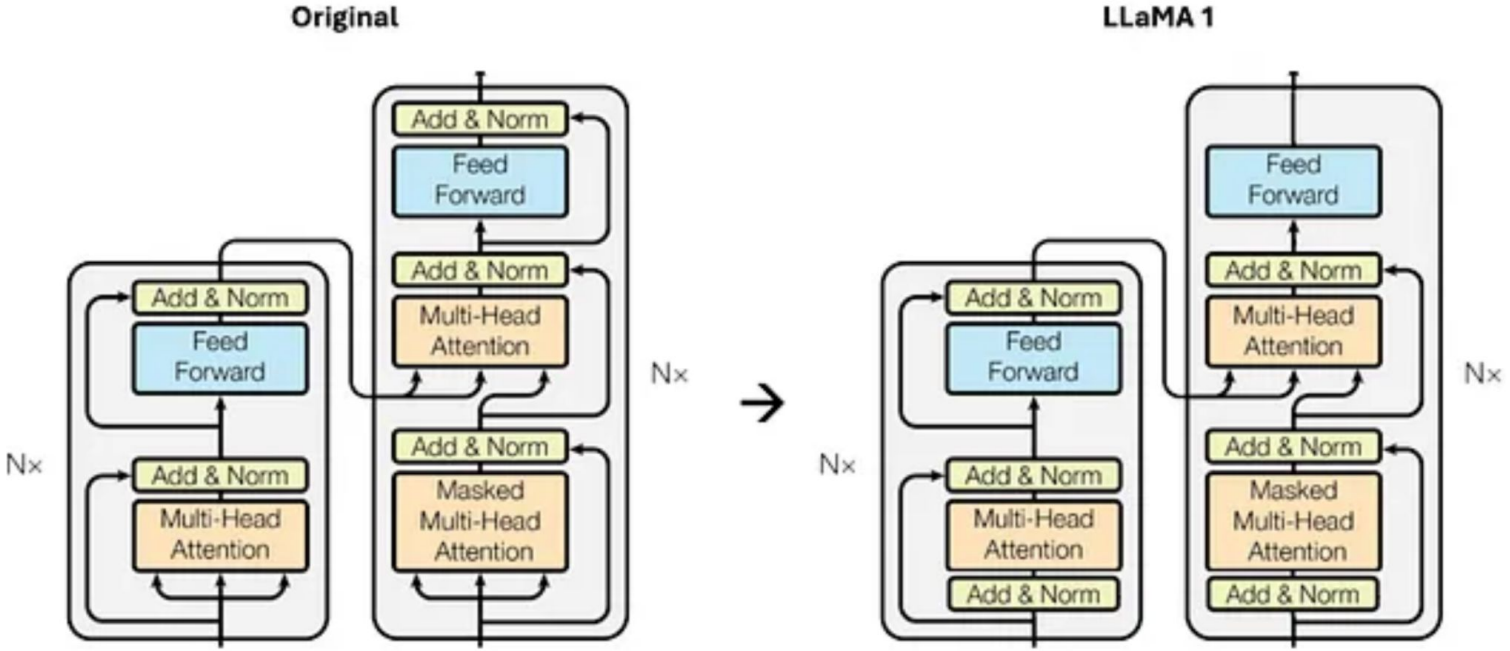


Figure 2: Differences between original and Llama 1 architectures where each input in the sub-layer transformer is normalized (image by author)

https://towardsdatascience.com/the-evolution-of-llama-from-llama-1-to-llama-3-1-13c4ebe96258

Original Transformer vs Llama 1 (other changes)

1. LayerNorm vs RMSNorm
2. ReLU vs SwiGLU activation function
3. Absolute Positional Embeddings vs RoPE (Rotary positional embeddings)

https://arxiv.org/pdf/2307.09288

Llama 2: Open Foundation and Fine-Tuned Chat Models

Hugo Touvron* Louis Martin† Kevin Stone†
Peter Albert Amjad Almahairi Yasmine Babaei Nikolay Bashlykov Soumya Batra
Prajjwal Bhargava Shruti Bhosale Dan Bikel Lukas Blecher Cristian Canton Ferrer Moya Chen
Guillem Cucurull David Esiobu Jude Fernandes Jeremy Fu Wenyin Fu Brian Fuller
Cynthia Gao Vedanuj Goswami Naman Goyal Anthony Hartshorn Saghar Hosseini Rui Hou
Hakan Inan Marcin Kardas Viktor Kerkez Madian Khabsa Isabel Kloumann Artem Korenev
Punit Singh Koura Marie-Anne Lachaux Thibaut Lavril Jenya Lee Diana Liskovich
Yinghai Lu Yuning Mao Xavier Martinet Todor Mihaylov Pushkar Mishra
Igor Molybog Yixin Nie Andrew Poulton Jeremy Reizenstein Rashi Rungta Kalyan Saladi
Alan Schelten Ruan Silva Eric Michael Smith Ranjan Subramanian Xiaoqing Ellen Tan Binh Tang
Ross Taylor Adina Williams Jian Xiang Kuan Puxin Xu Zheng Yan Iliyan Zarov Yuchen Zhang
Angela Fan Melanie Kambadur Sharan Narang Aurelien Rodriguez Robert Stojnic
Sergey Edunov Thomas Scialom*

GenAI, Meta

Abstract

In this work, we develop and release Llama 2, a collection of pretrained and fine-tuned large language models (LLMs) ranging in scale from 7 billion to 70 billion parameters. Our fine-tuned LLMs, called Llama 2-Chat, are optimized for dialogue use cases. Our models outperform open-source chat models on most benchmarks we tested, and based on our human evaluations for helpfulness and safety, may be a suitable substitute for closed-source models. We provide a detailed description of our approach to fine-tuning and safety improvements of Llama 2-Chat in order to enable the community to build on our work and contribute to the responsible development of LLMs.

Llama 2: The Evolved Form of Llama 1

Llama 2 [8] kept all the architecture changes made to the original Transformer architecture on Llama 1. Additionally, it increased the context length from 2048 to 4096 and replaced Multi-Head Attention (MHA) [9] with Grouped-Query Attention (GQA) [10] for the larger models (34B and 70B).

The Llama 3 Herd of Models

Llama Team, AI @ Meta[1]
[1]A detailed contributor list can be found in the appendix of this paper.

Modern artificial intelligence (AI) systems are powered by foundation models. This paper presents a new set of foundation models, called Llama 3. It is a herd of language models that natively support multilinguality, coding, reasoning, and tool usage. Our largest model is a dense Transformer with 405B parameters and a context window of up to 128K tokens. This paper presents an extensive empirical evaluation of Llama 3. We find that Llama 3 delivers comparable quality to leading language models such as GPT-4 on a plethora of tasks. We publicly release Llama 3, including pre-trained and post-trained versions of the 405B parameter language model and our Llama Guard 3 model for input and output safety. The paper also presents the results of experiments in which we integrate image, video, and speech capabilities into Llama 3 via a compositional approach. We observe this approach performs competitively with the state-of-the-art on image, video, and speech recognition tasks. The resulting models are not yet being broadly released as they are still under development.

Date: July 23, 2024

Llama 3: Size and Tokenization

Llama 3 [11] increased the context length from 4096 to 8192 and extended the GQA to the smaller model (8B). Besides that, the authors replaced the tokenizer Sentence Piece [12] with the TikToken [13] used in OpenAI models. It significantly improved model performance since it has a vocabulary size of 128k tokens instead of 32k.

Llama 3.1: The Latest (and Biggest) Release

Released in July 2024, Llama 3.1 introduces a significant leap in context length (128K tokens) and support for eight additional languages. One of the key pieces of the release was the larger model Llama 3.1 405B. Until then, open LLMs were generally released in sizes below 100B.

	Llama 1	Llama 2	Llama 3	Llama 3.1
Context Length	2048	4096	8192	128k
Vocabulary Size	32k	32k	128k	128k
Training data size	1.4T tokens	2T tokens	15T tokens	15T tokens
Language Support	1 Language	1 Language	1 Language	9 Languages

Table 1: Comparing Llama evolution regarding context length, vocabulary size, training data size, and languages they support.

Is Scale All You Need?

- Compute?
- Data?
- Information?
- Effective compression of information?

Scaling Laws for Neural Language Models

https://arxiv.org/pdf/2001.08361

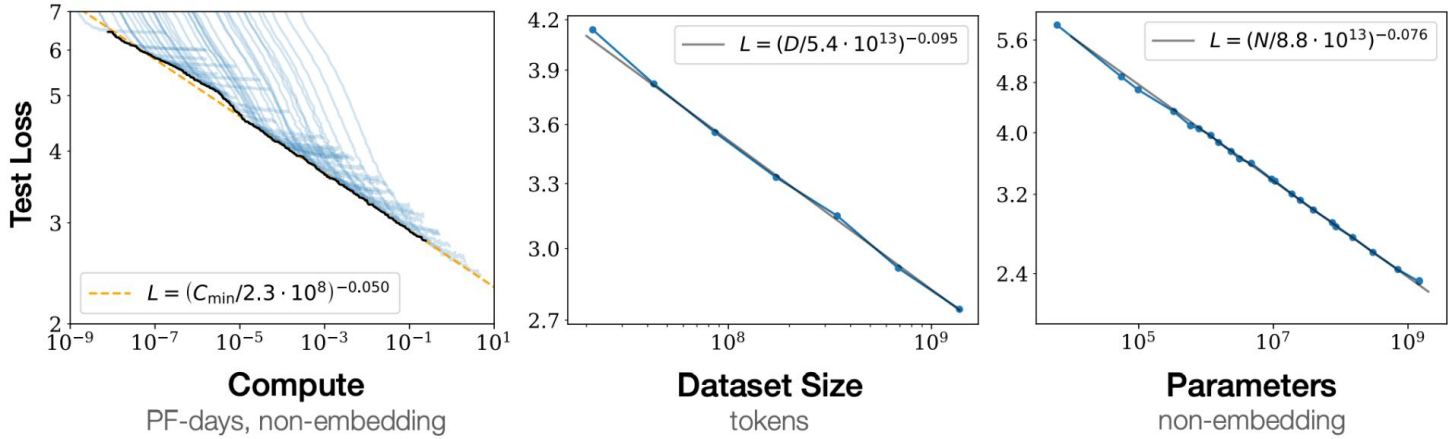


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

Performance depends strongly on scale, weakly on model shape: Model performance depends most strongly on scale, which consists of three factors: the number of model parameters N (excluding embeddings), the size of the dataset D, and the amount of compute C used for training. Within reasonable limits, performance depends very weakly on other architectural hyperparameters such as depth vs. width. (Section 3)

Scaling Laws for Neural Language Models

https://arxiv.org/pdf/2001.08361

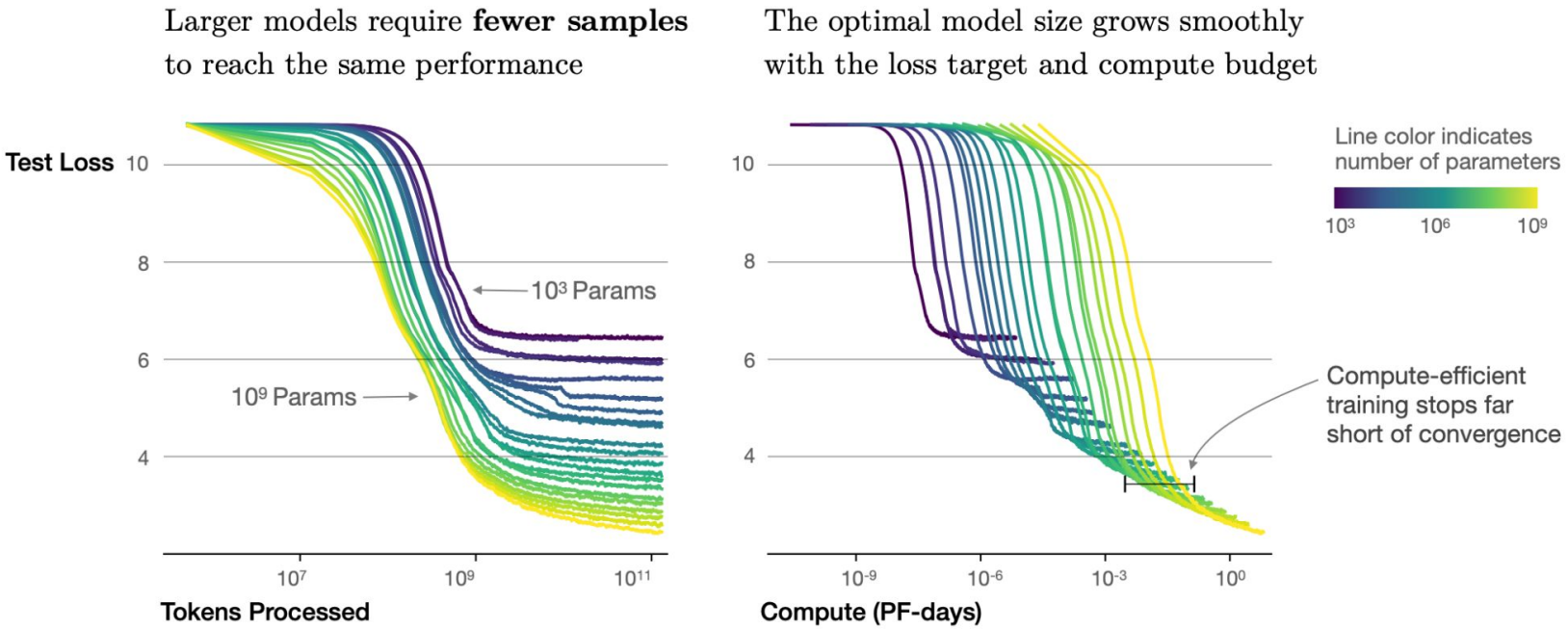


Figure 2 We show a series of language model training runs, with models ranging in size from 10^3 to 10^9 parameters (excluding embeddings).

Scaling Laws for Neural Language Models

- L – the cross entropy loss in nats. Typically it will be averaged over the tokens in a context, but in some cases we report the loss for specific tokens within the context.
- N – the number of model parameters, *excluding all vocabulary and positional embeddings*
- $C \approx 6NBS$ – an estimate of the total non-embedding training compute, where B is the batch size, and S is the number of training steps (ie parameter updates). We quote numerical values in PF-days, where one PF-day $= 10^{15} \times 24 \times 3600 = 8.64 \times 10^{19}$ floating point operations.
- D – the dataset size in tokens
- B_{crit} – the critical batch size [MKAT18], defined and discussed in Section 5.1. Training at the critical batch size provides a roughly optimal compromise between time and compute efficiency.
- C_{min} – an estimate of the minimum amount of non-embedding compute to reach a given value of the loss. This is the training compute that would be used if the model were trained at a batch size much less than the critical batch size.
- S_{min} – an estimate of the minimal number of training steps needed to reach a given value of the loss. This is also the number of training steps that would be used if the model were trained at a batch size much greater than the critical batch size.
- α_X – power-law exponents for the scaling of the loss as $L(X) \propto 1/X^{\alpha_X}$ where X can be any of $N, D, C, S, B, C^{\text{min}}$.

Scaling Laws for Neural Language Models

The optimal parameters for compute efficient training are given by:

Compute-Efficient Value	Power Law	Scale
$N_{\text{opt}} = N_e \cdot C_{\text{min}}^{p_N}$	$p_N = 0.73$	$N_e = 1.3 \cdot 10^9$ params
$B \ll B_{\text{crit}} = \frac{B_*}{L^{1/\alpha_B}} = B_e C_{\text{min}}^{p_B}$	$p_B = 0.24$	$B_e = 2.0 \cdot 10^6$ tokens
$S_{\text{min}} = S_e \cdot C_{\text{min}}^{p_S}$ (lower bound)	$p_S = 0.03$	$S_e = 5.4 \cdot 10^3$ steps
$D_{\text{opt}} = D_e \cdot C_{\text{min}}^{p_D}$ (1 epoch)	$p_D = 0.27$	$D_e = 2 \cdot 10^{10}$ tokens

https://people.cs.umass.edu/~miyyer/cs685_s23/slides/scaling_laws.pdf

Some potential issues in the scaling laws paper

- Used same learning rate schedule for all training runs, regardless of how many training tokens / batches

- This schedule needs to be adjusted based on the number of training steps; otherwise, it can impair performance

- The resulting “scaling laws” from Kaplan et al., are flawed because of this!

Chinchilla paper

https://arxiv.org/pdf/2203.15556

2022

DeepMind

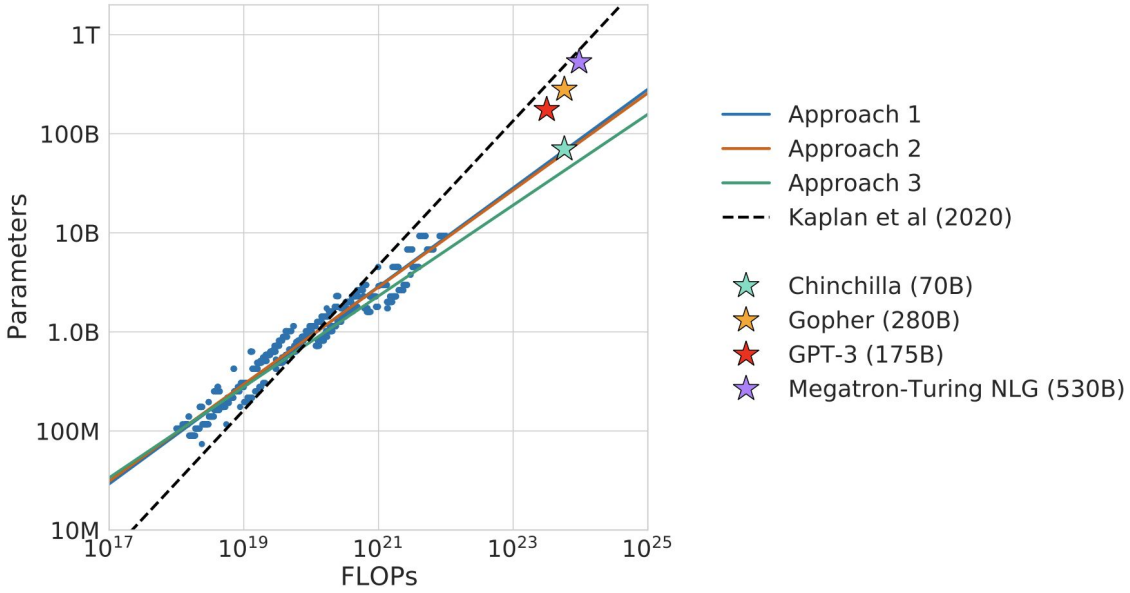
Training Compute-Optimal Large Language Models

Jordan Hoffmann*, Sebastian Borgeaud*, Arthur Mensch*, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals and Laurent Sifre*

*Equal contributions

Chinchilla paper

Our approach leads to considerably different results than that of Kaplan et al. (2020). We highlight our results in Figure 1



Chinchilla paper

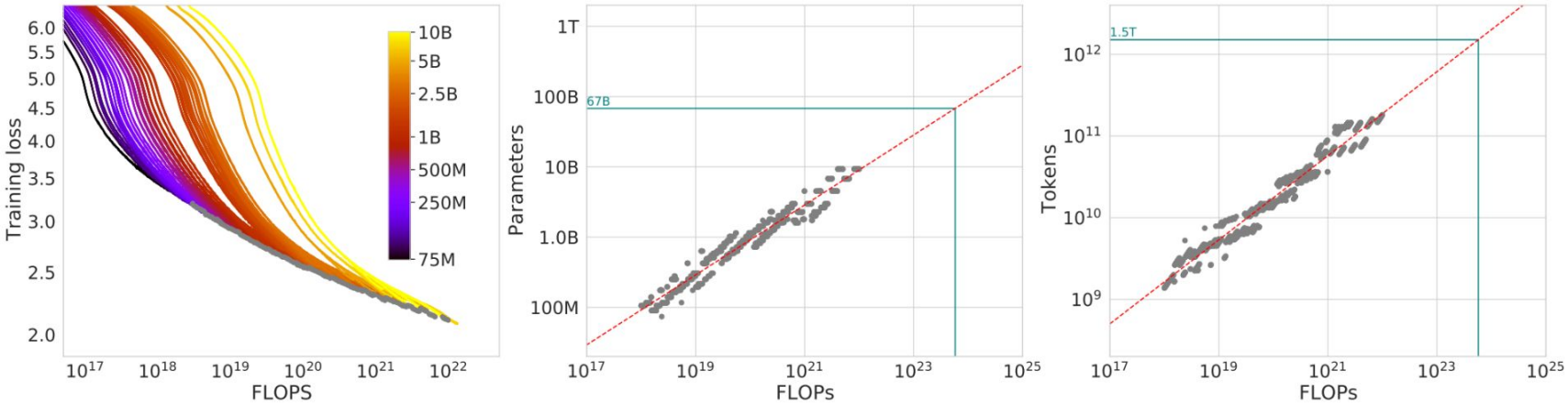
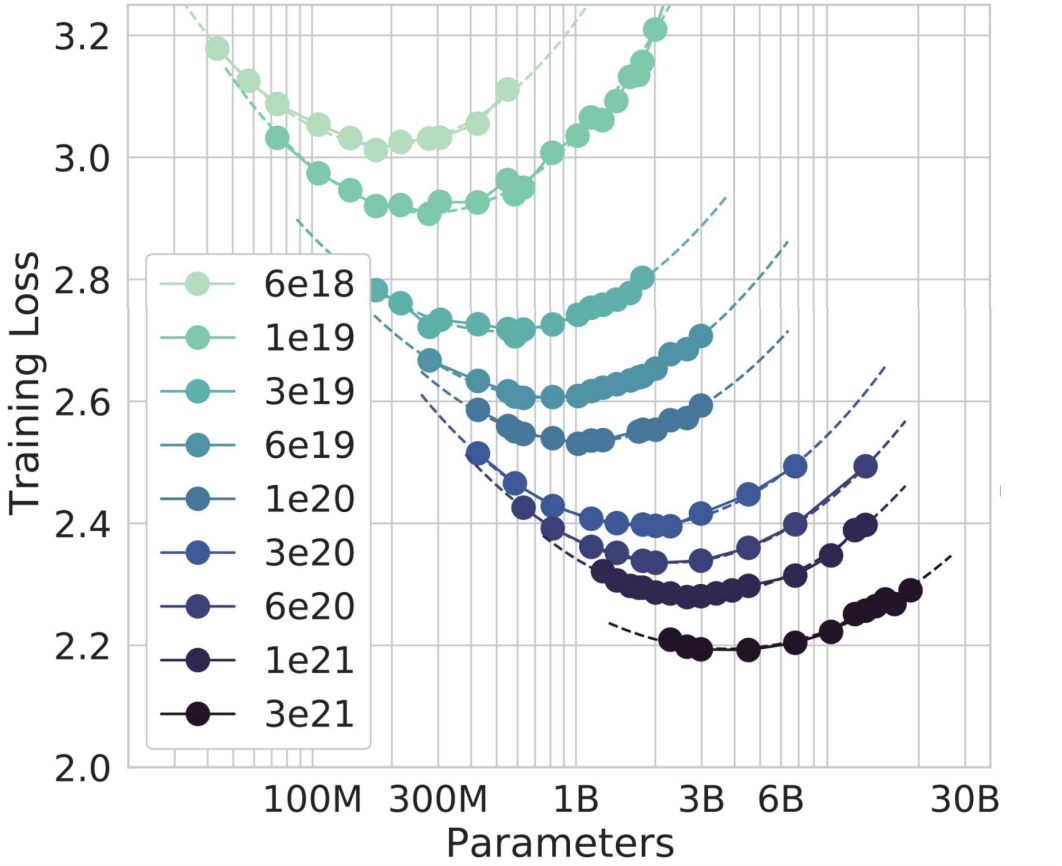


Figure 2 | **Training curve envelope.** On the **left** we show all of our different runs. We launched a range of model sizes going from 70M to 10B, each for four different cosine cycle lengths. From these curves, we extracted the envelope of minimal loss per FLOP, and we used these points to estimate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

Chinchilla paper



Chinchilla paper

Table 2 | **Estimated parameter and data scaling with increased training compute.** The listed values are the exponents, a and b, on the relationship $N_{opt} \propto C^a$ and $D_{opt} \propto C^b$. Our analysis suggests a near equal scaling in parameters and data with increasing compute which is in clear contrast to previous work on the scaling of large models. The 10^{th} and 90^{th} percentiles are estimated via bootstrapping data (80% of the dataset is sampled 100 times) and are shown in parenthesis.

Approach	Coeff. a where $N_{opt} \propto C^a$	Coeff. b where $D_{opt} \propto C^b$
1. Minimum over training curves	0.50 (0.488, 0.502)	0.50 (0.501, 0.512)
2. IsoFLOP profiles	0.49 (0.462, 0.534)	0.51 (0.483, 0.529)
3. Parametric modelling of the loss	0.46 (0.454, 0.455)	0.54 (0.542, 0.543)
Kaplan et al. (2020)	0.73	0.27

Chinchilla paper

Table 3 | **Estimated optimal training FLOPs and training tokens for various model sizes.** For various model sizes, we show the projections from Approach 1 of how many FLOPs and training tokens would be needed to train compute-optimal models. The estimates for Approach 2 & 3 are similar (shown in Section D.3)

Parameters	FLOPs	FLOPs (in *Gopher* unit)	Tokens
400 Million	1.92e+19	1/29,968	8.0 Billion
1 Billion	1.21e+20	1/4,761	20.2 Billion
10 Billion	1.23e+22	1/46	205.1 Billion
67 Billion	5.76e+23	1	1.5 Trillion
175 Billion	3.85e+24	6.7	3.7 Trillion
280 Billion	9.90e+24	17.2	5.9 Trillion
520 Billion	3.43e+25	59.5	11.0 Trillion
1 Trillion	1.27e+26	221.3	21.2 Trillion
10 Trillion	1.30e+28	22515.9	216.2 Trillion

Chinchilla paper

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

Chinchilla paper

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).** We report the average 5-shot accuracy over 57 tasks with model and human accuracy comparisons taken from Hendrycks et al. (2020). We also include the average prediction for state of the art accuracy in June 2022/2023 made by 73 competitive human forecasters in Steinhardt (2021).

Quick takeaways

- **Kaplan et al., 2020**: if you're able to increase your compute budget, you should prioritize increasing model size over data size
  - With a 10x compute increase, you should increase model size by 5x and data size by 2x
  - With a 100x compute increase, model size 25x and data 4x
- **Hoffmann et al., 2022**: you should increase model and data size at the same rate
  - With a 10x compute increase, you should increase both model size and data size by 3.1x
  - With a 100x compute increase, both model and data size 10x

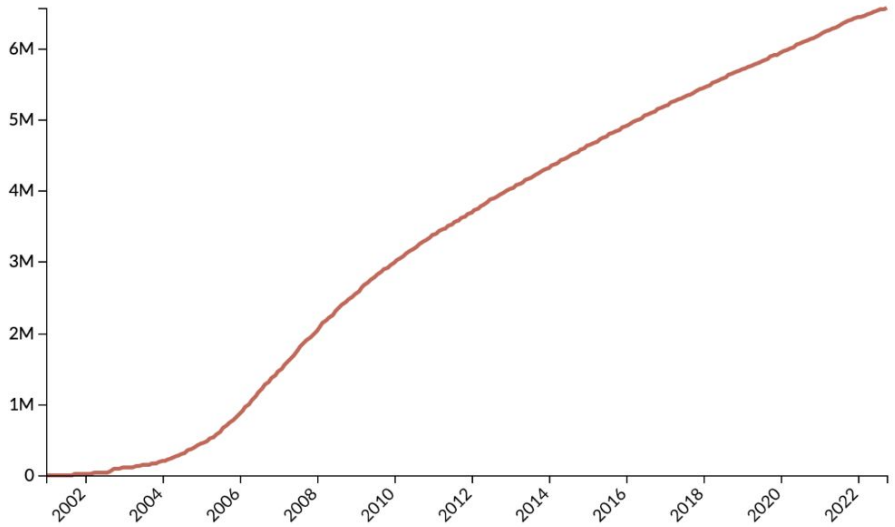
https://jhu-intro-hlt.github.io/slides/scaling-llms-2023.pdf

Is Scale All You Need?

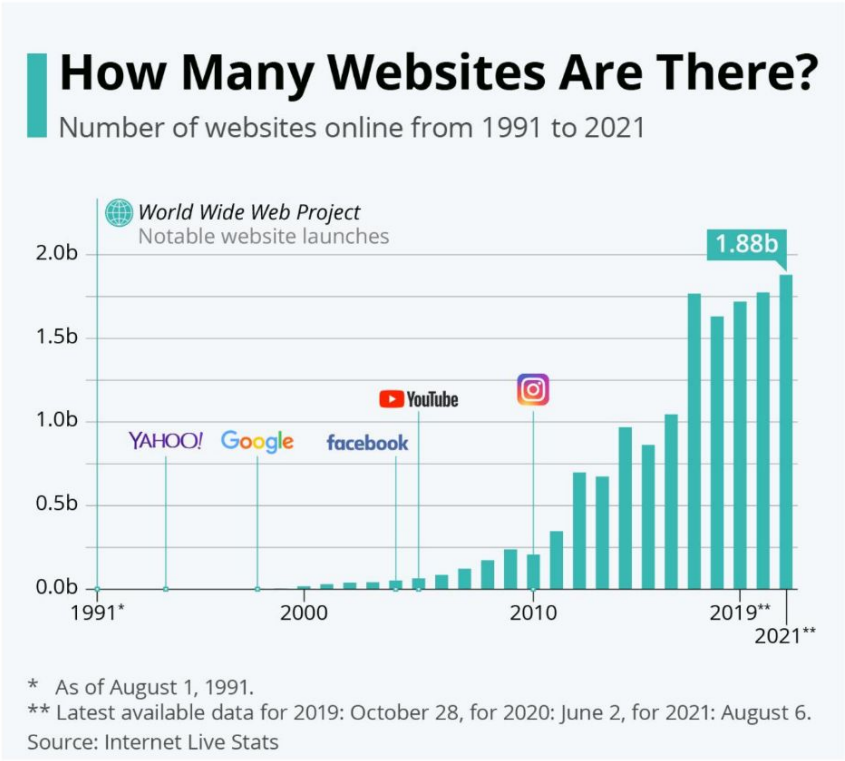
- Compute?
- Data?
- Information?
- Effective compression of information?

Is Scale All You Need?

- Data is growing exponentially (?)



Wikipedia size

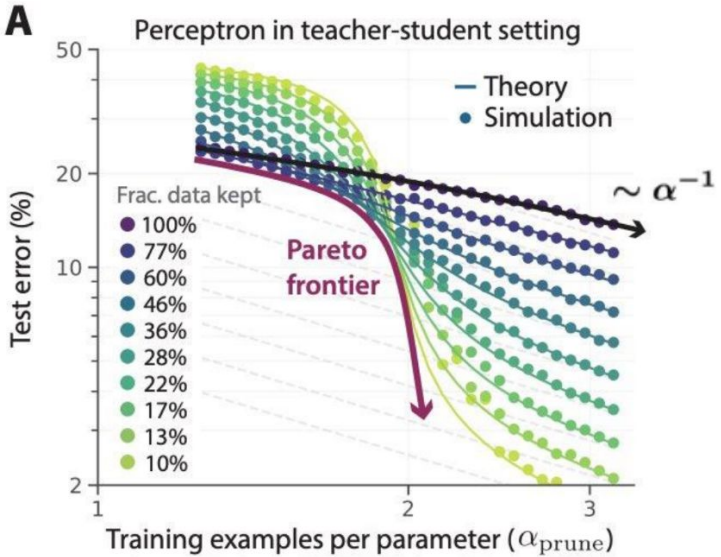


Is Scale All You Need?

- You can harness data from other modalities.
  - For example, to get more text data we can build a solid speech processor model that converts speech to text.
  - (aside: more than 80% if internet traffic is video)

Is Scale All You Need?

- You can use data more effectively.
- Sorscher et al. lays out recipes to achieve *exponential* scaling instead through statistical mechanics theory.
- Carefully curating a small subset goes a long way!



[Beyond neural scaling laws: beating power law scaling via data pruning. Sorscher+ 2022]

How does the future look like to you?

Which future will we have?

1. One very large model
2. Few very large models
3. Many vey large models

Important Notes

- Read:
  - **Reproducing the GPT-2 (124M) model**. https://github.com/karpathy/llm.c
  - https://github.com/karpathy/llm.c/discussions/481

- Due
  - HW1 (Saturday)