10-423/10-623 Generative AI

Machine Learning Department
School of Computer Science
Carnegie Mellon University

Querying Transformer + Scaling Laws

Aran Nayebi & Matt Gormley
Lecture 15
Oct. 22, 2025
Scaling slides credit: Pat Virtue

Reminders

- Homework 3: Applying and Adapting LLMs
  - Due: Tomorrow! (Thu, Oct 23 at 11:59pm)
- Quiz 4
  - In-class: Mon, Oct 27
  - Lectures 12 (text2image part) - 15
- Homework 4: Multimodal Foundation Models
  - Out: Thu, Oct 23
  - Due: Mon, Nov 3 at 11:59pm

QUERYING TRANSFORMER

Recall: PaliGemma

- SigLIP is a variant of CLI
- Gemma is a 2B LLM (open source counterpart to Gemini)
- Linear projection is creating image embeddings, but they live in the same high dimensional space as th word embeddings
- The LLM has to figure out how to use the image embeddings in some useful way so as t maximize the likelihood of the correct text response

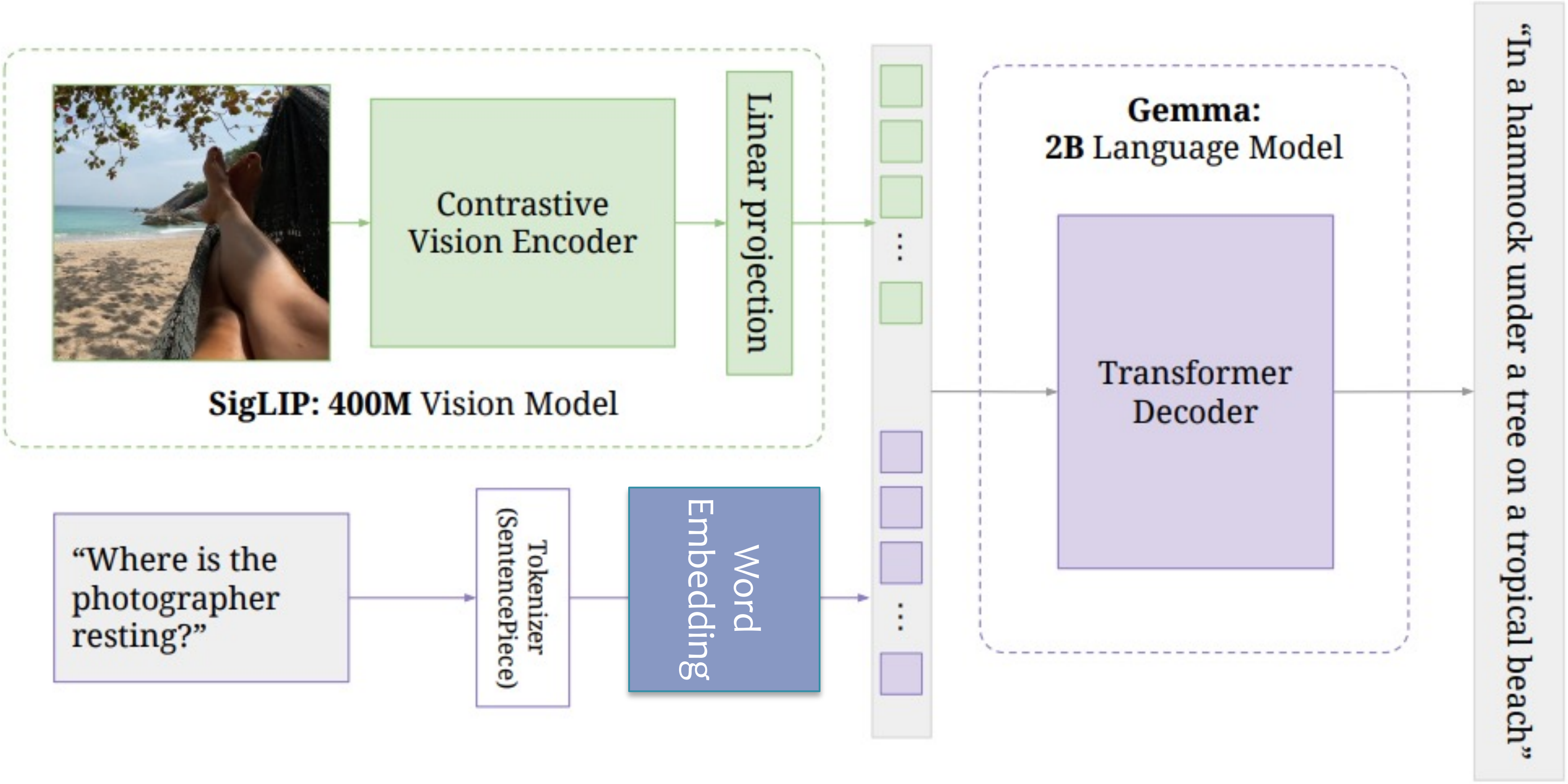


Figure from https://arxiv.org/pdf/2407.07726

Querying Transformer (BLIP-2)

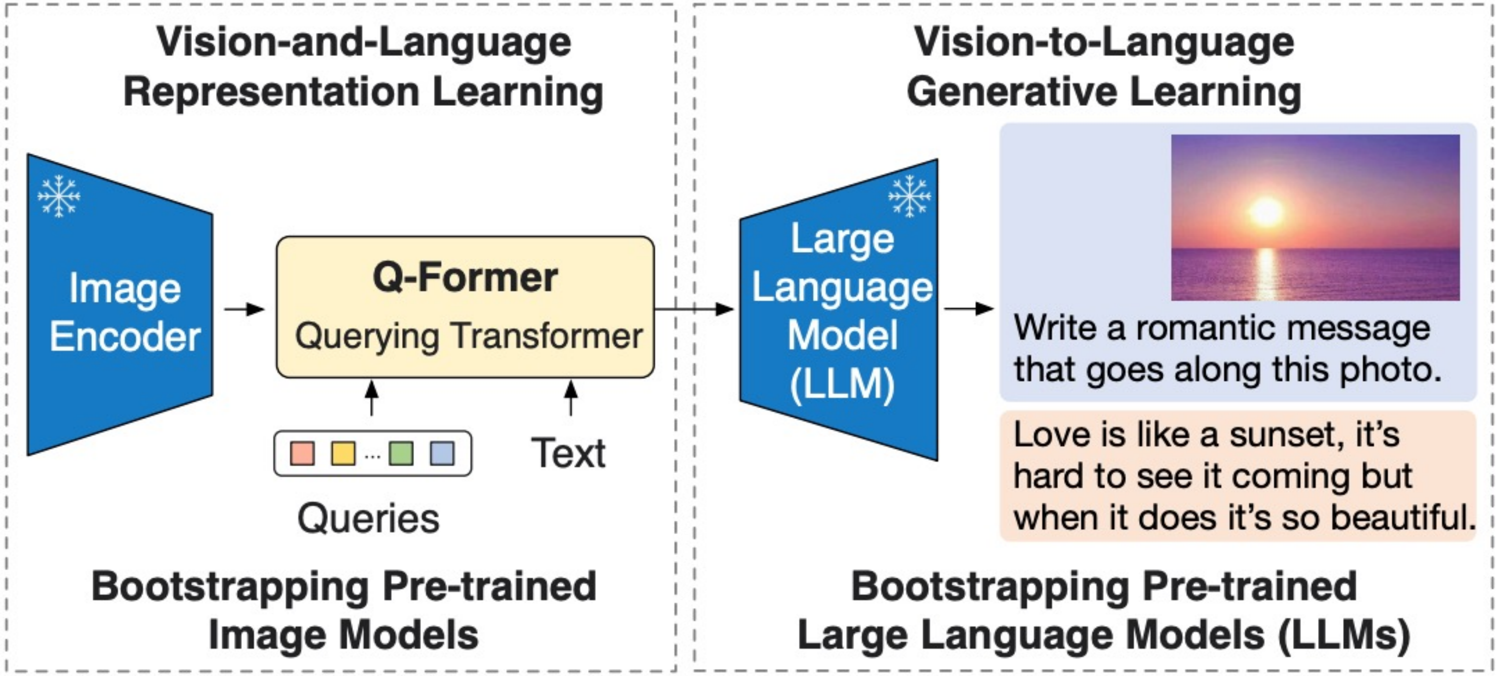


Figure 1. Overview of BLIP-2's framework. We pre-train a lightweight Querying Transformer following a two-stage strategy to bridge the modality gap. The first stage bootstraps vision-language representation learning from a frozen image encoder. The second stage bootstraps vision-to-language generative learning from a frozen LLM, which enables zero-shot instructed image-to-text generation (see Figure 4 for more examples).

Figure from
https://arxiv.org/pdf/2301.12597

Querying Transformer (BLIP-2): Stage 1

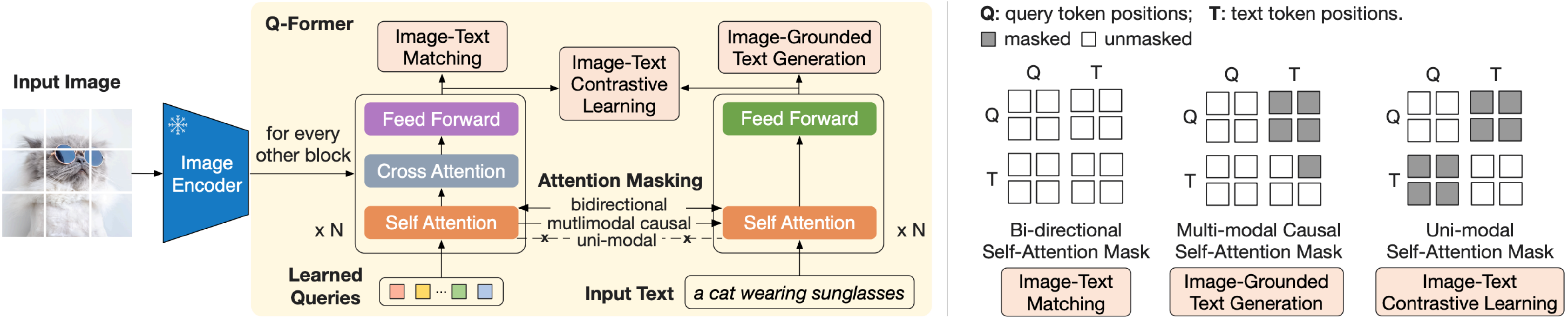


Figure 2. (**Left**) Model architecture of Q-Former and BLIP-2's first-stage vision-language representation learning objectives. We jointly optimize three objectives which enforce the queries (a set of learnable embeddings) to extract visual representation most relevant to the text. (**Right**) The self-attention masking strategy for each objective to control query-text interaction.

Figure from https://arxiv.org/pdf/2301.12597

Querying Transformer (BLIP-2): Stage 2

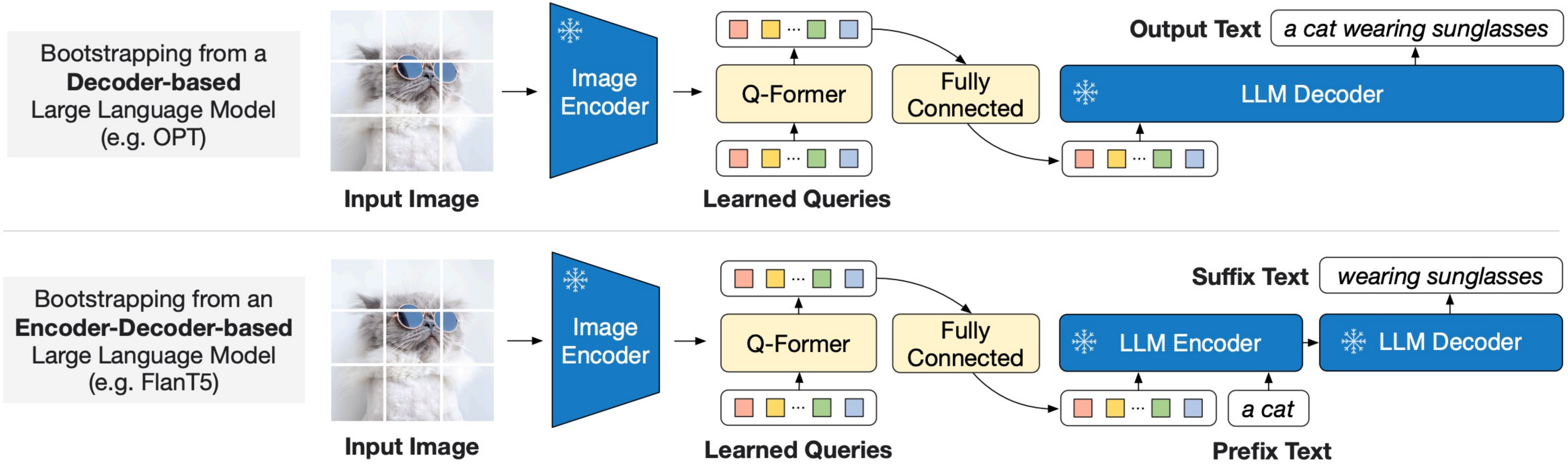


Figure 3. BLIP-2's second-stage vision-to-language generative pre-training, which bootstraps from frozen large language models (LLMs). (**Top**) Bootstrapping a decoder-based LLM (e.g. OPT). (**Bottom**) Bootstrapping an encoder-decoder-based LLM (e.g. FlanT5). The fully-connected layer adapts from the output dimension of the Q-Former to the input dimension of the chosen LLM.

Figure from https://arxiv.org/pdf/2301.12597

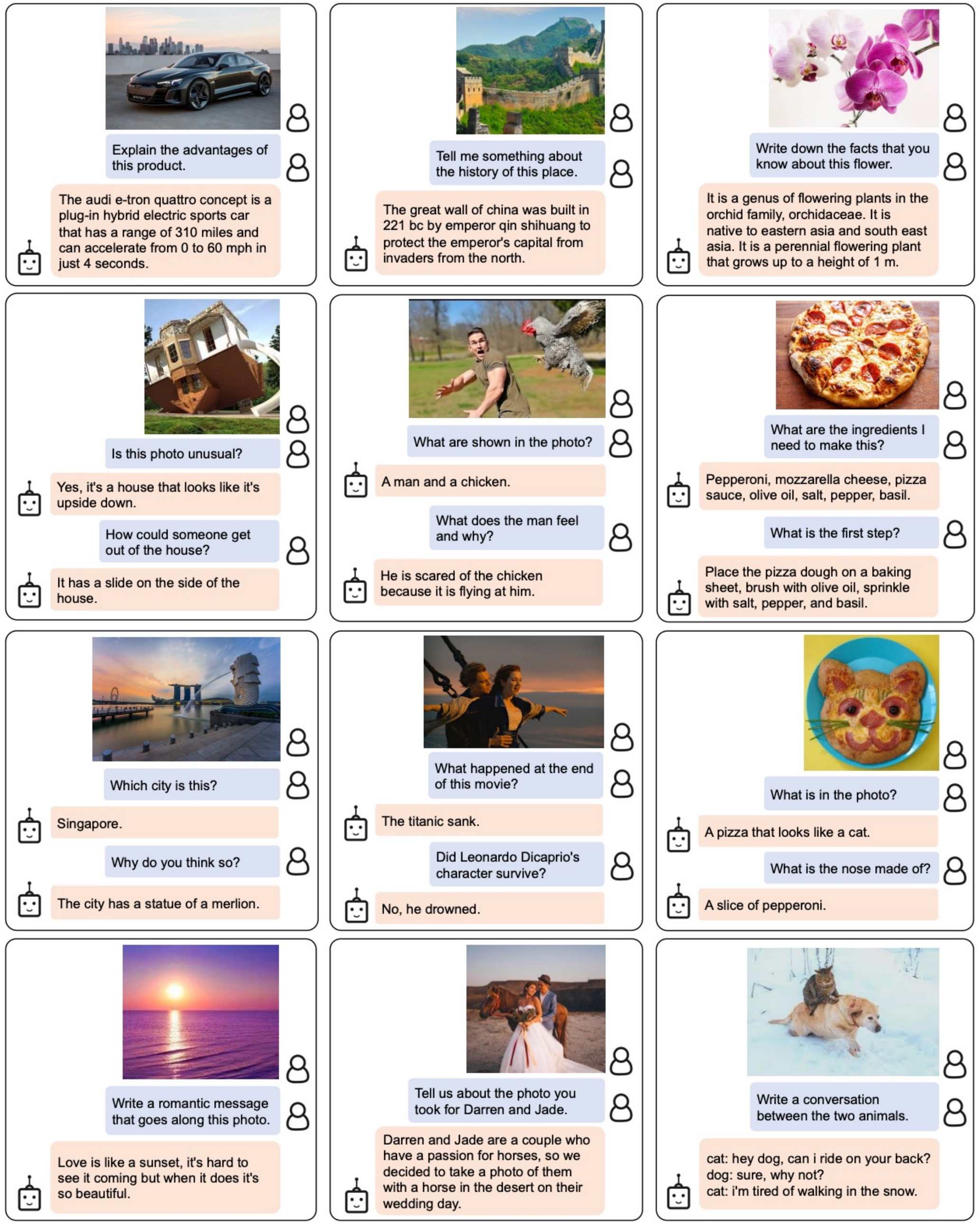


Figure 4. Selected examples of **instructed zero-shot image-to-text generation** using a BLIP-2 model w/ ViT-g and FlanT5$_{XXL}$, where it shows a wide range of capabilities including visual conversation, visual knowledge reasoning, visual commensense reasoning, storytelling, personalized image-to-text generation, etc.

Figure from https://arxiv.org/pdf/2301.12597

In this paper, we aim to deliver the promise of unified models via a simpler philosophy: *Render unto diffusion what is generative, and unto LLMs what is understanding.*

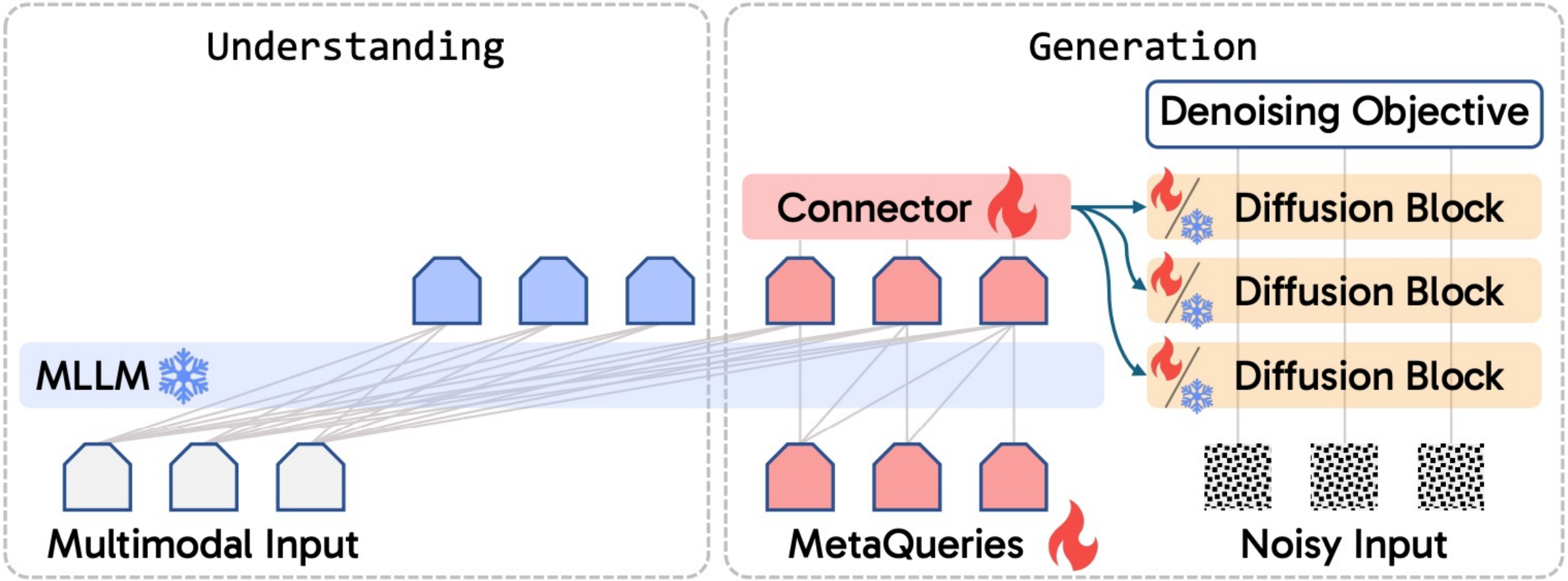


Figure 1 Overview of our model. Blue tokens maintain SOTA multimodal understanding; MetaQueries are learnable queries that directly applied to frozen MLLMs to query out conditions for generation. The model is tuned using only denoising objective with paired data. The generative diffusion models can be either frozen or further instruction-tuned for advanced generation tasks.

Figure from https://arxiv.org/pdf/2504.06256

Historical Note: PerceiverIO

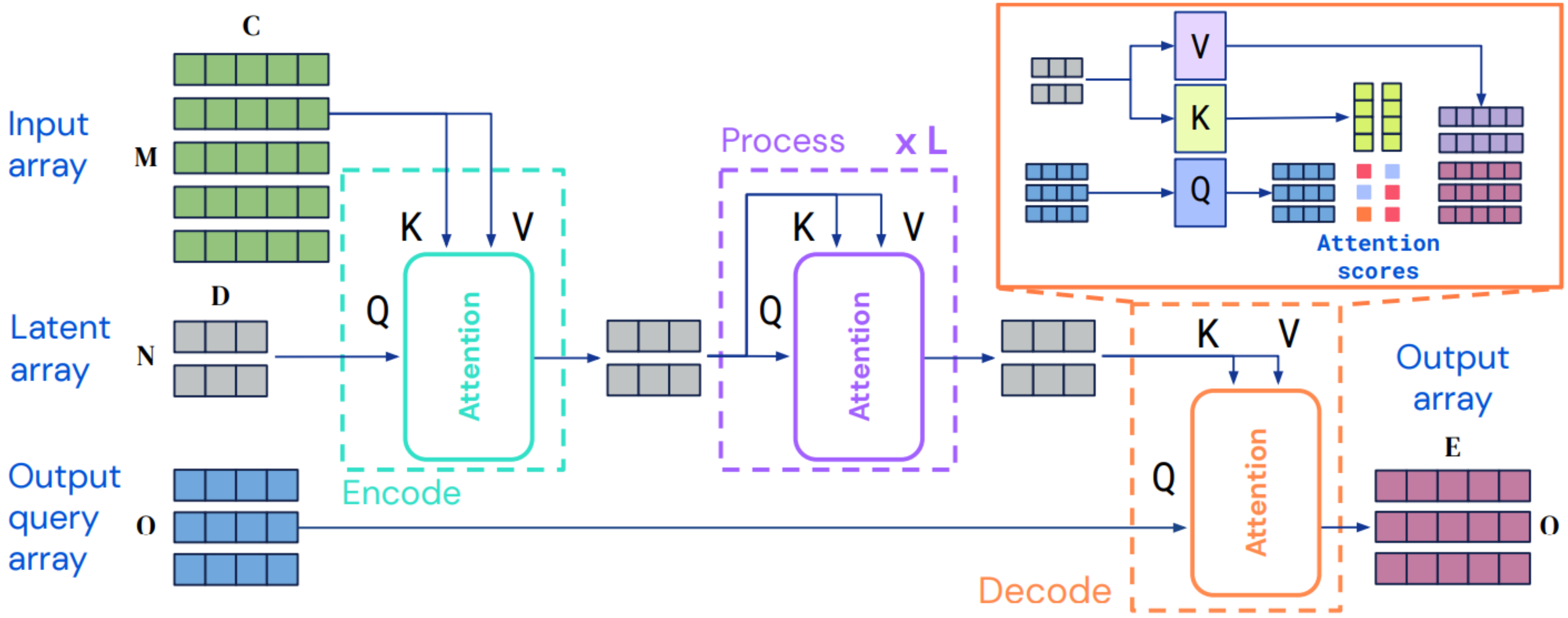
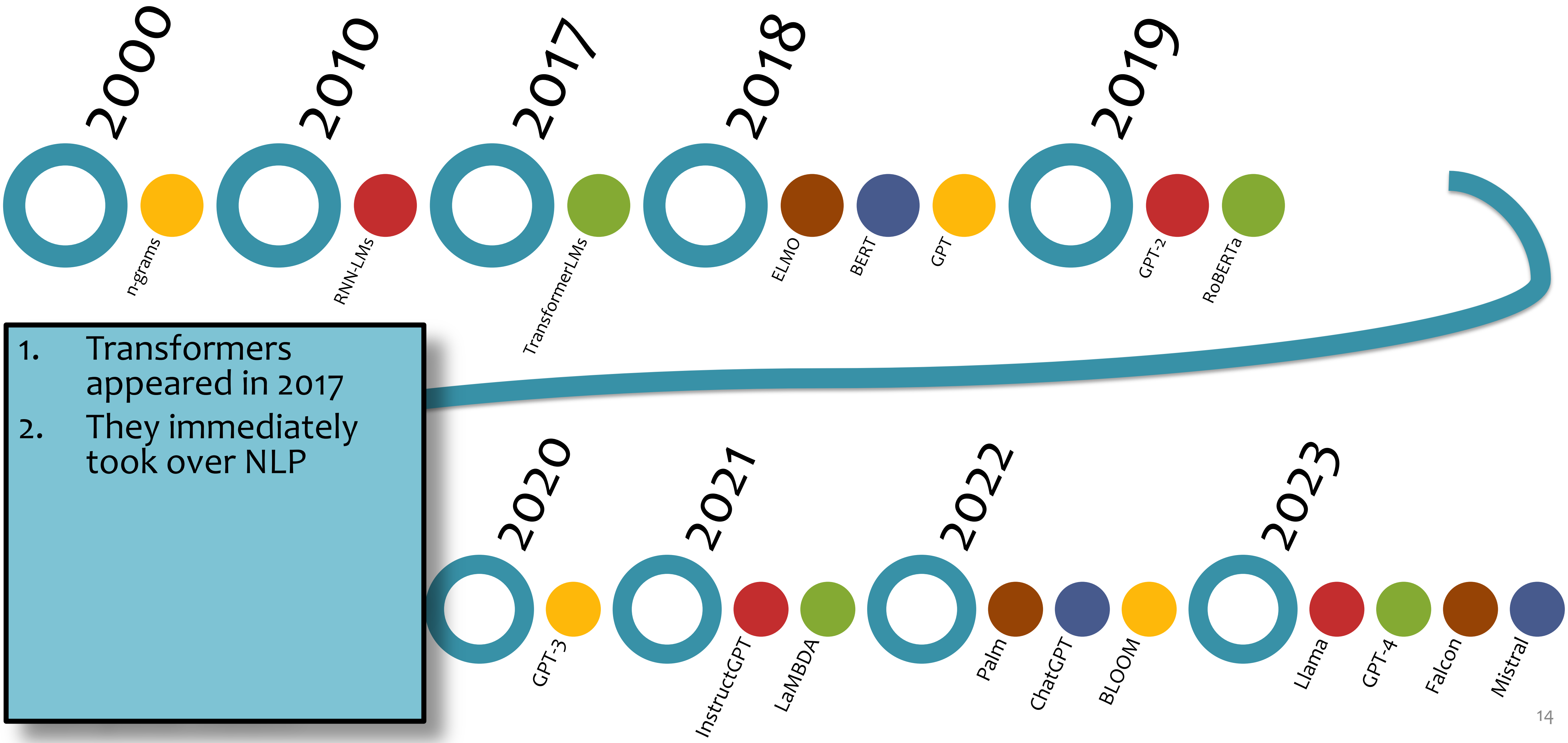


Figure 2: The Perceiver IO architecture. Perceiver IO maps arbitrary input arrays to arbitrary output arrays in a domain agnostic process. The bulk of the computation happens in a latent space whose size is typically smaller than the inputs and outputs, which makes the process computationally tractable even for very large inputs & outputs. See Fig. 5 for a more detailed look at encode, process, and decode attention.

Figure from https://arxiv.org/pdf/2107.14795

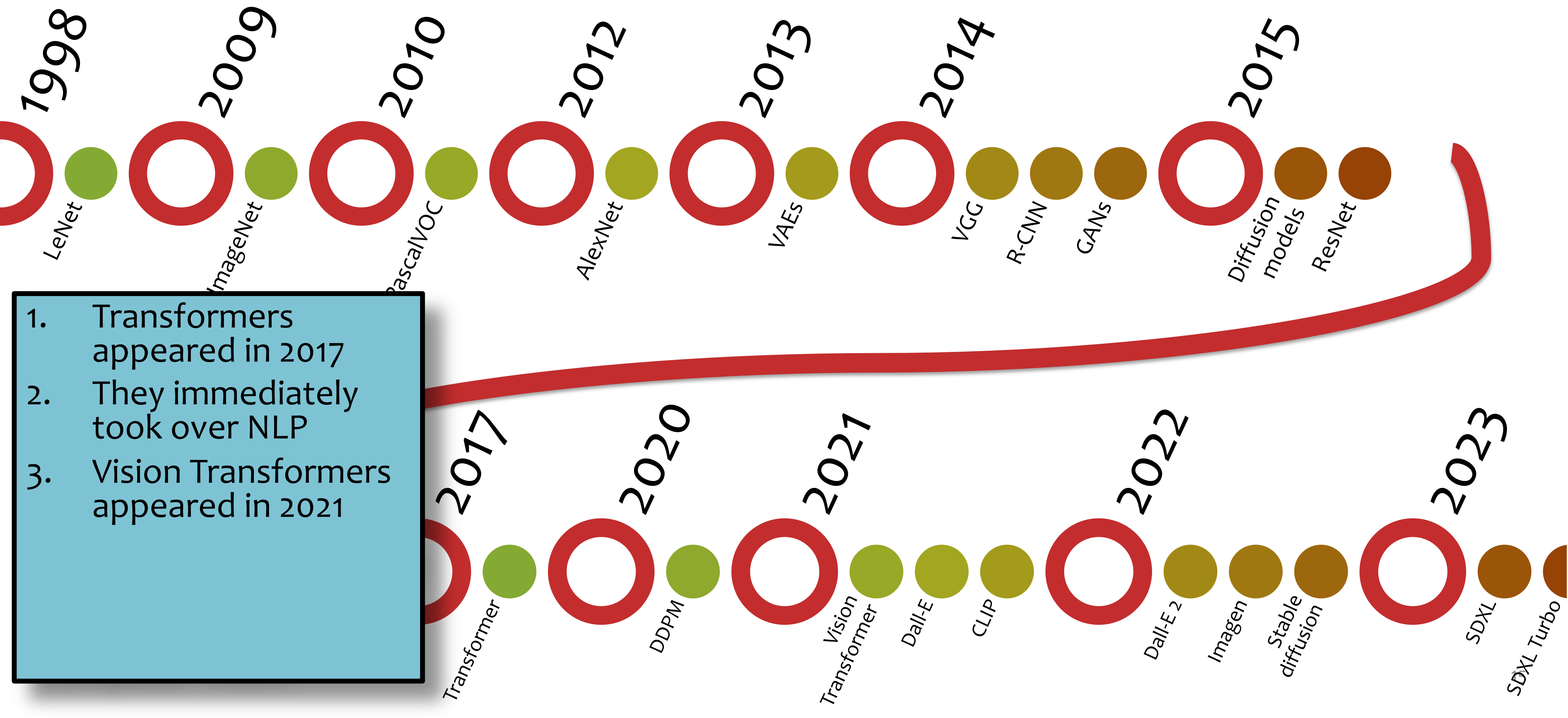
SCALING LAWS

Timeline: Language Modeling



1. Transformers appeared in 2017
2. They immediately took over NLP

Timeline: Image Generation



1. Transformers appeared in 2017
2. They immediately took over NLP
3. Vision Transformers appeared in 2021

How large are LLMs?

Comparison of some recent **large language models** (LLMs)

Model	Creators	Year of release	Training Data (# tokens)	Model Size (# parameters)
GPT-2	OpenAI	2019	~10 billion (40Gb)	1.5 billion
GPT-3 (cf. ChatGPT)	OpenAI	2020	300 billion	175 billion
PaLM	Google	2022	780 billion	540 billion
Chinchilla	DeepMind	2022	1.4 trillion	70 billion
LaMDA (cf. Bard)	Google	2022	1.56 trillion	137 billion
LLaMA	Meta	2023	1.4 trillion	65 billion
LLaMA-2	Meta	2023	2 trillion	70 billion
GPT-4 & GPT-5	OpenAI	2023 & 2025	?	? (1.76 trillion & 2-5 trillion)
Gemini (Ultra)	Google	2023	?	? (1.5 trillion)
LLaMA-3	Meta	2024	15 trillion	405 billion

Guiding Question this section: How did Meta choose this combination of training tokens / model parameters?

How much did it cost to train LLaMa?

Llama-1

When training a 65B-parameter model, our code processes around 380 tokens/sec/GPU on 2048 A100 GPU with 80GB of RAM. This means that training over our dataset containing 1.4T tokens takes approximately 21 days.

Llama-2

		Time (GPU hours)	Power Consumption (W)	Carbon Emitted (tCO_2eq)
LLAMA 2	7B	184320	400	31.22
	13B	368640	400	62.44
	34B	1038336	350	153.90
	70B	1720320	400	291.42
Total		3311616		539.00

Llama-3

Compute. Llama 3 405B is trained on up to 16K H100 GPUs, each running at 700W TDP with 80GB HBM3, using Meta's Grand Teton AI server platform (Matt Bowman, 2022). Each server is equipped with eight GPUs and two CPUs. Within a server, the eight GPUs are connected via NVLink. Training jobs are scheduled

	Time (GPU hours)	Power Consumption (W)	Carbon Emitted(tCO2eq)
Llama 3 8B	1.3M	700	390
Llama 3 70B	6.4M	700	1900

GPU Costs

- modern GPUs cost around $15k
- the cost of a cloud GPU per hour ranges $1-$4
- 700W = 0.7 kWh ➔ $0.084 per hour

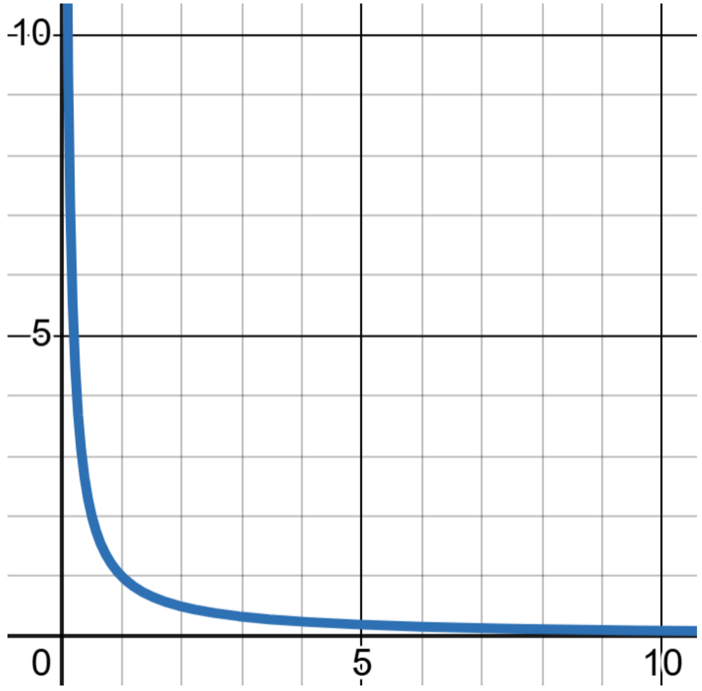
Question: How much did Llama-3 70B cost to train (say in a month)?

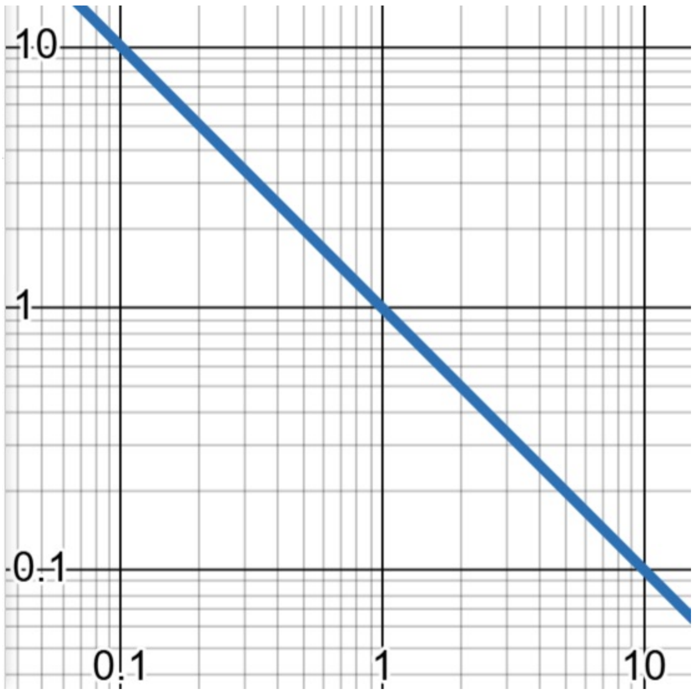
Answer:

Power Law

- Most scaling laws for LLMs assume we are fitting a power law function
- *Definition*: a **power law function** has the form

$$f(x) = cx^{-k}$$





Scaling Laws: Kaplan et al. (2020)

Experimental Design

- Varied a number LLM hyperparams
  - N: # parameters (768 – 1.5B)
  - D: # tokens (22M – 23B)
  - C: # FLOPS
  - model (depth, width, # heads, d_{model})
  - context length (1024 or less)
  - batch size (2^{19} or less)
- Evaluated test loss of each model

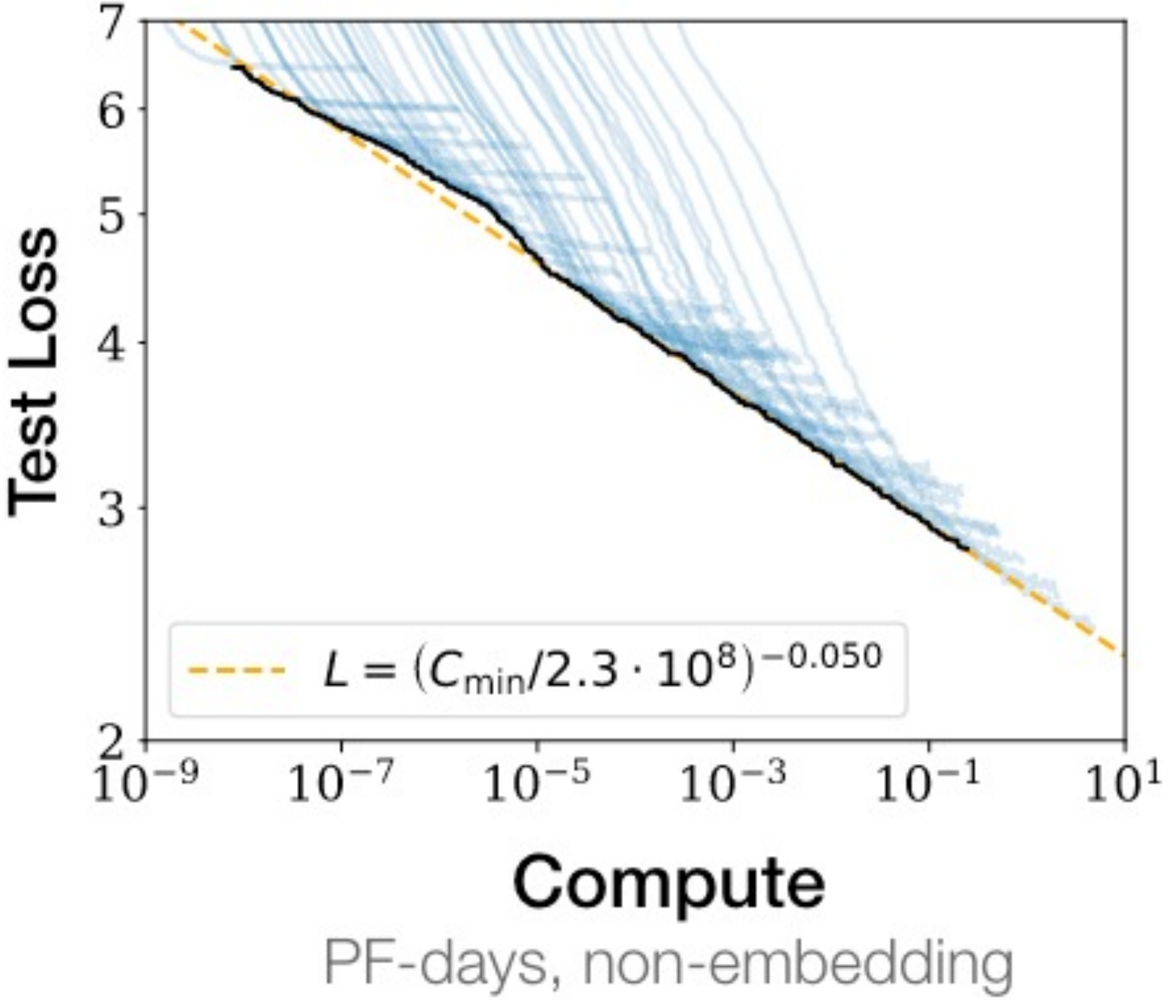


Figure from http://arxiv.org/abs/2001.08361

Scaling Laws: Kaplan et al. (2020)

Experimental Design

- Varied a number LLM hyperparams
  - N: # parameters (768 – 1.5B)
  - D: # tokens (22M – 23B)
  - C: # FLOPS
  - model (depth, width, # heads, d_{model})
  - context length (1024 or less)
  - batch size (2^{19} or less)
- Evaluated test loss of each model

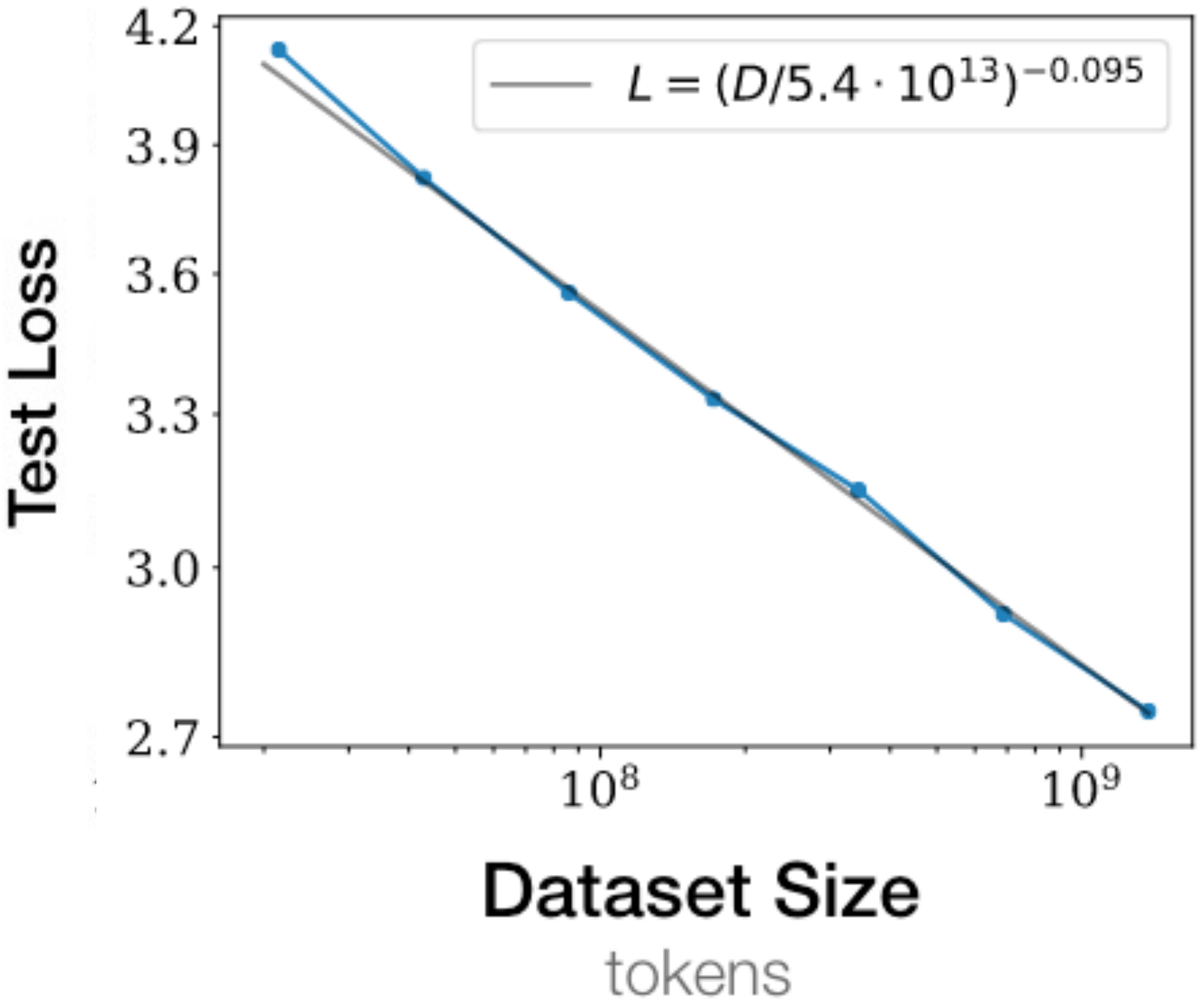


Figure from http://arxiv.org/abs/2001.08361

Scaling Laws: Kaplan et al. (2020)

Experimental Design

- Varied a number LLM hyperparams
  - N: # parameters (768 – 1.5B)
  - D: # tokens (22M – 23B)
  - C: # FLOPS
  - model (depth, width, # heads, d_{model})
  - context length (1024 or less)
  - batch size (2^{19} or less)
- Evaluated test loss of each model

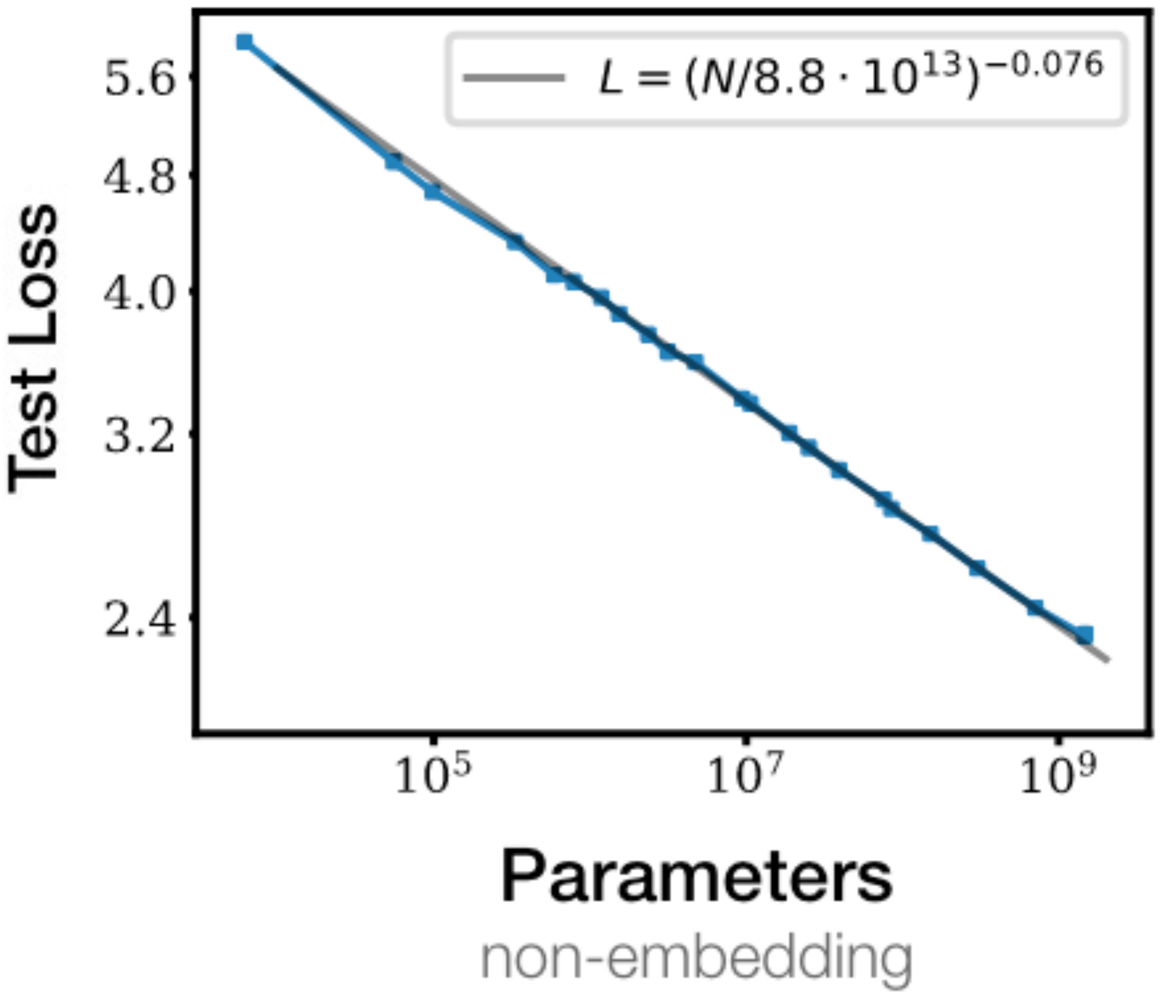
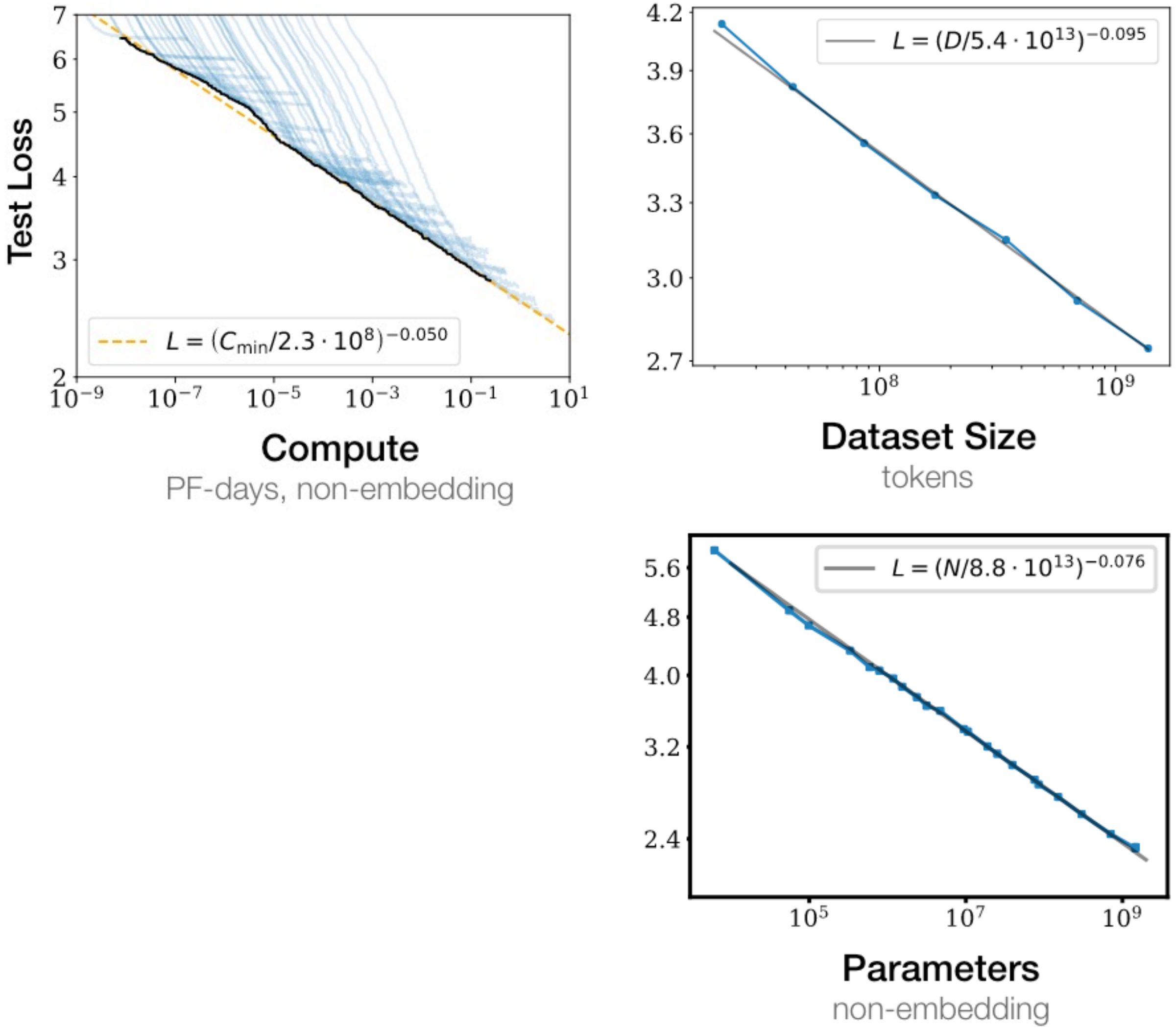


Figure from http://arxiv.org/abs/2001.08361

Scaling Laws: Kaplan et al. (2020)

Key takeaways:

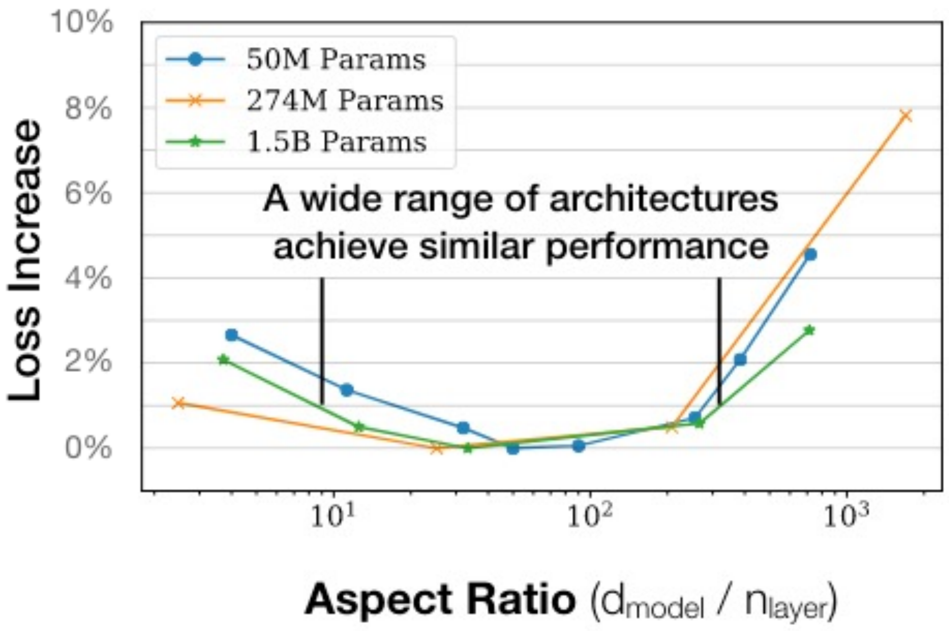
1. **three quantities dominate: N = # parameters, D = # tokens, C = # FLOPS**
2. model shape doesn't matter very much
3. performance improves as long as we increase both N and D
4. training loss curves follow predictable power laws
5. larger models are more sample efficient
6. convergence is not critical for good performance
7. best batch size follows a power law (and is huge: 1-2M tokens)



Scaling Laws: Kaplan et al. (2020)

Key takeaways:

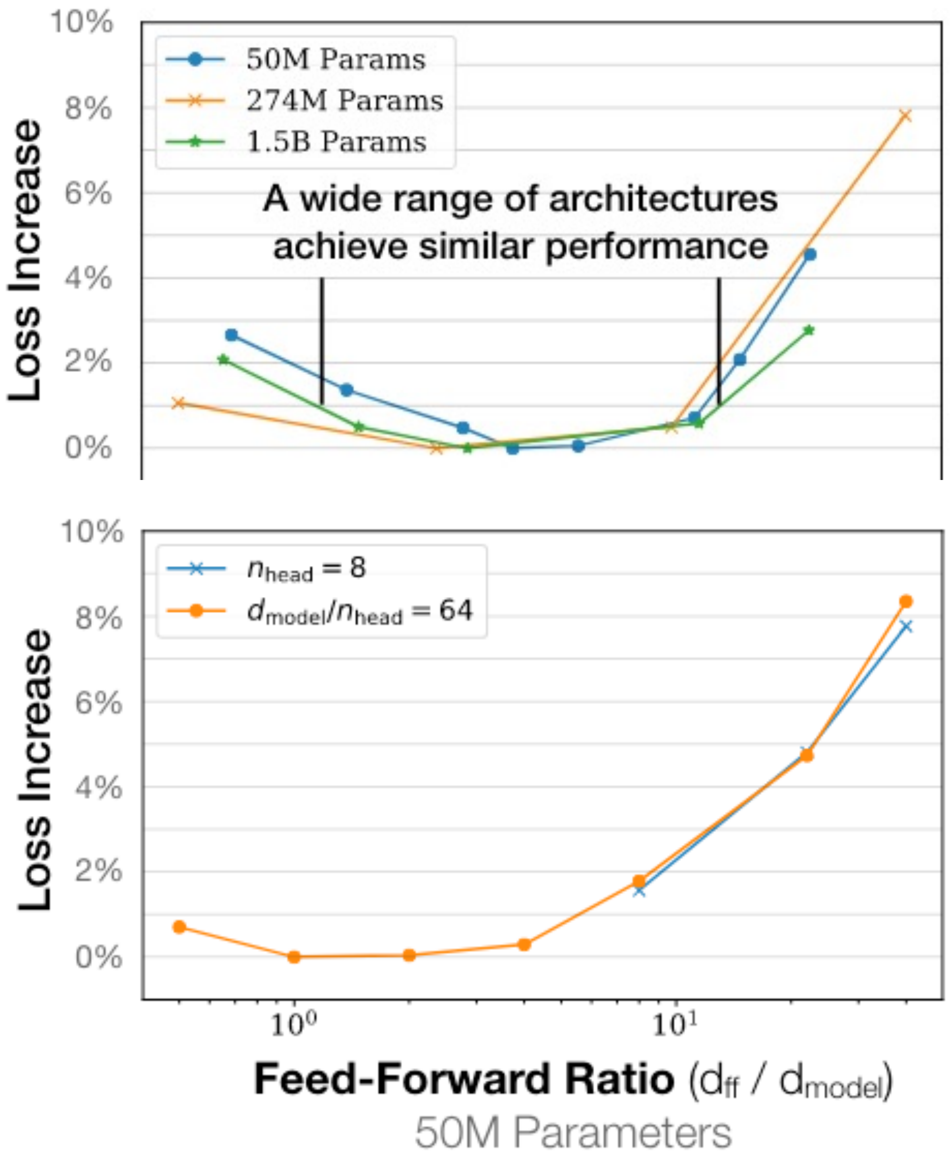
1. three quantities dominate:
   N = # parameters, D = # tokens, C = # FLOPS
2. **model shape doesn't matter very much**
3. performance improves as long as we increase both N and D
4. training loss curves follow predictable power laws
5. larger models are more sample efficient
6. convergence is not critical for good performance
7. best batch size follows a power law (and is huge: 1-2M tokens)



Scaling Laws: Kaplan et al. (2020)

Key takeaways:

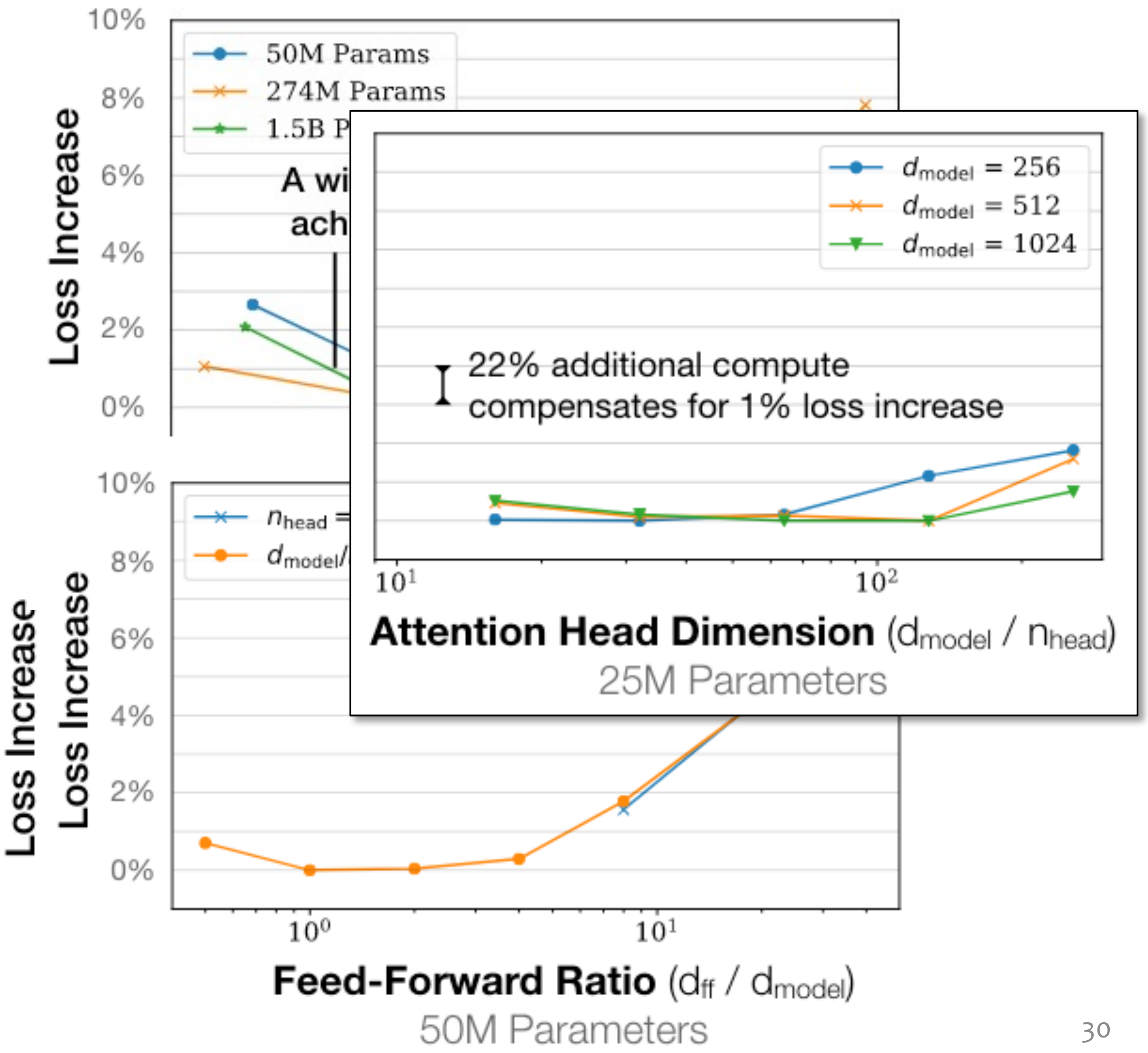
1. three quantities dominate: N = # parameters, D = # tokens, C = # FLOPS
2. **model shape doesn't matter very much**
3. performance improves as long as we increase both N and D
4. training loss curves follow predictable power laws
5. larger models are more sample efficient
6. convergence is not critical for good performance
7. best batch size follows a power law (and is huge: 1-2M tokens)



Scaling Laws: Kaplan et al. (2020)

Key takeaways:

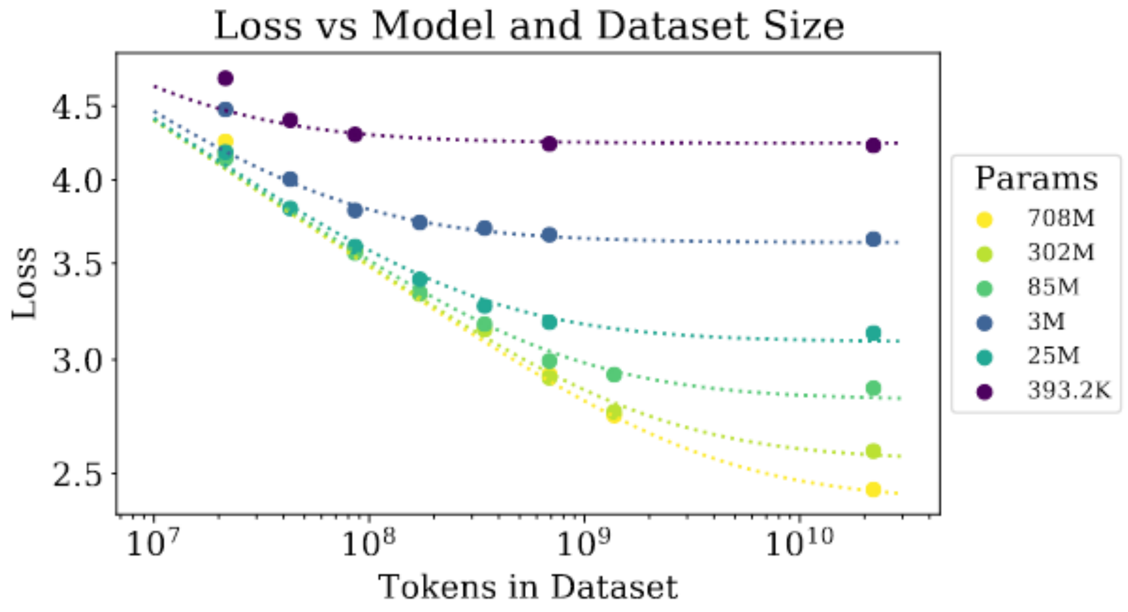
1. three quantities dominate:
   N = # parameters, D = # tokens, C = # FLOPS
2. **model shape doesn't matter very much**
3. performance improves as long as we increase both N and D
4. training loss curves follow predictable power laws
5. larger models are more sample efficient
6. convergence is not critical for good performance
7. best batch size follows a power law (and is huge: 1-2M tokens)



Scaling Laws: Kaplan et al. (2020)

Key takeaways:

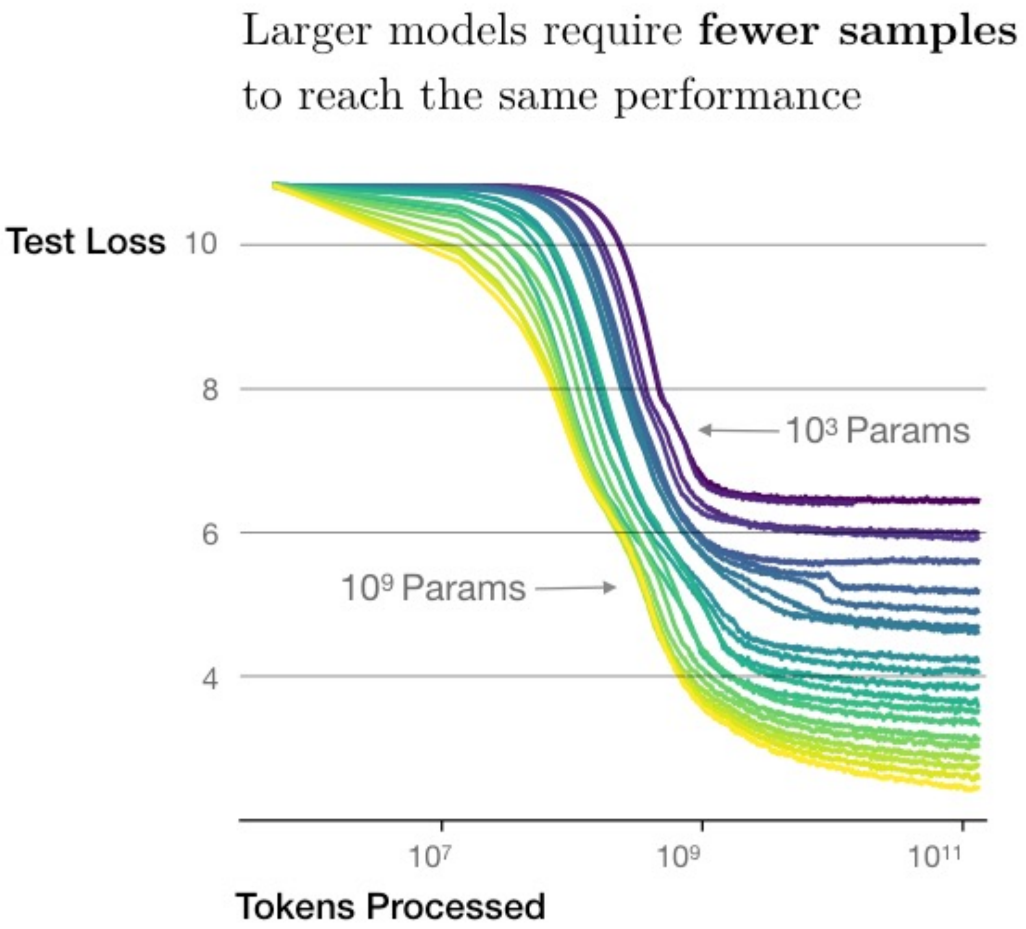
1. three quantities dominate:
   N = # parameters, D = # tokens, C = # FLOPS
2. model shape doesn't matter very much
3. performance improves as long as we increase both N and D
4. **training loss / test loss curves follow predictable power laws**
5. larger models are more sample efficient
6. convergence is not critical for good performance
7. best batch size follows a power law (and is huge: 1-2M tokens)



Scaling Laws: Kaplan et al. (2020)

Key takeaways:

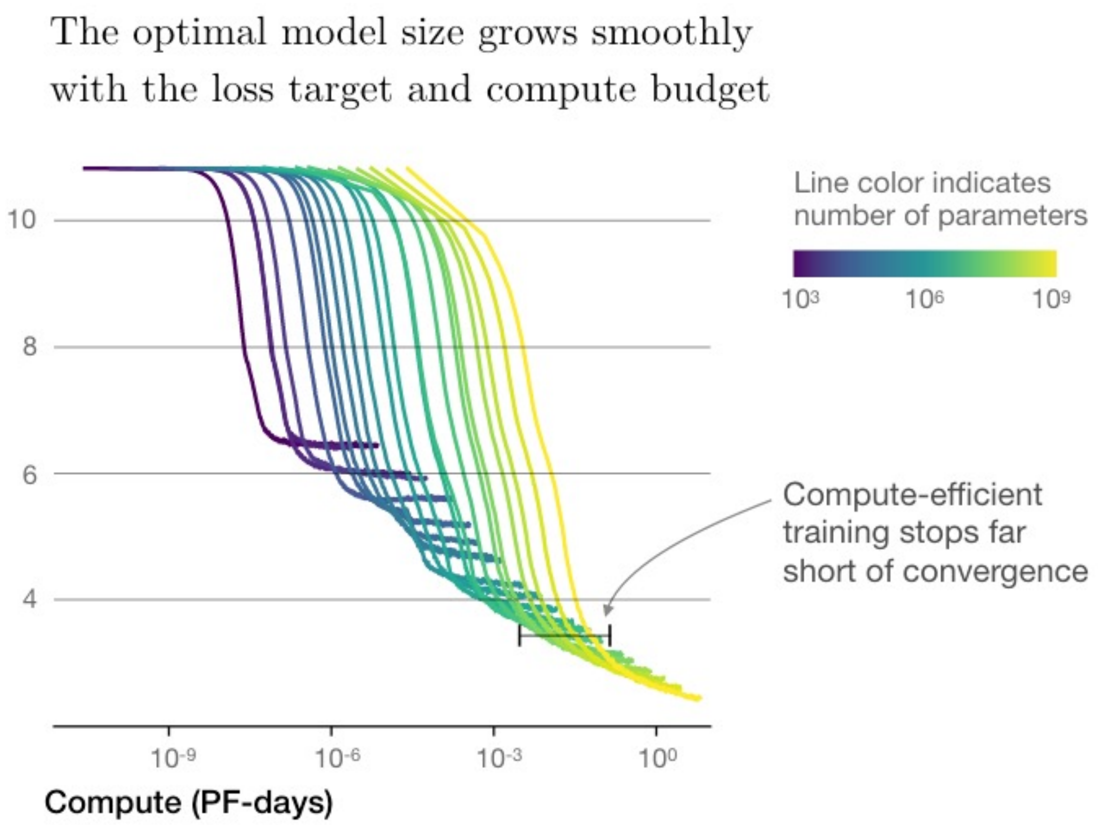
1. three quantities dominate:
   N = # parameters, D = # tokens, C = # FLOPS
2. model shape doesn't matter very much
3. performance improves as long as we increase both N and D
4. training loss / test loss curves follow predictable power laws
5. **larger models are more sample efficient**
6. convergence is not critical for good performance
7. best batch size follows a power law (and is huge: 1-2M tokens)



Scaling Laws: Kaplan et al. (2020)

Key takeaways:

1. three quantities dominate:
   N = # parameters, D = # tokens, C = # FLOPS
2. model shape doesn't matter very much
3. performance improves as long as we increase both N and D
4. training loss / test loss curves follow predictable power laws
5. larger models are more sample efficient
6. **convergence is not critical for good performance**
7. best batch size follows a power law (and is huge: 1-2M tokens)



Scaling Laws: Kaplan et al. (2020)

Key takeaways:

1. three quantities domi
N = # parameters, D =
FLOPS
2. model shape doesn't
much
3. performance improve
we increase both N ar
4. training loss / test loss
predictable power law
5. larger models are mor
efficient
6. convergence is not cri
performance
7. **best batch size follows a power law (and is huge: 1-2M tokens)**

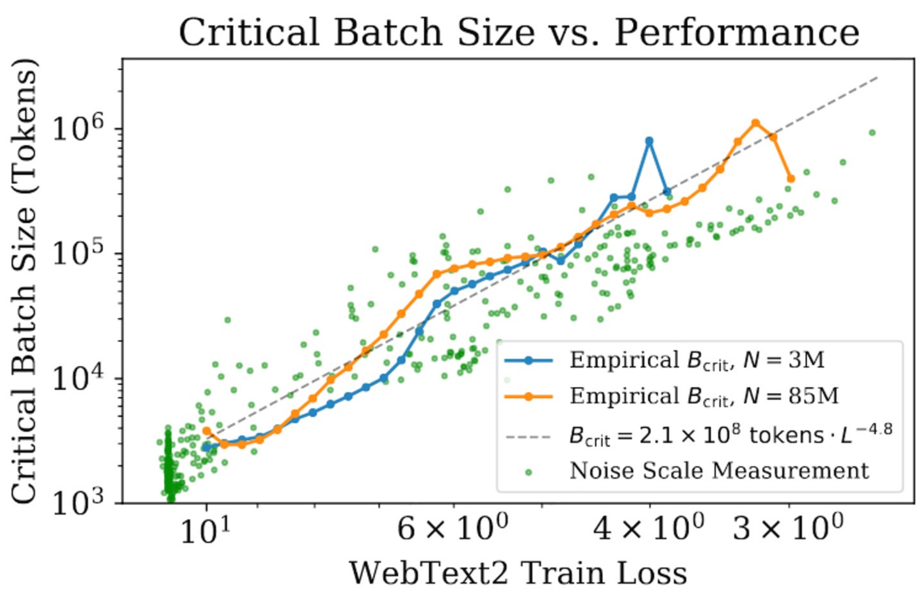


Figure 10 The critical batch size B_{crit} follows a power law in the loss as performance increase, and does not depend directly on the model size. We find that the critical batch size approximately doubles for every 13% decrease in loss. B_{crit} is measured empirically from the data shown in Figure 18, but it is also roughly predicted by the gradient noise scale, as in [MKAT18].

Scaling Laws: Kaplan et al. (2020)

Key takeaways:

1. three quantities dominate:
   N = # parameters, D = # tokens, C = # FLOPS
2. model shape doesn't matter very much
3. **performance improves as long as we increase both N and D**
4. training loss curves follow predictable power laws
5. larger models are more sample efficient
6. convergence is not critical for good performance
7. best batch size follows a power law (and is huge: 1-2M tokens)

"every time we increase the model size 8x, we only need to increase the data by roughly 5x to avoid a penalty."

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

Parameter	α_N	α_D	N_c	D_c
Value	0.076	0.103	6.4×10^{13}	1.8×10^{13}

Table 2 Fits to $L(N, D)$

But Hoffman et al. (2022) tell a very different story!

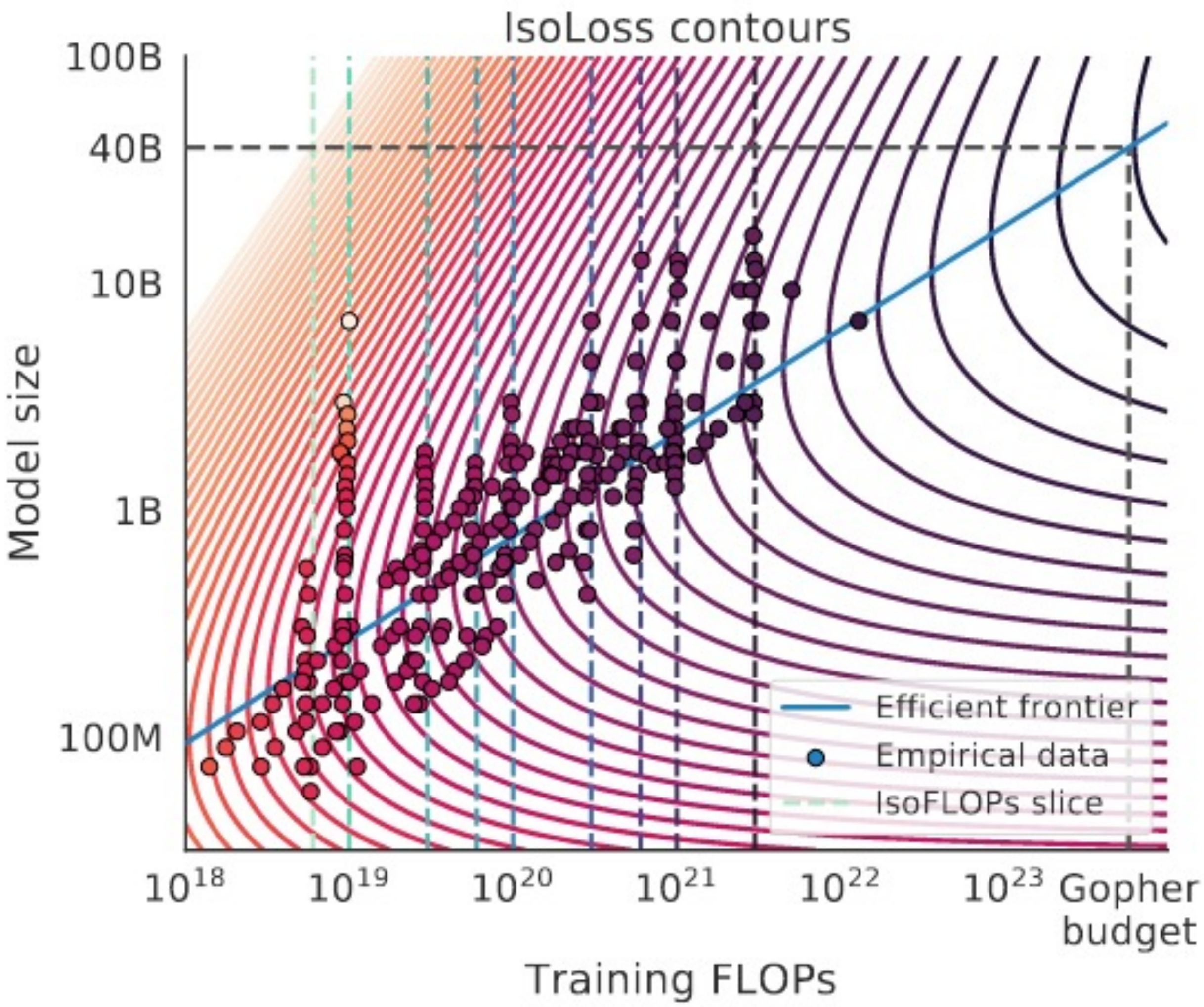
Improved Scaling Laws: Hoffman et al. (2022)

- The big shift for Chinchilla was a dramatic increase in the number of tokens
- Kaplan et al. (2020) said 8x increase in # parameters should have a 5x increase in # tokens
- But Chinchilla found you should increase both proportionally (2x # parameters + 2x # tokens)

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

Improved Scaling Laws: Hoffman et al. (2022)

- Data:
  - Fixed: C = # FLOPS
  - Experiments varied:
    - D = # tokens
    - N = # parameters
  - Measured: L(N, D)
- Learned a model to predict L(N,D) for any N and D
- Used this model to predict optimal model size



Improved Scaling Laws: Hoffman et al. (2022)

The key finding is that everyone had been using way **too little data**

Increasing the amount of data and **decreasing** the model size,
→ Same computational budget but get much better performance!

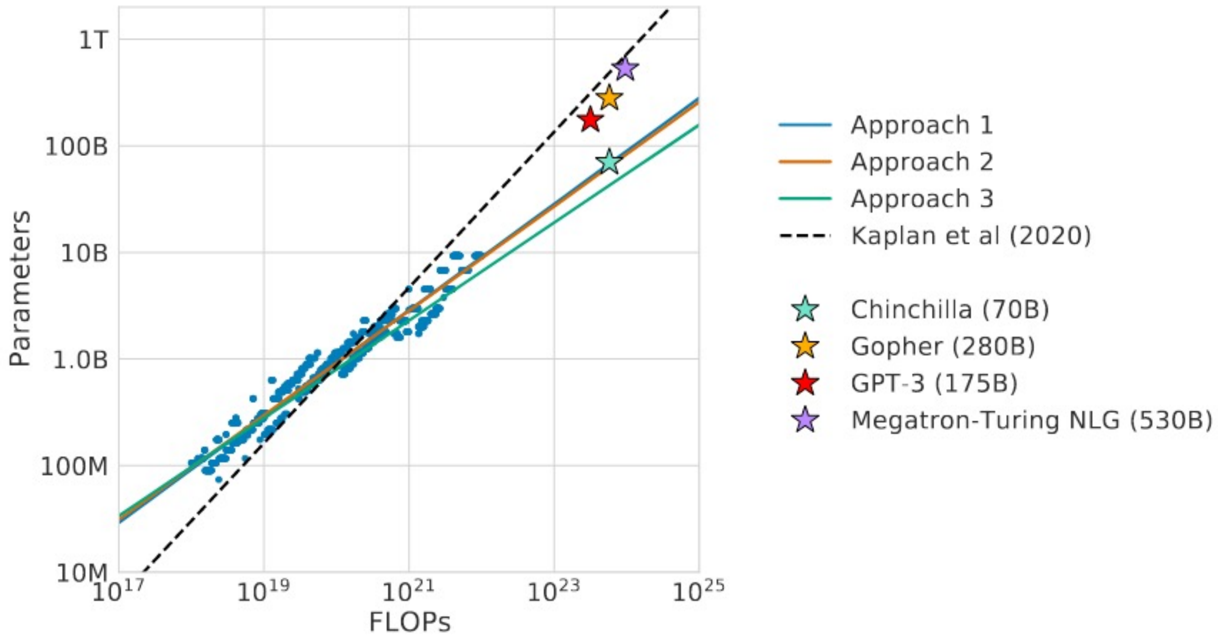


Figure 1 | **Overlaid predictions.** We overlay the predictions from our three different approaches, along with projections from Kaplan et al. (2020). We find that all three methods predict that current large models should be substantially smaller and therefore trained much longer than is currently done.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

Improved Scaling Laws: Hoffman et al. (2022)

The key finding is that everyone had been using way **too little data**

Increasing the amount of data and **decreasing** the model size,
→ Same computational budget but get much better performance!

	Chinchilla	*Gopher*	GPT-3	MT-NLG 530B	Supervised SOTA
HellaSWAG	**80.8%**	79.2%	78.9%	80.2%	93.9%
PIQA	81.8%	81.8%	81.0%	**82.0%**	90.1%
Winogrande	**74.9%**	70.1%	70.2%	73.0%	91.3%
SIQA	**51.3%**	50.6%	-	-	83.2%
BoolQ	**83.7%**	79.3%	60.5%	78.2%	91.4%

Table 8 | **Zero-shot comparison on Common Sense benchmarks.** We show a comparison between *Chinchilla*, *Gopher*, and MT-NLG 530B on various Common Sense benchmarks. We see that *Chinchilla* matches or outperforms *Gopher* and GPT-3 on all tasks. On all but one *Chinchilla* outperforms the much larger MT-NLG 530B model.

	Chinchilla	*Gopher*	GPT-3	MT-NLG 530B
LAMBADA Zero-Shot	**77.4**	74.5	76.2	76.6
RACE-m Few-Shot	**86.8**	75.1	58.1	-
RACE-h Few-Shot	**82.3**	71.6	46.8	47.9

Table 7 | **Reading comprehension.** On RACE-h and RACE-m (Lai et al., 2017), *Chinchilla* considerably improves performance over *Gopher*. Note that GPT-3 and MT-NLG 530B use a different prompt format than we do on RACE-h/m, so results are not comparable to *Gopher* and *Chinchilla*. On LAMBADA (Paperno et al., 2016), *Chinchilla* outperforms both *Gopher* and MT-NLG 530B.

ADJUSTING QUALITY OF DATA

Phi Family of (small) LLMs

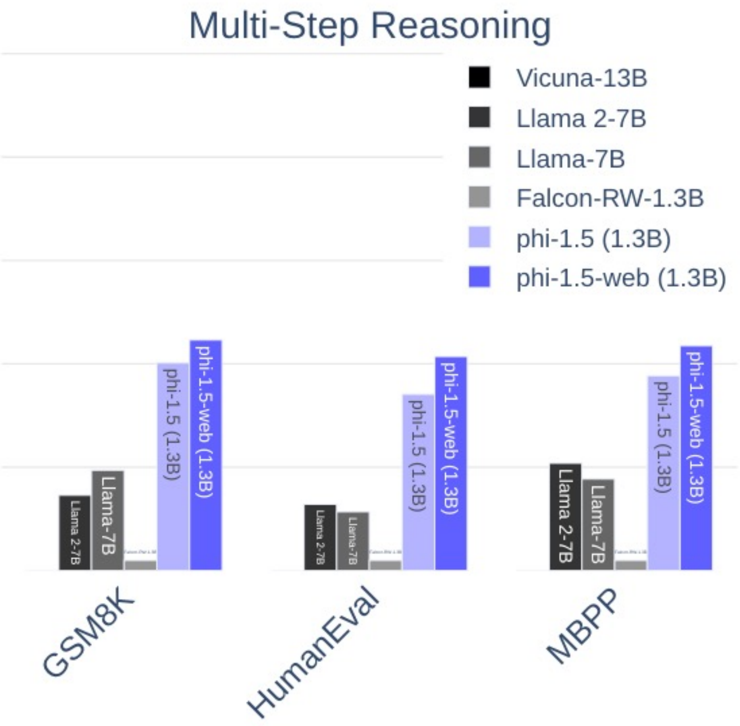
- Key idea: Instead of increasing the size of the model / data, increase the quality of your data
- Paper Title: "Textbooks Are All You Need"
- Results from Phi-1 show performance commensurate with much larger coding models

Date	Model	Model size (Parameters)	Dataset size (Tokens)	HumanEval (Pass@1)	MBPP (Pass@1)
2021 Jul	Codex-300M [CTJ+21]	300M	100B	13.2%	-
2021 Jul	Codex-12B [CTJ+21]	12B	100B	28.8%	-
2022 Mar	CodeGen-Mono-350M [NPH+23]	350M	577B	12.8%	-
2022 Mar	CodeGen-Mono-16.1B [NPH+23]	16.1B	577B	29.3%	35.3%
2022 Apr	PaLM-Coder [CND+22]	540B	780B	35.9%	47.0%
2022 Sep	CodeGeeX [ZXZ+23]	13B	850B	22.9%	24.4%
2022 Nov	GPT-3.5 [Ope23]	175B	N.A.	47%	-
2022 Dec	SantaCoder [ALK+23]	1.1B	236B	14.0%	35.0%
2023 Mar	GPT-4 [Ope23]	N.A.	N.A.	67%	-
2023 Apr	Replit [Rep23]	2.7B	525B	21.9%	-
2023 Apr	Replit-Finetuned [Rep23]	2.7B	525B	30.5%	-
2023 May	CodeGen2-1B [NHX+23]	1B	N.A.	10.3%	-
2023 May	CodeGen2-7B [NHX+23]	7B	N.A.	19.1%	-
2023 May	StarCoder [LAZ+23]	15.5B	1T	33.6%	52.7%
2023 May	StarCoder-Prompted [LAZ+23]	15.5B	1T	40.8%	49.5%
2023 May	PaLM 2-S [ADF+23]	N.A.	N.A.	37.6%	50.0%
2023 May	CodeT5+ [WLG+23]	2B	52B	24.2%	-
2023 May	CodeT5+ [WLG+23]	16B	52B	30.9%	-
2023 May	InstructCodeT5+ [WLG+23]	16B	52B	35.0%	-
2023 Jun	WizardCoder [LXZ+23]	16B	1T	57.3%	51.8%
2023 Jun	**phi-1**	1.3B	7B	50.6%	55.5%

Figure from http://arxiv.org/abs/2306.11644

Phi Family of (small) LLMs

- Key idea: Instead of increasing the size of the model / data, increase the quality of your data
- Paper Title: "Textbooks Are All You Need"
- Results from Phi-1.5 show performance commensurate with much larger LLMs



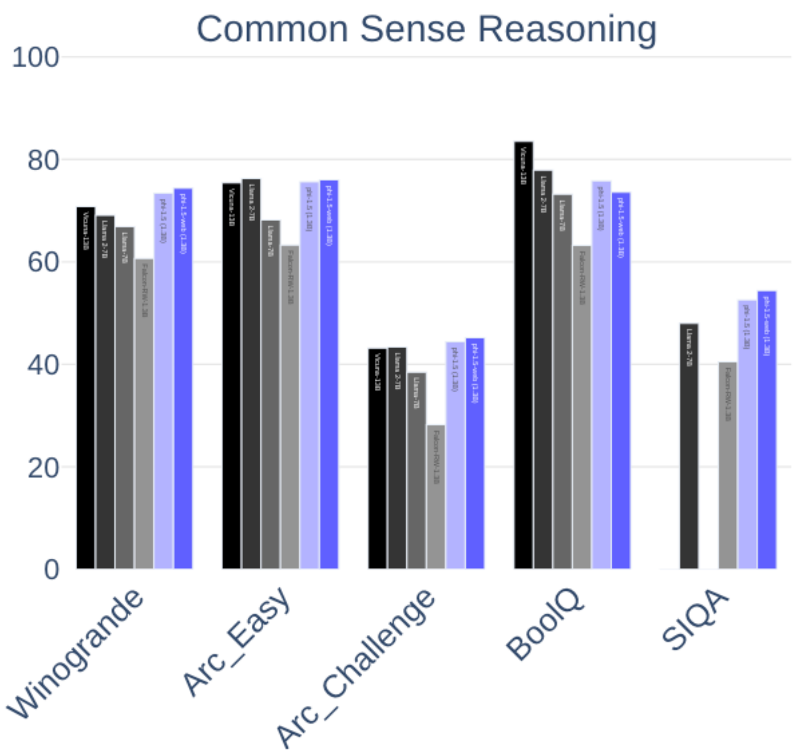


Figure from http://arxiv.org/abs/2309.05463

Scaling Laws for Data Filtering

- Recent trend towards emphasizing data quality and not just data quantity
- But difficult to predict how to tradeoff between data **quantity** and **quality** and **compute**
- Scaling laws for data quality and data size (Goyal et al. 2024) suggest that as your amount of compute goes up, you can get away with less filtering

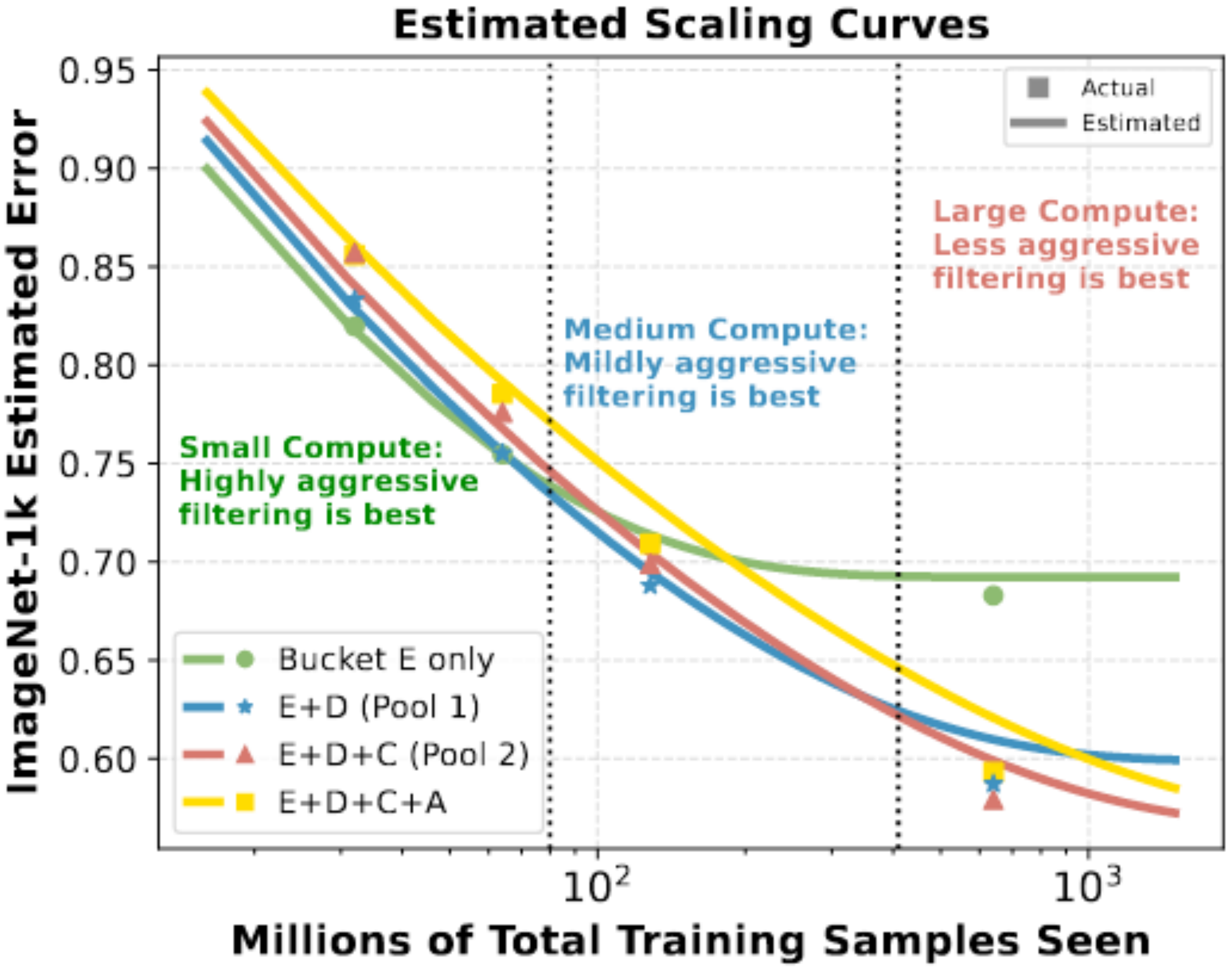
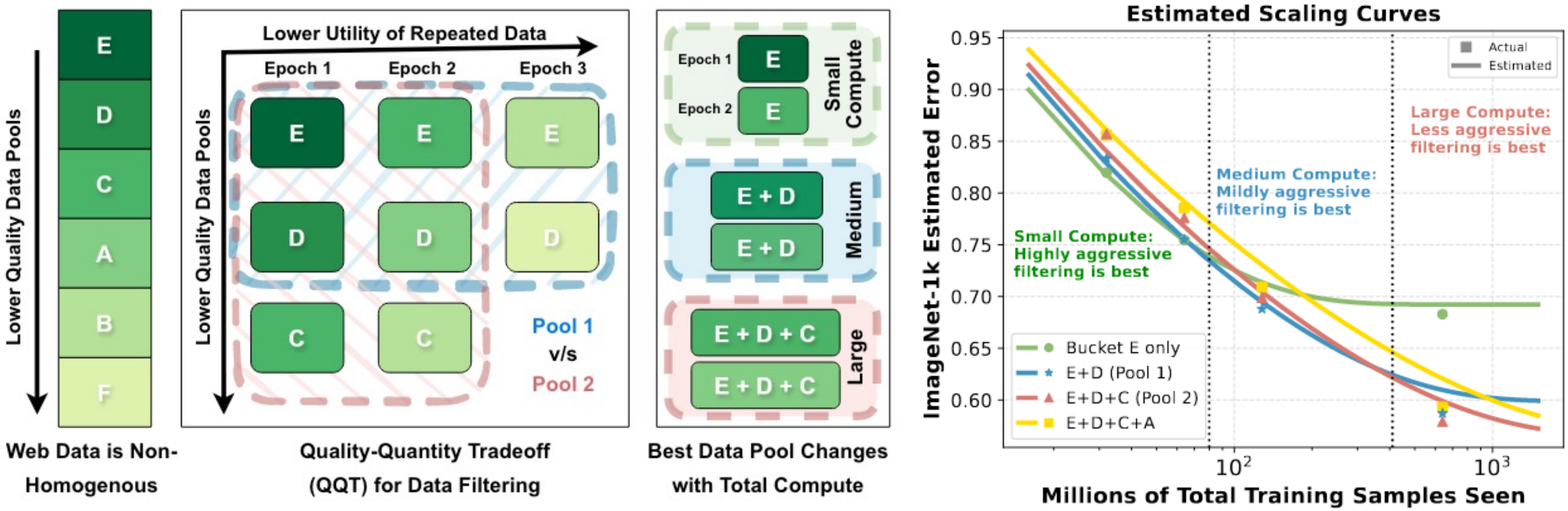


Figure from http://arxiv.org/abs/2404.07177

Scaling Laws for Data Filtering



Web Data is Non-Homogenous

Quality-Quantity Tradeoff (QQT) for Data Filtering

Best Data Pool Changes with Total Compute

Figure from http://arxiv.org/abs/2404.07177