CS 639: Foundation Models
Scaling II

Fred Sala

University of Wisconsin-Madison

April 16, 2026

Announcements

- **Homework 4**: due soon
- Fred's OH: time changed to **1:30-3:00 pm today**
- **Project---ongoing!**
  - Grading information here:
  - https://pages.cs.wisc.edu/~fredsala/cs639/files/project_rubric_639.pdf
- Class outline

Thursday April 16	Scaling II
Tuesday April 21	Agents I
Thursday April 23	Agents II
Tuesday April 28	Applications

Building Models and Systems

Outline

- **Scaling Laws Review & Breaking Beyond Laws**
  - Laws, Chinchilla-style compute optimality, data pruning, mixture-of-experts, etc.
- **Test-Time Scaling**
  - Motivation, taxonomy and forms of test-time scaling, aggregation and verification
- **Combined Scaling**
  - One example of combined training and test-time scaling recipes

Outline

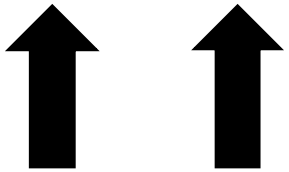
- **Scaling Laws Review & Breaking Beyond Laws**
  - Laws, Chinchilla-style compute optimality, data pruning, mixture-of-experts, etc.
- **Test-Time Scaling**
  - Motivation, taxonomy and forms of test-time scaling, aggregation and verification
- **Combined Scaling**
  - One example of combined training and test-time scaling recipes

Scaling: **Power Laws**

How to model relationships measured?

- Power laws

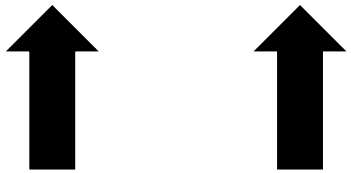
$$f(x) = ax^{-k}$$



Coefficient Exponent

- In our case, for model size and training to convergence,

$$L(N) = (N_c/N)^{\alpha_N}; \quad \alpha_N \sim 0.076, \quad N_c \sim 8.8 \times 10^{13}$$

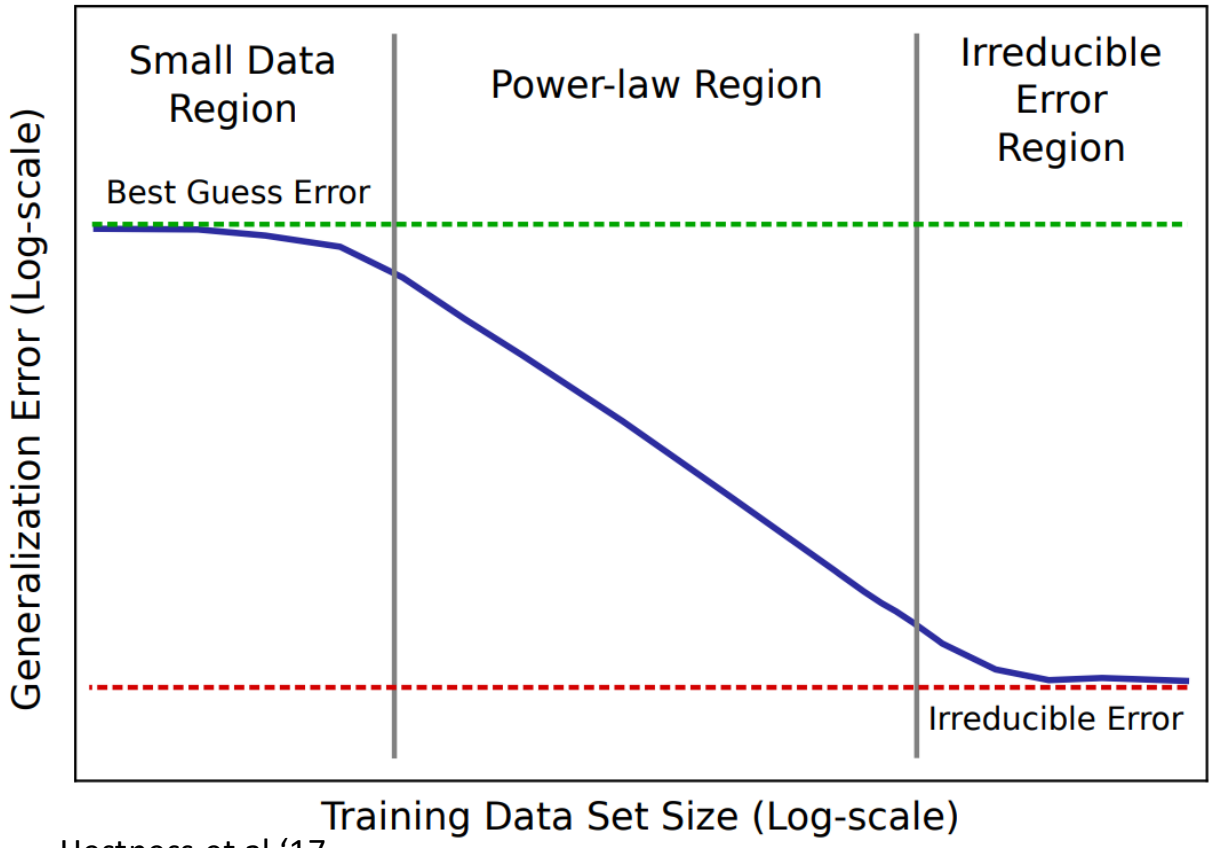


Coefficient Exponent

Scaling: Power Laws

Not a new idea. For data: hypothetical power-law like scaling

- **Note:** different regimes

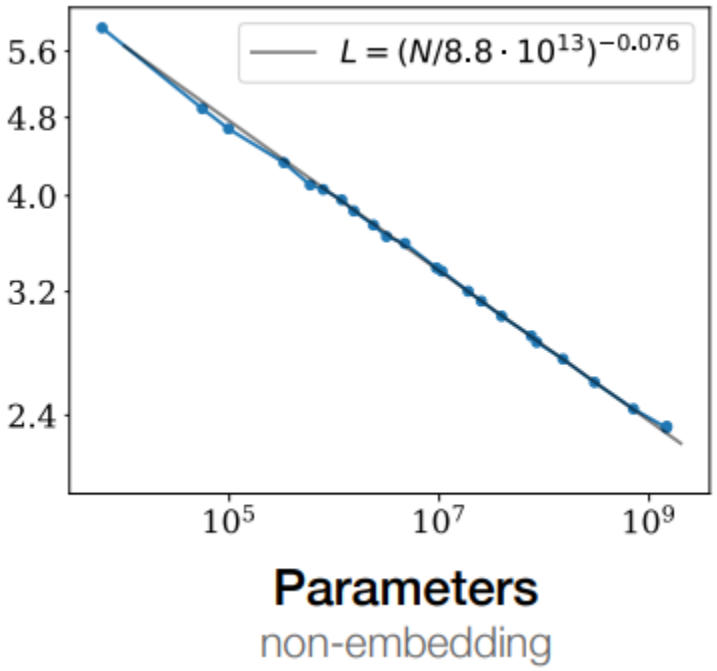


Hestness et al ‘17

Scaling: **Varying the Model Size**

Let's see this in detail.

Kaplan et al '20. Fix the dataset (large).



- **Vary model size**: 769 to 1.5B
- Measure test loss

- Fit the curve as before:

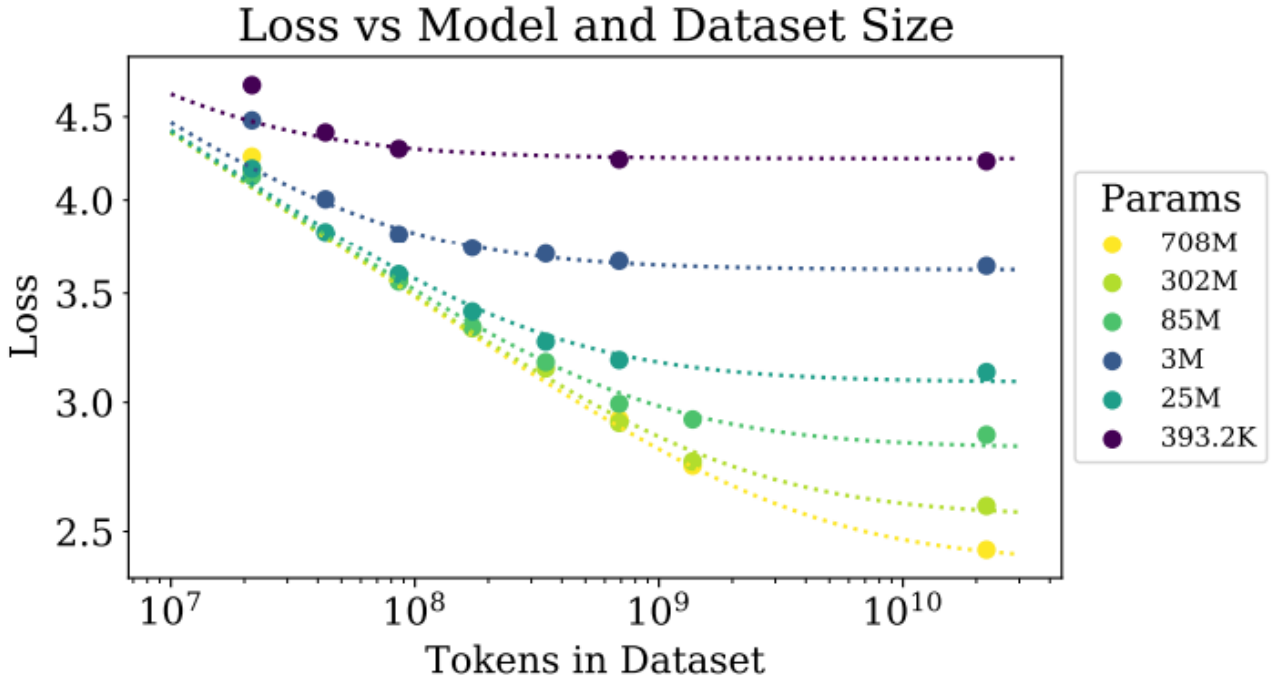
$$L(N) = (N_c/N)^{\alpha_N}\ ;\ \ \alpha_N \sim 0.076, \quad N_c \sim 8.8 \times 10^{13}$$

Scaling: **Interactions**

What about the effect of both model size and data?

- **Why?** Need to figure out what to prioritize: get more data or increase the model size?
  - "as we increase the model size, we should increase the dataset size sublinearly according to $D \propto N^{\alpha\_N/\alpha\_D} \sim N^{0.74}$"

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



SL2 **Conclusion**

Note all results fairly similar:

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

“All three approaches suggest that as compute budget increases, model size and the amount of training data should be increased in approximately equal proportions”

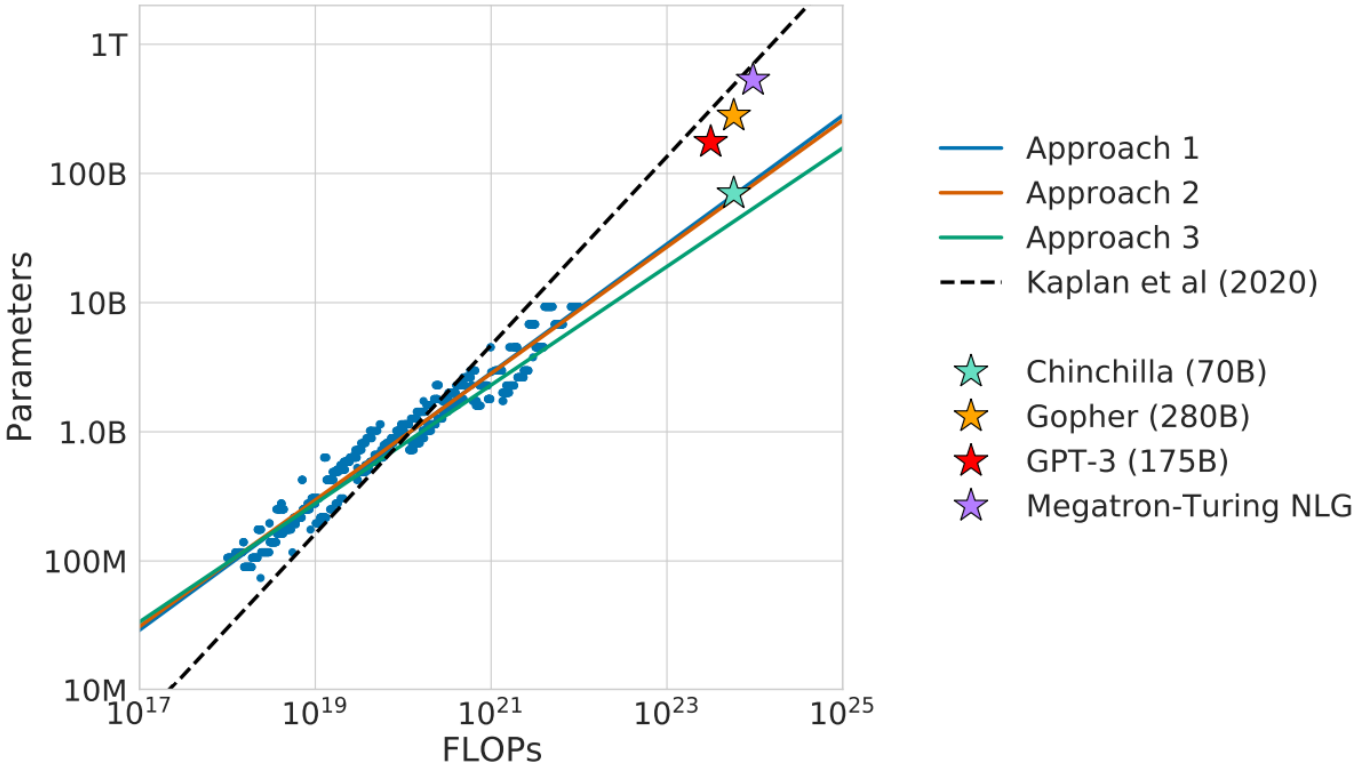
- Quite different from Kaplan et al!

SL2 **Chinchilla**

What are the implications?

- For a particular (large) compute budget, very massive models are not the way to go,
- "**Smaller**" is better.
- Chinchilla model: 70B parameters, 1.4T tokens
  - Comparison against Gopher: same compute in FLOPs, but much larger

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%*



Back to Universality

Even if we could estimate these law parameters correctly, are we stuck with the implications?

- Maybe not!

- Better **data** via pruning

Beyond neural scaling laws:
beating power law scaling via data pruning

Ben Sorscher[*1] **Robert Geirhos**[*2] **Shashank Shekhar**[3]

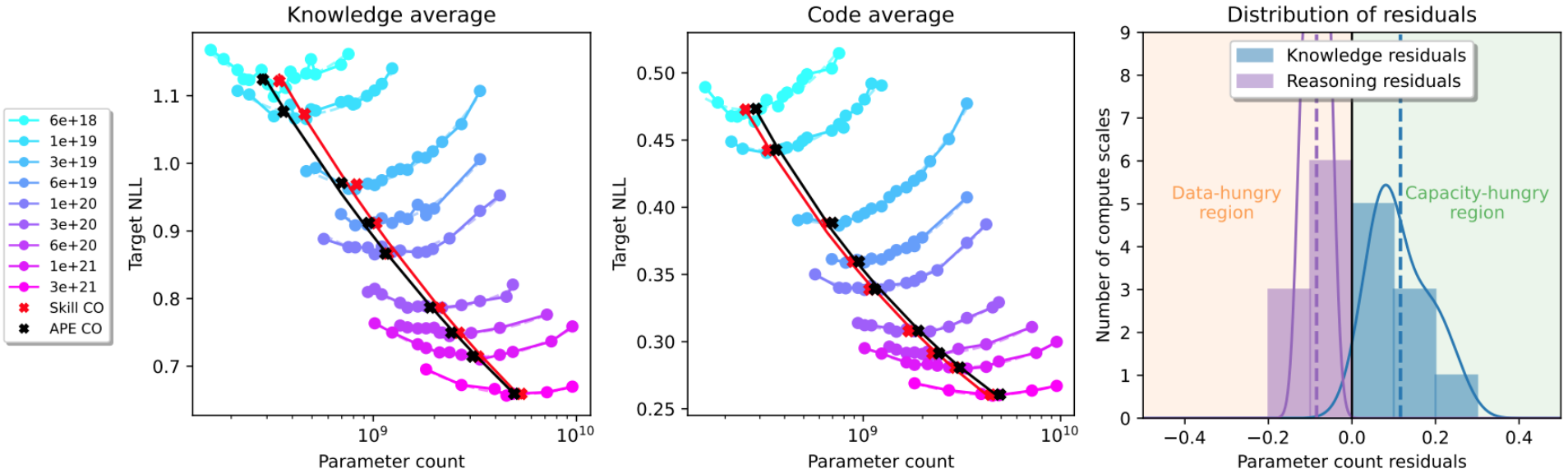
Surya Ganguli[1,3§] **Ari S. Morcos**[3§]

[*]equal contribution
[1]Department of Applied Physics, Stanford University
[2]University of Tübingen
[3]Meta AI (FAIR)
[§]Joint senior authors

Beyond Single Data Types

Do all data/tasks show the same scaling?

- Maybe not!
- Reasoning versus knowledge:
  - Knowledge: more **data-hungry**, reasoning: more **compute-hungry**

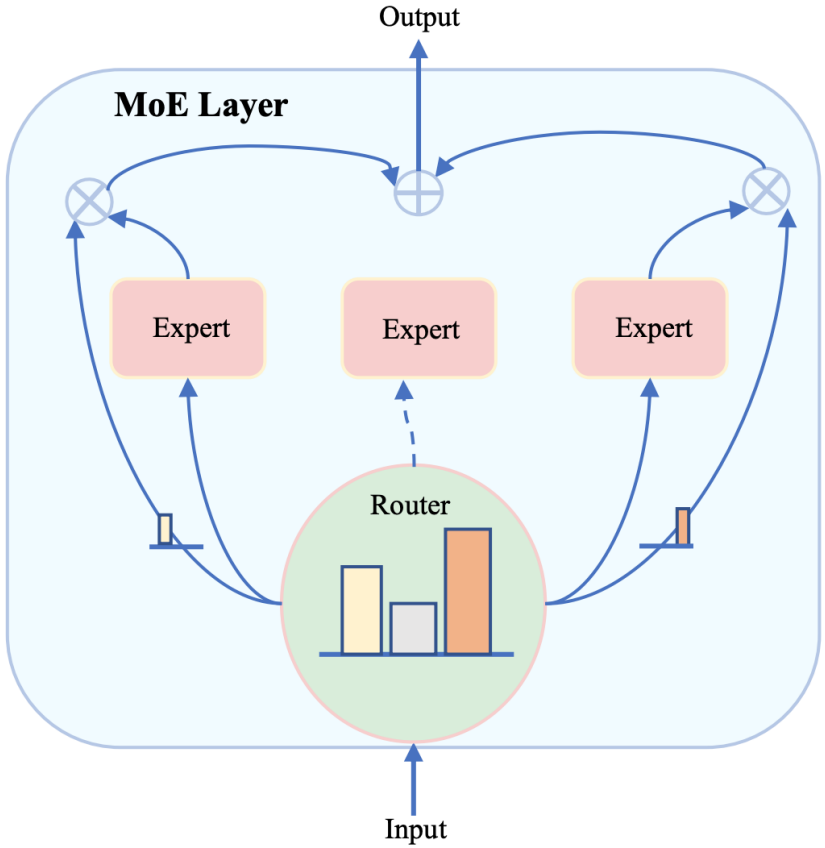


(a) Knowledge QA. (b) Code. (c) Data- vs capacity-hunger.

Beyond Standard Architectures: MoE

Another approach: modify architecture to decouple scaling from inference-time costs

- A popular approach: mixture of experts (MoE)
- Router points to path to take
- ***Active*** parameters much smaller than ***full model-size***: can scale up models further



Break & Questions

Outline

- **Scaling Laws Review & Breaking Beyond Laws**
  - Laws, Chinchilla-style compute optimality, data pruning, mixture-of-experts, etc.
- **Test-Time Scaling**
  - Motivation, taxonomy and forms of test-time scaling, aggregation and verification
- **Combined Scaling**
  - One example of combined training and test-time scaling recipes

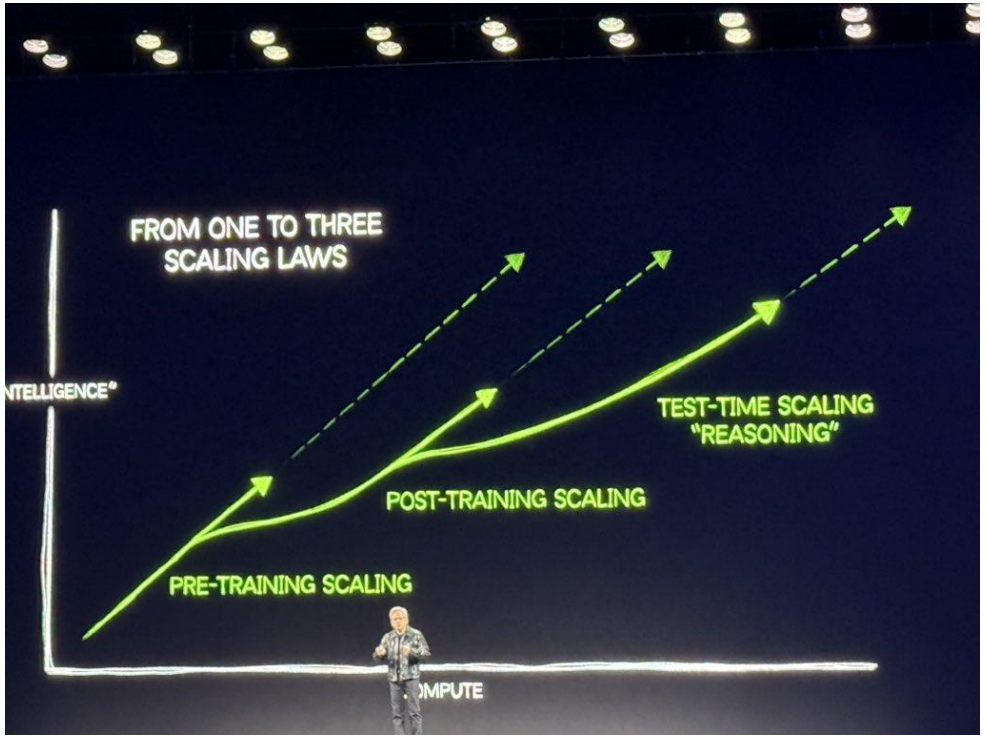
Test-Time Scaling

Basic idea: we can **choose** where to invest compute. Can improve performance by

- Scaling models at **training time** (where we used scaling laws)
  - E.g., bigger model, more data, more compute

- Or… keep the trained model

  and use it more → **test-time scaling**
  - Ask questions repeatedly
  - Reason/think longer
  - Search and other techniques



Test-Time Scaling

Basic idea: we can **choose** where to invest compute. Can improve performance by

- Or… keep the trained model and use it more → **test-time scaling**
- Many different techniques! We'll cover a handful of them
- Several good surveys; a good choice:

Zhang et al '25: "A Survey on Test-Time Scaling in Large Language Models: What, How, Where, and How Well"

A Survey on Test-Time Scaling in Large Language Models: What, How, Where, and How Well

Qiyuan Zhang[1*], Fuyuan Lyu[2*], Zexu Sun[3†], Lei Wang[5†], Weixu Zhang[2†], Wenyue Hua[8†], Haolun Wu[2,7†], Zhihan Guo[4†], Yufei Wang[6‡], Niklas Muennighoff[7], Irwin King[4], Xue Liu[2], Chen Ma[1]

[1]City University of Hong Kong, [2]McGill University & MILA, [3]Gaoling School of Artificial Intelligence, Renmin University of China, [4]Chinese University of Hong Kong, [5]Salesforce AI Research, [6]Macquarie University, [7]Stanford University, [8]University of California, Santa Barbara
qzhang732-c@my.cityu.edu.hk, fuyuan.lyu@mail.mcgill.ca

Page: https://testtimescaling.github.io/

Abstract

As enthusiasm for scaling computation (data and parameters) in the pretraining era gradually diminished, test-time scaling (*TTS*)—also referred to as "test-time computing"—has emerged as a prominent research focus. Recent studies demonstrate that *TTS* can further elicit the problem-solving capabilities of large language models (LLMs), enabling significant breakthroughs not only in specialized reasoning tasks, such as mathematics and coding, but also in general tasks like open-ended Q&A. However, despite the explosion of recent efforts in this area, there remains an urgent need for a comprehensive survey offering systemic understanding. To fill this gap, we propose a unified, hierarchical framework structured along four core dimensions of *TTS* research: ***what to scale***, ***how to scale***, ***where to scale***, and ***how well to scale***. Building upon this taxonomy, we conduct an extensive review of methods, application scenarios, and assessment aspects, and present an organized decomposition that highlights the unique contributions of individual techniques within the broader *TTS* landscape. From this analysis, we distill the major developmental trajectories of *TTS* to date and offer hands-on guidelines for practical deployment. Furthermore, we identify several open challenges and offer insights into promising future directions, including further scaling, clarifying the functional essence of techniques, generalizing to more tasks, and more attributions. Our repository is available on https://github.com/testtimescaling/testtimescaling.github.io/.

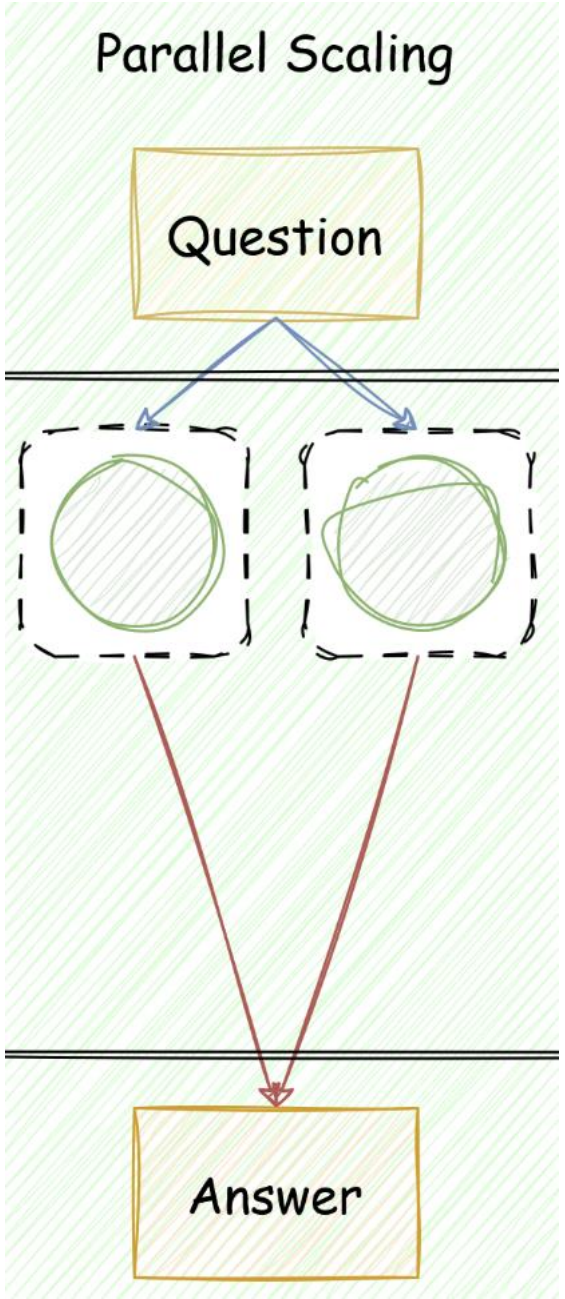
Parallel Scaling

The form of test-time scaling we've talked about already.

- **Repeatedly** try out the same problem

- This can be done in **parallel**, since none of the solutions depend on each other

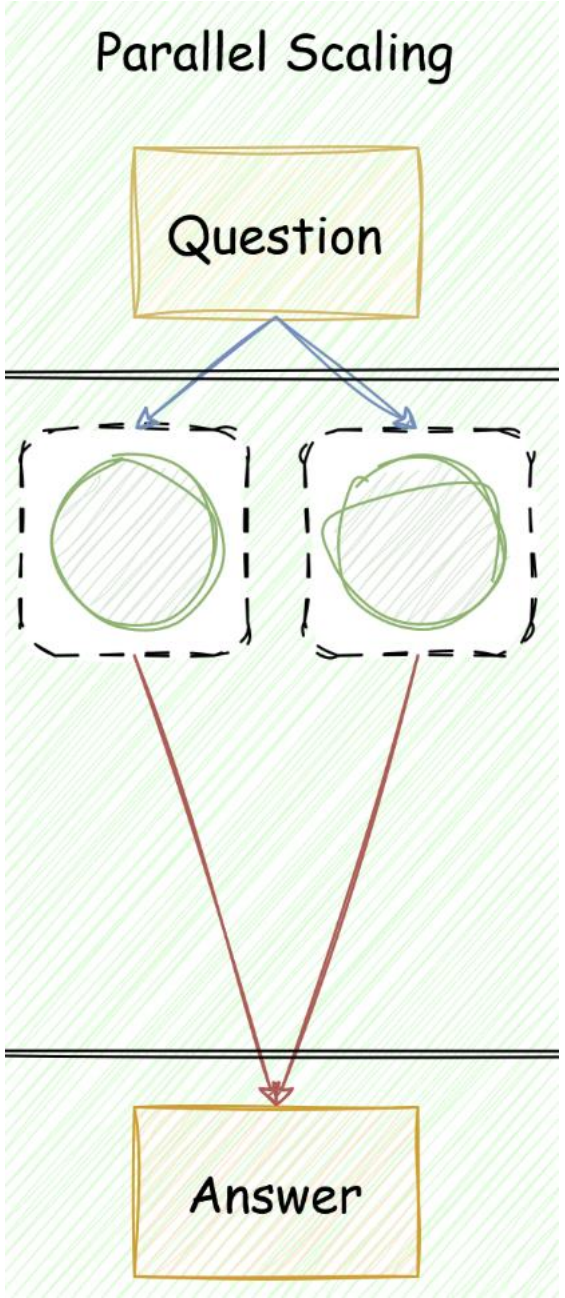
- Must **aggregate** the solution at the end



Parallel Scaling

Aggregation techniques

- If we have **nothing else**, try majority vote / self-consistency
- If we have a **verifier**, check solutions until verifier confirms a correct one
- Even without a perfect verifier, we can use a **reward model**, or **LLM-as-a-judge** to perform selection/aggregation
- Note: don't have to just choose, can also combine! (we'll see an example soon)



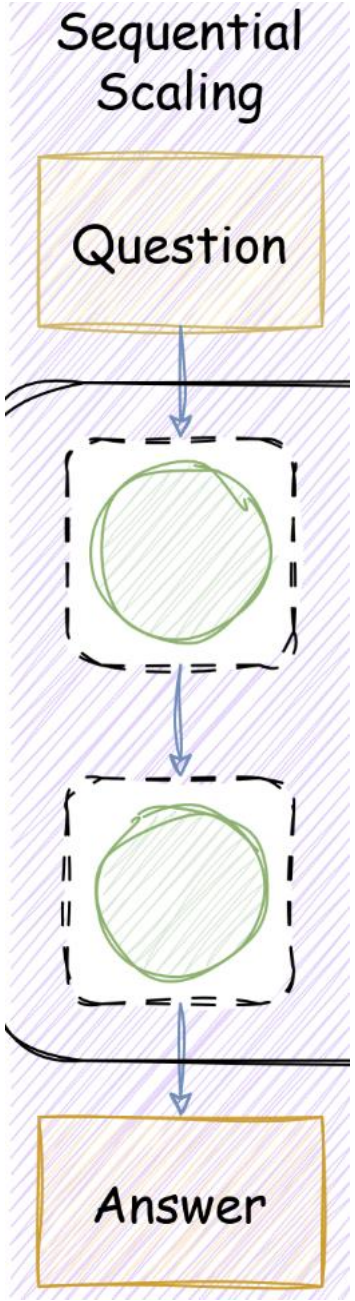
Sequential Scaling

Later computations are based on intermediate steps

- Not too different from chain-of-thought

We can

- **Append** new steps,
- **Break down problems** further,
- **Reflect/refine/iterate** existing steps,
- Etc.

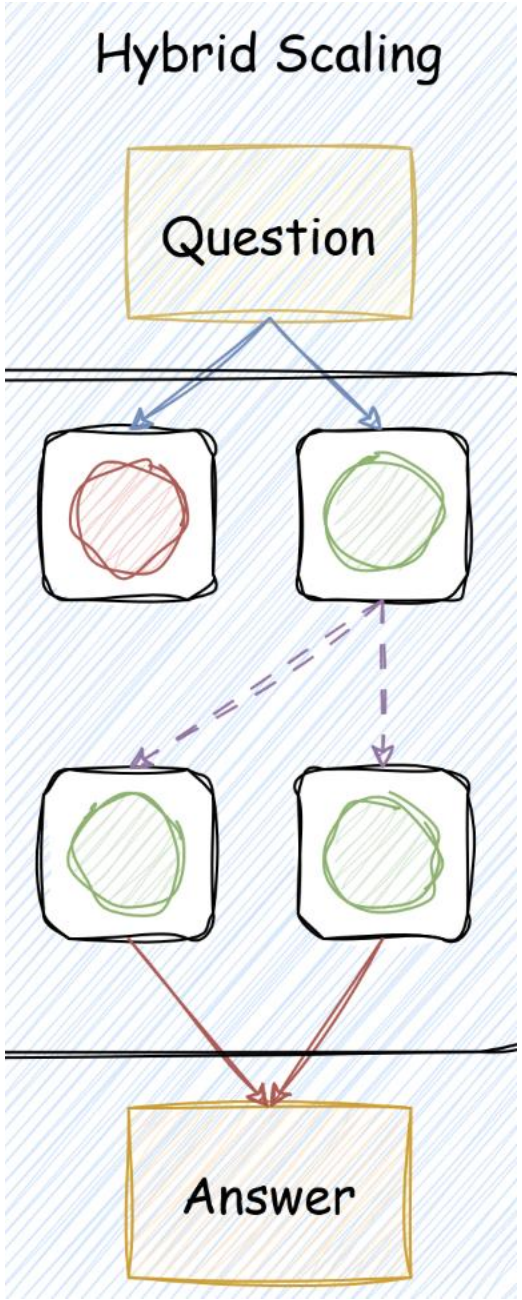


Hybrid Scaling

Combination of the **sequential** and **parallel**,

- E.g., do some steps/solutions in parallel,
  - Pick one out, continue from there
  - Can still aggregate at the end

- Just as CoT was a type of early sequential scaling, tree-of-thoughts is a form of hybrid scaling



Hybrid Scaling

Note that all of our aggregation methods work

- One example: MoA (mixture-of-agents)

Parallel+ sequential scaling, with final **aggregator model** (Wang et al '24)

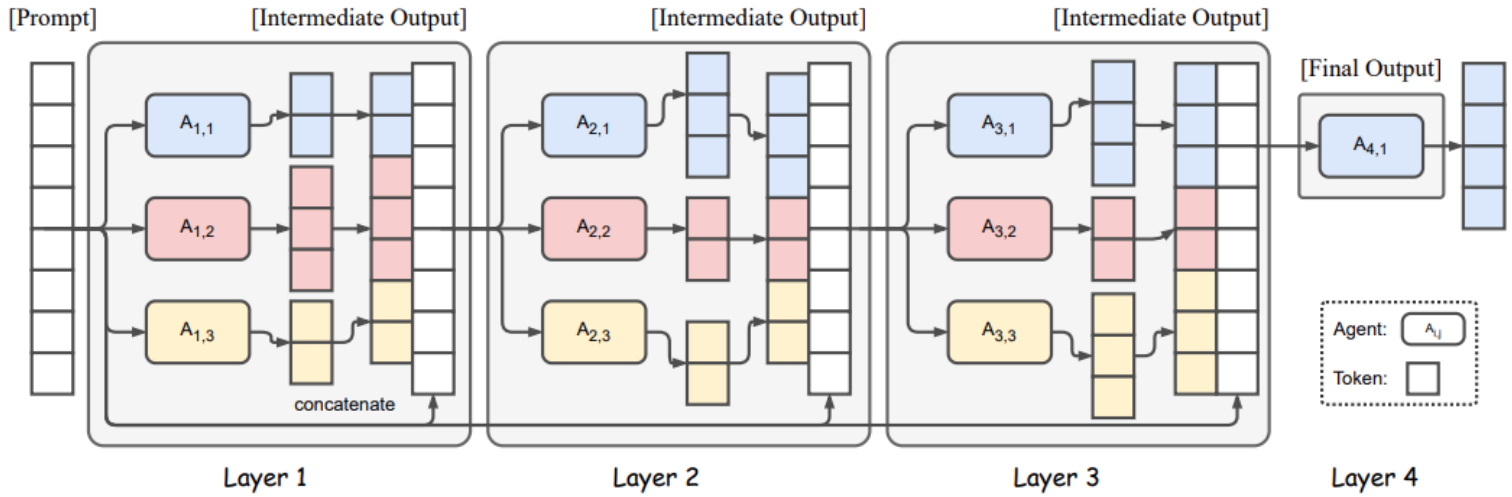
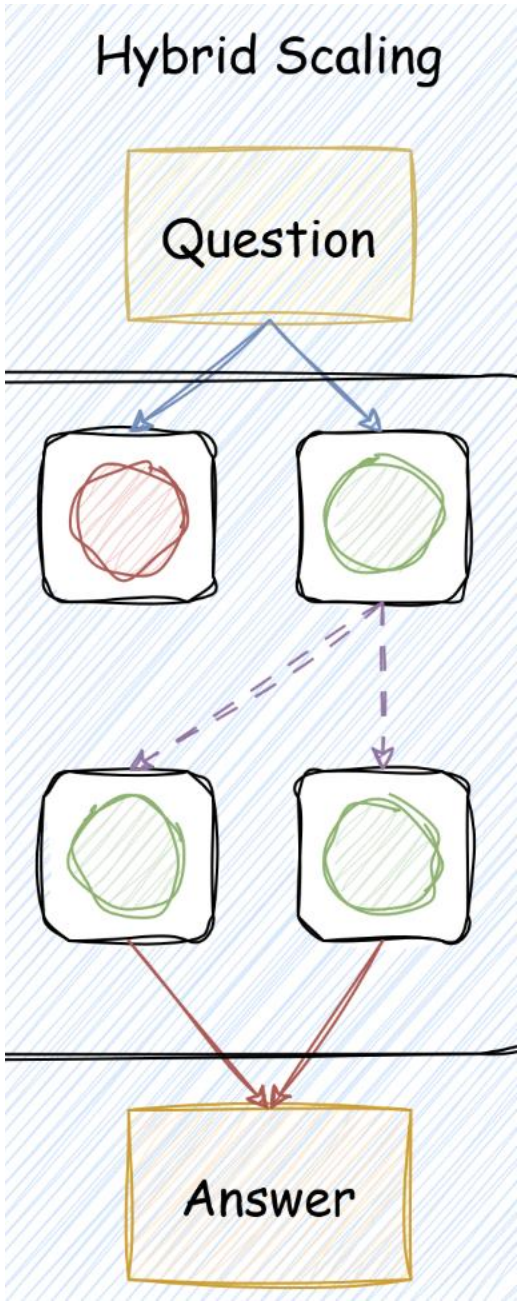


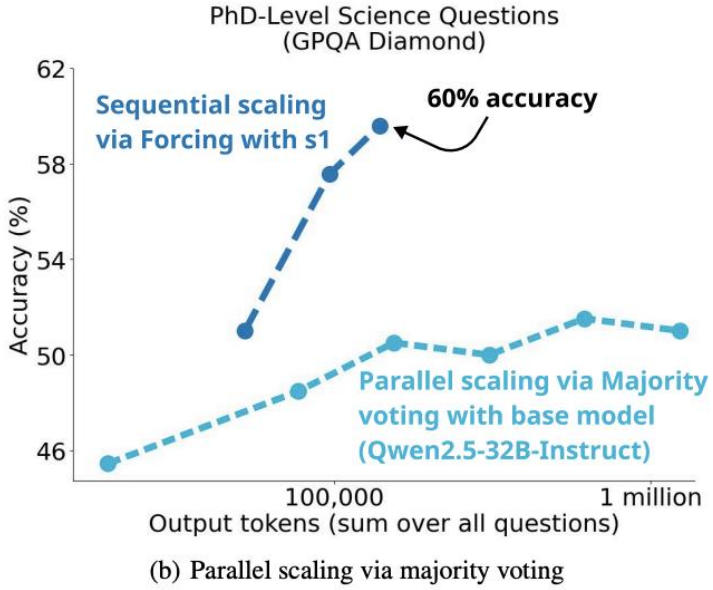
Figure 2: Illustration of the Mixture-of-Agents Structure. This example showcases 4 MoA layers with 3 agents in each layer. The agents here can share the same model.



“Internal” Scaling

Control how much reasoning is happening inside the model

- A good example: **S1** paper
  - Budget “forcing”: ensure a minimum & maximum amount of thinking per problem



(b) Parallel scaling via majority voting

s1: Simple test-time scaling

Niklas Muennighoff [*134] Zitong Yang [*1] Weijia Shi [*23] Xiang Lisa Li [*1] Li Fei-Fei [1] Hannaneh Hajishirzi [23]
Luke Zettlemoyer [2] Percy Liang [1] Emmanuel Candès [1] Tatsunori Hashimoto [1]

Abstract

Test-time scaling is a promising new approach to language modeling that uses extra test-time compute to improve performance. Recently, OpenAI's o1 model showed this capability but did not publicly share its methodology, leading to many replication efforts. We seek the simplest approach to achieve test-time scaling and strong reasoning performance. First, we curate a small dataset **s1K** of 1,000 questions paired with reasoning traces relying on three criteria we validate through ablations: difficulty, diversity, and quality. Second, we develop budget forcing to control test-time compute by forcefully terminating the model's thinking process or lengthening it by appending "Wait" multiple times to the model's generation when it tries to end. This can lead the model to double-check its answer, often fixing incorrect reasoning steps. After supervised finetuning the Qwen2.5-32B-Instruct language model on **s1K** and equipping it with budget forcing, our model **s1-32B** exceeds o1-preview on competition math questions by up to 27% (MATH and AIME24). Further, scaling **s1-32B** with budget forcing allows extrapolating beyond its performance without test-time intervention: from 50% to 57% on AIME24. Our model, data, and code are open-source at https:

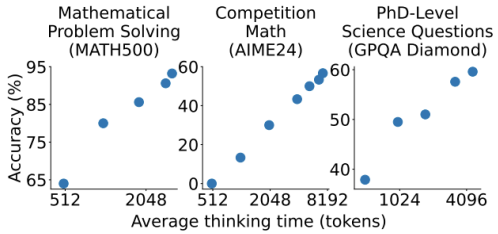


Figure 1. **Test-time scaling with s1-32B.** We benchmark **s1-32B** on reasoning-intensive tasks and vary test-time compute.

of this approach is to increase the compute at test time to get better results. There has been much work exploring this idea (Snell et al., 2024; Welleck et al., 2024), and the viability of this paradigm was recently validated by OpenAI o1 (OpenAI, 2024). o1 has demonstrated strong reasoning performance with consistent gains from scaling test-time compute. OpenAI describes their approach as using large-scale reinforcement learning (RL) implying the use of sizable amounts of data (OpenAI, 2024). This has led to various attempts to replicate their models relying on techniques like Monte Carlo Tree Search (Gao et al., 2024b; Zhang et al., 2024a), multi-agent approaches (Qin et al., 2024), and others (Wang et al., 2024a; Huang et al., 2024b; 2025). Among

“Internal” Scaling

Control how much reasoning is happening inside the model

- More broadly, this is done at training time
- We’ve talked about RL on verifiable outputs, but another way to help the model reason longer is to train (SFT or otherwise) on long reasoning traces
  - These can come from other, stronger models
  - DeepSeek-R1 is an example that was warm-started this way + various **distilled** models also do this
  - **Distilling**: taking a larger model’s outputs or other information and using it to train a smaller model to reproduce capabilities

Break & Questions

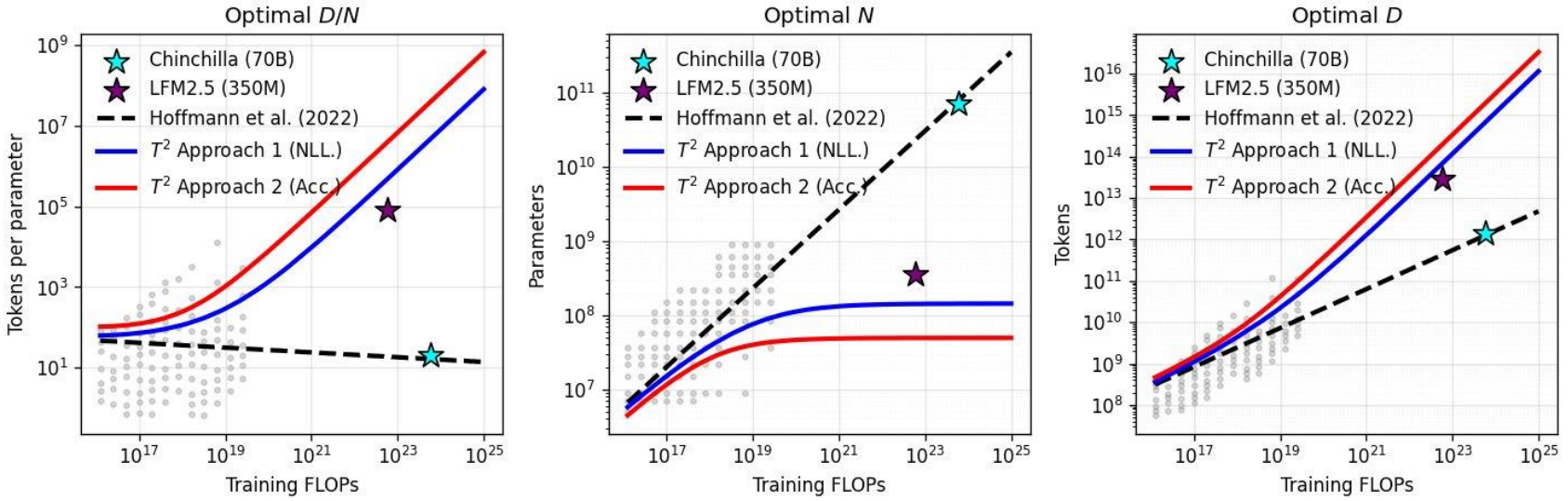
Outline

- **Scaling Laws Review & Breaking Beyond Laws**
  - Laws, Chinchilla-style compute optimality, data pruning, mixture-of-experts, etc.
- **Test-Time Scaling**
  - Motivation, taxonomy and forms of test-time scaling, aggregation and verification
- **Combined Scaling**
  - One example of combined training and test-time scaling recipes

Combining Scaling Laws and Test-Time Scaling

We can also do combinations that produce more **general recipes**

- So: if we expect a certain level of test-time scaling, we can adjust train-time scaling laws to make up for it
- Some early conclusions (full disclosure: this is our paper) we should train longer than Chinchilla makes predicts



Roberts et al '26

Thank You!