Test-Time Scaling Makes Overtraining Compute-Optimal

Nicholas Roberts[μ*] **Sungjun Cho**[μ] **Zhiqi Gao**[μ] **Tzu-Heng Huang**[μ] **Albert Wu**[μ]
Gabriel Orlanski[μ] **Avi Trost**[μ] **Kelly Buchanan**[σ] **Aws Albarghouthi**[μ] **Frederic Sala**[μ]
[μ]University of Wisconsin-Madison [σ]Stanford University

Abstract

Modern LLMs scale at test-time, e.g. via repeated sampling, where inference cost grows with model size and the number of samples. This creates a trade-off that pretraining scaling laws, such as Chinchilla, do not address. We present Train-to-Test (T^2) scaling laws that jointly optimize model size, training tokens, and number of inference samples under fixed end-to-end budgets. T^2 modernizes pretraining scaling laws with pass@k modeling used for test-time scaling, then jointly optimizes pretraining and test-time decisions. Forecasts from T^2 are robust over distinct modeling approaches: measuring joint scaling effect on the task loss and modeling impact on task accuracy. Across eight downstream tasks, we find that when accounting for inference cost, optimal pretraining decisions shift radically into the overtraining regime, well-outside of the range of standard pretraining scaling suites. We validate our results by pretraining heavily overtrained models in the optimal region that T^2 scaling forecasts, confirming their substantially stronger performance compared to pretraining scaling alone. Finally, as frontier LLMs are post-trained, we show that our findings survive the post-training stage, making T^2 scaling meaningful in modern deployments.

1 Introduction

Pretraining scaling laws tell us how to optimally train language models, but not how to deploy them (Kaplan et al., 2020; Hoffmann et al., 2022). Test-time scaling laws tell us how to optimally allocate compute at deployment, but not how to train models (Snell et al., 2024; Brown et al., 2025). The two have developed largely in *isolation*, yet are fundamentally coupled. Model size and training duration determine both the quality and cost of inference samples. Models designed to reason through frontier research problems will be sampled from hundreds or thousands of times (Jaech et al., 2024; Guo et al., 2025); these should be trained differently from chat models that instantly answer everyday questions.

Should parameter and token counts change if you know how your model will be used at test time? In practice, Chinchilla (Hoffmann et al., 2022) scaling laws guide the allocation of pretraining compute for flagship models. However, modern model releases are families spanning a range of sizes (Touvron et al., 2023; Groeneveld et al., 2024; Qwen et al., 2024), with the lower end intentionally *overtrained* well beyond Chinchilla-optimal ratios to reduce per-query inference cost. This makes them natural candidates for test-time scaling, yet nothing connects pretraining decisions to this inference strategy. **No existing scaling law captures the core tradeoff**: smaller models are cheaper per sample but weaker per sample, and the benefit of repeated sampling is a highly nonlinear function of per-sample quality.

Unifying pretraining and inference scaling is challenging because the two regimes operate under fundamentally different evaluation criteria. Pretraining is evaluated using the loss, a smooth, continuous quantity. Test-time scaling, by contrast, is evaluated through downstream task metrics such as pass@k—the probability of producing at least one correct answer in k independent attempts. Should a unified scaling law across pretraining and test-time scaling model the loss or model the pass@k accuracy?

*Corresponding author: nr2756@princeton.edu. Work done while at UW–Madison.

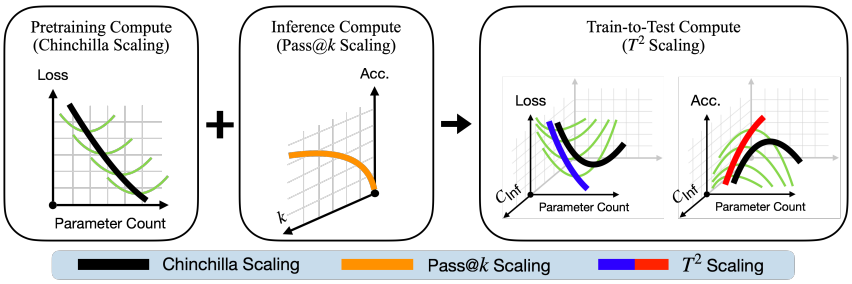

Figure 1: Our T^2 scaling laws combine Chinchilla scaling for pretraining with pass@k modeling for test-time scaling via repeated sampling to obtain optimal pretraining allocations subject to a test-time scaling budget. T^2 recommends overtraining compared to Chinchilla.

Prior work has addressed pieces of this problem but not the whole. Sardana et al. (2023) extends Chinchilla to account for inference cost, but considers only the aggregate volume of *single-pass* serving instead of the multiplicative cost and performance gains from repeated sampling. Recent studies empirically show that allocating more inference compute to smaller models via repeated sampling can match or exceed the performance of larger ones (Brown et al., 2025; Snell et al., 2024), but they treat pretrained models as given and do not address how they should have been trained. Schaeffer et al. (2026) develop scaling laws that predict pass@k from pretraining compute, but treat this as forecasting rather than an optimization problem—they predict what performance *will be* for a given model, not what model *should be* trained for a given budget. No existing work jointly optimizes model size, training duration, and the number of inference samples under a single compute budget.

In this work, we close the loop between pre-training and test-time scaling. We propose **Train-to-Test (T^2) scaling laws** that predict performance as a function of model size N, training tokens D, and number of samples k, and optimize over all three under a total compute budget that includes both training ($6ND$) and inference ($2Nk$) cost. Following Chinchilla, we evaluate multiple modeling approaches: whether to model the loss or pass@k as functions of N, D, and k. Although the two approaches are quite different, we find that they agree closely: both suggest substantial overtraining and test-time scaling across our evaluations. We build on an existing set of Chinchilla scaling checkpoints from Porian et al. (2024), extending it into the overtrained regime and assembling a testbed of over 100 models across 12 compute levels spanning three orders of magnitude.

Using T^2 scaling laws, we find that ***optimal pretraining decisions shift radically into the overtraining regime*** when considering test-time compute. When we correct for the cost of repeated sampling, the optimal model is substantially smaller and more overtrained than what Chinchilla prescribes. Our evaluation spans eight tasks covering knowledge, reasoning, and language understanding, on which we investigate three research questions:

RQ1 Should pretraining change if you know your test-time scaling budget? Yes—T^2 scaling consistently recommends small overtrained models. (§4.1)

RQ2 Does T^2 extrapolate to overtrained checkpoints? Yes—we overtrain models from scratch and show that they consistently outperform Chinchilla checkpoints. (§4.2)

RQ3 Does T^2 scaling survive post-training? Yes—we find that compute-optimal trade-offs derived from base models persist after supervised fine-tuning. (§4.3)

To answer these questions, we make the following contributions:

Contributions

- **End-to-end scaling:** We formalize train-to-test scaling as a joint optimization over model size N, dataset size D, and inference compute k under train and test budgets.
- **Loss and accuracy scaling:** We introduce two complementary approaches: (i) loss- and (ii) accuracy-based formulations that explicitly incorporate inference cost.
- **Validation on overtrained checkpoints:** We train models in the predicted overtrained regime and show improved performance under a range of fixed inference budgets.
- **Interactions with post-training:** The predictions from our scaling approach persist after post-training, even though overtrained models are harder to fine-tune.

2 Background

Our work connects two important areas: (i) pretraining scaling laws and (ii) test-time sampling strategies after deployment. We begin with their setups, and then dive into our new modeling techniques. A summary of additional related work can be found in Appendix A.

Chinchilla scaling laws for pretraining. The Chinchilla scaling law (Hoffmann et al., 2022) models the pretraining loss as a function of finite model capacity N and dataset size D (number of training tokens): $L(N, D) = E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}$, where E represents an irreducible loss floor fit for the given data distribution and evaluation setup while the remaining terms capture reducible contributions from N and D. The parameters A, B, α, β, and E are all non-negative and are fit empirically from a grid of training runs. Here, the loss is assumed to be the negative log-likelihood (NLL) over the data distribution: $\mathbb{E}_{(x,y)\sim\mathcal{D}}[-\log(p(y|x))]$ with $p(y|x)$ being the probability assigned by the model. Given a pretraining budget $C_{\text{train}} \approx 6ND$, the *compute-optima* minimize L subject to this constraint, yielding $N^*(C_{\text{train}}) \propto C_{\text{train}}^a$ and $D^*(C_{\text{train}}) \propto C_{\text{train}}^b$ with $a \approx b \approx 0.5$. That is, the optimal model size and training tokens should scale at similar rates as a function of the pretraining compute budget.

Pass@k estimation for test-time scaling. The standard metric for evaluating repeated sampling is pass@k: draw k independent samples from a model and succeed if *any* sample is correct. For a single problem i with per-sample success probability p_i, the probability of at least one answer in k attempts being correct is pass@$k_i = 1 - (1 - p_i)^k$. Aggregating over a benchmark $\mathcal{D}$ of M problems gives the expected pass@k:

$$\text{pass@}k_{\mathcal{D}} = \mathbb{E}_{i\sim\mathcal{D}}\left[\text{pass@}k_i\right] = \frac{1}{M}\sum_{i=1}^{M}\left[1 - (1 - p_i)^k\right].$$

3 Estimating Optimal Pretraining Allocations for Test-Time Scaling

We present two modeling approaches for T^2 scaling that answer our central research question: **should choices made during pretraining change if you know your test-time scaling budget?** In our first approach, we model the impact of repeated sampling on the loss by fitting a parametric function of the negative log pass@k. In our second approach, we model the pass@k accuracy directly by composing Chinchilla scaling with a pass@k estimator. In §4, we show that our findings are robust across both approaches. Finally, once we establish these two approaches, we answer our main research question by standardizing the test-time scaling budget: using more repeated samples for smaller models and fewer for larger models. Standardizing the inference budget of test-time scaling across checkpoints allows us to see how optimal pretraining decisions shift in light of test-time scaling considerations. **If the optimal pretraining decisions (model size and the number of training tokens) shift compared to those recommended by standard Chinchilla scaling, then the answer to RQ1 is yes: pretraining decisions should change if you know your test-time scaling budget.**

We first describe the optimization objectives of our T^2 approaches. Given a compute budget for training (C_{train}) and inference (C_{inf}), the optimization problem in terms of the NLL is:

$$\min_{N,D,k} \; L(N, D, k) \qquad \text{s.t.} \quad 6ND \leq C_{\text{train}} \;\text{ and }\; 2Nk \leq C_{\text{inf}}, \tag{1}$$

or similarly, in terms of the pass@k accuracy:

$$\max_{N,D,k} \; \text{Acc}(N, D, k) \qquad \text{s.t.} \quad 6ND \leq C_{\text{train}} \;\text{ and }\; 2Nk \leq C_{\text{inf}}. \tag{2}$$

$L(N, D, k)$ and $\text{Acc}(N, D, k)$ represent the aggregated NLL and accuracy respectively, as functions of model capacity N, dataset size D, and number of sampling attempts k.

3.1 Approach 1: T^2 as a Parametric Model of the Task Loss

Our first approach models the loss as a function of the parameter count N, training tokens D, and the number of repeated samples k used at test-time in order to optimize Equation 1.

First, in order to make repeated sampling compatible with the negative log likelihood (NLL), we rewrite the single-sample probability in terms of the probability that the target outcome is obtained at least once under k repeated samples, following prior work on pass@k (Chen et al., 2021; Brown et al., 2025; Ehrlich et al., 2025; Schaeffer et al., 2025). That is, working with the definition of pass@k_i allows us to define the corresponding NLL-style objective under repeated sampling as

$$\mathbb{E}_{i\sim\mathcal{D}_{\text{task}}}[-\log \text{pass@}k_i] = \mathbb{E}_{i\sim\mathcal{D}_{\text{task}}}\left[-\log\left(1-(1-p_i)^k\right)\right],$$

where $\mathcal{D}_{\text{task}}$ is a distribution over samples i representing a downstream task.

With this in place, we can model the negative log pass@k as an extension of the Chinchilla scaling law, $\widehat{L}(N,D)$ by adding a power-law term in k:

$$\widehat{L}(N,D,k) = \widehat{L}(N,D) + \frac{G}{k^\gamma} = E + \frac{A}{N^\alpha} + \frac{B}{D^\beta} + \frac{G}{k^\gamma}.$$

We choose this model because prior work has found that the negative log pass@k contribution from k yields power law scaling[1] under an assumption that the task difficulty distribution can be modeled by a Beta distribution, which has been found to hold in practice (Brown et al., 2025; Schaeffer et al., 2025). This has convenient properties when combined with the other power law terms in N and D in the Chinchilla scaling law:

First, when $k = 1$, we recover standard Chinchilla scaling:

$$\widehat{L}(N,D,1) = E' + \frac{A}{N^\alpha} + \frac{B}{D^\beta} = \widehat{L}(N,D),$$

where $E' = E + G$ absorbs the additional constant. Second, a property of Chinchilla scaling is that as $N, D \to \infty$, the model approaches the 'irreducible loss' term E. Given its power law form, this is still true when k approaches infinity alongside N and D.

3.2 Approach 2: T^2 as a Parametric Model of the Task Accuracy

While the previous model is simple, it trades off interpretability—practitioners often value pass@k forecasts due to their interpretation as the likelihood of solving a problem given a certain compute investment. Our second approach addresses this by modeling the pass@k directly as an accuracy-like metric as a function of N, D, and k, which optimizes Equation 2.

A naive approach to modeling pass@k might be to begin with $\widehat{L}(N,D)$, and simply map the NLL to accuracy p for the same task, then compute pass@$k = 1-(1-p)^k$. Prior work has shown that the relationship between the mean NLL and the mean accuracy can be well approximated using a fitted sigmoid (Grattafiori et al., 2024). In other words, we can model the mean single-pass task accuracy, $\mathbb{E}_{\mathcal{D}_{\text{task}}}[\text{Acc}(N,D)]$, as $\sigma_\theta(\widehat{L}(N,D))$ with a parameterized sigmoid σ_θ fit to pairs of NLL and accuracy values on the task distribution across the model population. So this naive model of the pass@k might take the following form:

$$\widehat{\text{Acc}}_{\text{naive}}(N,D,k) = 1-(1-\sigma_\theta(L(N,D)))^k.$$

However, our goal is instead to obtain an estimator of the *mean pass@k accuracy*, $\mathbb{E}_{\mathcal{D}_{\text{task}}}[\text{Acc}(N,D,k)]$ that depends on the scaling parameters, rather than the single-pass accuracy, so this naive model overestimates due to the concavity of the pass@k:

$$\begin{aligned}1-(1-\mathbb{E}_{\mathcal{D}_{\text{task}}}[\text{Acc}(N,D)])^k &\geq \mathbb{E}_{\mathcal{D}_{\text{task}}}[1-(1-\text{Acc}(N,D))^k]\\ &= \mathbb{E}_{\mathcal{D}_{\text{task}}}[\text{Acc}(N,D,k)].\end{aligned}$$

[1] By Jensen's inequality, our NLL-style objective acts as an upper-bounding surrogate on the negative log expected pass@k, which scales as a power law (we minimize the expected negative log pass@k). Therefore, minimizing our surrogate minimizes the quantity of interest.

A simple way to avoid overestimating the pass@k would be to directly use the per-question probabilities from model likelihoods, which would allow us to compute the mean pass@k exactly. However, our goal is a scaling law, a parametric model that can forecast pass@k at unevaluated (N, D, k) configurations. This requires us to model the *distribution* of per-question probabilities and how this distribution varies with model size and training tokens.

Intuitively, we want to account for the natural spread of difficulty between tasks in our data distribution. We do this by modeling the per-question single-pass accuracies as a Beta distribution, following prior work (Kazdan et al., 2025). We model $\mathrm{Acc}(N, D) \sim \mathrm{Beta}(a_{N,D}, b_{N,D})$, and parameters $a_{N,D}$ with $b_{N,D}$ related to N and D via the NLL, which we model as a Beta regression problem. Using the mean (μ) and sample size (ν) parameterization of the Beta distribution, we model $\mu \in (0, 1)$ and $\nu \in (0, \infty)$ using standard link functions from Beta regression: a logit link for the mean (which we rescale with an additional parameter), and a log link for the sample size. We relate this to the loss by using the Chinchilla loss estimate as our linear predictor. This yields the following parameterization of $a_{N,D}$ and $b_{N,D}$:

$$\begin{aligned}
\mu_{N,D} &= \sigma_\theta(\widehat{L}(N, D)) = \frac{\theta_2}{1 + \exp\big(\theta_1 \cdot (\widehat{L}(N, D) - \theta_0)\big)}, \\
\nu_{N,D} &= \exp(\theta_3 + \theta_4 \cdot \widehat{L}(N, D)), \\
a_{N,D} &= \mu_{N,D}\nu_{N,D}, \\
b_{N,D} &= (1 - \mu_{N,D})\nu_{N,D}.
\end{aligned}$$

Finally, using this model of the single-pass accuracy, we obtain the following pass@k model via properties of the Beta distribution:[2]

$$\begin{aligned}
\widehat{\mathrm{Acc}}(N, D, k) &= \mathbb{E}_{\mathrm{Acc}(N,D)\sim\mathrm{Beta}(a_{N,D}, b_{N,D})}\left[1 - (1 - \mathrm{Acc}(N, D))^k\right] \\
&= 1 - \mathbb{E}_{\mathrm{Acc}(N,D)\sim\mathrm{Beta}(a_{N,D}, b_{N,D})}\left[(1 - \mathrm{Acc}(N, D))^k\right] \\
&= 1 - \frac{\mathrm{B}(a_{N,D},\, b_{N,D} + k)}{\mathrm{B}(a_{N,D},\, b_{N,D})} \\
&= 1 - \frac{\mathrm{B}(\mu_{N,D}\nu_{N,D},\, (1 - \mu_{N,D})\nu_{N,D} + k)}{\mathrm{B}(\mu_{N,D}\nu_{N,D},\, (1 - \mu_{N,D})\nu_{N,D})}.
\end{aligned}$$

3.3 Inference Cost Correction

We equalize our T^2 scaling laws over an inference budget, C_{inf}, measured as the inference FLOPs per-token served. Just as the pretraining cost, $C_{\text{train}} = 6ND$, scales multiplicatively as a function of N and the number of training tokens D, the inference budget C_{inf} scales multiplicatively in k and approximately $2N$ FLOPs for a forward pass:

$$C_{\text{inf}} = 2Nk.$$

Then for a fixed budget C_{inf}, this gives us

$$k = \frac{C_{\text{inf}}}{2N},$$

where smaller models are allocated more repeated samples compared to larger models, subject to the same inference budget. We plug this into both of our T^2 scaling approaches, which gives us our inference-corrected loss model:[3]

Approach 1

$$\widehat{L}\left(N, D, \frac{C_{\text{inf}}}{2N}\right) = \widehat{L}(N, D) + \frac{G}{k^\gamma} = E + \frac{A}{N^\alpha} + \frac{B}{D^\beta} + \frac{G}{\left(\frac{C_{\text{inf}}}{2N}\right)^\gamma},$$

[2] $\mathrm{B}(a, b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$ is the Beta function, where Γ is the Gamma function.

[3] Optimization details for fitting Approach 1 and Approach 2 can be found in Appendix F.

and our inference-corrected pass@k accuracy model:

Approach 2

$$\widehat{\text{Acc}}\left(N, D, \frac{C_{\text{inf}}}{2N}\right) = 1 - \frac{\text{B}(\mu_{N,D}\nu_{N,D},\ (1-\mu_{N,D})\nu_{N,D} + \frac{C_{\text{inf}}}{2N})}{\text{B}(\mu_{N,D}\nu_{N,D},\ (1-\mu_{N,D})\nu_{N,D})}.$$

Now for both models, we can choose an inference budget C_{inf}, and observe the pretraining decisions that optimize both the pretraining and inference budgets C_{train} and C_{inf}. We represent Approach 1 in blue and Approach 2 in red for consistency with our Figures.

Modeling inference cost. Our inference cost model is deliberately simple, following the conventions of the scaling laws literature. Just as $C_{\text{train}} = 6ND$ is a linear abstraction of training cost, $C_{\text{inf}} = 2Nk$ abstracts away factors such as model shape, memory, prompt prefill, and KV-caching. This abstraction is what makes it general. It applies to any matmul-dominated architecture, whereas optimizations like KV-caching are Transformer-specific. Notably, modeling serving efficiency gains as a discount, $(2N - \Delta)k$, would simply lead the compute-optimal decision to compensate by proportionally increasing k for smaller models, preserving our prescription toward smaller, more overtrained models. Finally, since C_{inf} is measured in FLOPs per token served, our conclusions are agnostic to generation length.

4 Experiments

In this section, we provide experimental results addressing the three research questions about our T^2 scaling approaches.[4] First, in §4.1, we show that if you know your test-time scaling budget prior to pretraining, you should overtrain significantly beyond the standard Chinchilla recommendation of 20 tokens per parameter. In §4.2, we validate our predictions against overtrained checkpoints that extend standard Chinchilla scaling suites, showing that our scaling approaches extrapolate to the optimal regions that they predict. Finally, in §4.3, we show that overtraining predictions from our T^2 approaches persist after post-training. We fit T^2 scaling to checkpoints from Porian et al. (2024), which we extend with additional overtrained checkpoints, all trained on RefinedWeb (Penedo et al., 2023).

Tasks. We evaluate T^2 across eight real and synthetic tasks that we select to be simple enough for small base models, as all of our checkpoints have fewer than 1B parameters. This is a methodological necessity rather than a limitation of our prescription: fitting scaling laws requires a checkpoint population spanning orders of magnitude in compute, and at the resulting model scales, larger standard benchmarks sit below the capability floor, where accuracy is near chance and offers no signal to fit. The real tasks that we evaluate include the OpenAI variant of LAMBADA (Paperno et al., 2016; Radford et al., 2019), ARC-Easy (Clark et al., 2018), SciQ (Johannes Welbl, 2017), and OpenBookQA (Mihaylov et al., 2018). We also evaluate on four synthetic tasks: simple knowledge recall, multi-step arithmetic reasoning, commonsense causal reasoning, and spatial reasoning, each consisting of 1,000 fill-in-the-blank or short completion questions that were generated using GPT-5 and Claude Opus 4.6. The synthetic tasks enable us to control difficulty so that per-sample accuracy is in a regime where the pass@k is informative, while spanning four distinct skill types to check that our conclusions are not an artifact of any single task; indeed, they yield the same qualitative conclusions as our real tasks (Appendix B). We provide additional task details in Appendix E. Unless otherwise noted, we present macro averaged results over all tasks.

Metric. Throughout, accuracy refers to *exact match on generated sequences* under pass@k: a sample is correct only if the model generates the reference answer. We deliberately do not use multiple-choice accuracy, and our reported values are not comparable to it—they can fall below random-choice baselines (e.g., 25% for four-way multiple-choice; see Table 1) because the model must produce the answer from its full vocabulary rather than select

[4]Our overtrained checkpoints are available here: https://hf.co/collections/nick11roberts/train-to-test-scaling-overtrained-checkpoints. Experiment code along with our generated datasets are available here: https://github.com/SprocketLab/train-to-test-scaling.

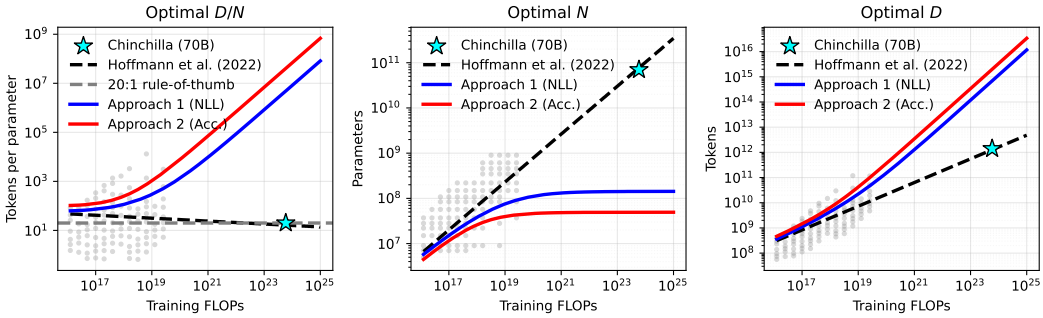

Figure 2: Optimal pretraining forecasts predicted by both T^2 approaches, compared to Hoffmann et al. (2022). **(Left)** Optimal tokens per parameter (including the 20 tokens per parameter rule of thumb used by practitioners), **(Middle)** Optimal model sizes. **(Right)** Optimal training set sizes. Both T^2 approaches forecast extreme overtraining.

among a small set of options. Multiple-choice accuracy is a poor target for a repeated sampling scaling law, as test-time scaling via repeated sampling is typically deployed over open-ended generation. Even setting this aside, restricting the output space to a small set of answer choices deceptively inflates pass@k: with c choices, per-sample success is at least $1/c$ under random guessing, so pass@k saturates after only a handful of samples (with $c = 4$, $1 - (1 - 1/4)^k > 0.99$ for $k > 16$), leaving no signal across the orders of magnitude of k that T^2 must fit. Moreover, the multiple-choice setting cannot usually be realized at deployment, where answer choices are not available a priori; deterministic cloze-formulation scoring is similarly unsuitable, as it does not change under repeated samples. Exact match instead keeps per-sample success probabilities small, which is precisely the regime where test-time scaling via repeated sampling is used in practice.

4.1 RQ1: Should Pretraining Change if You Know Your Test-Time Scaling Budget?

We evaluate **RQ1** by comparing the predictions from T^2 to Chinchilla scaling and find that if you know your test-time scaling budget, you should significantly overtrain.

Setup. We fit both T^2 approaches to a suite of 106 checkpoints ranging in size from 5M to 901M parameters trained on roughly 50M to 120B tokens. Next, we set the per-token inference budget $C_{\text{inf}} = 140$B FLOPs, or approximately the cost of a single forward pass using the 70B Chinchilla model (Hoffmann et al., 2022). Finally, to compare T^2 forecasts to Chinchilla, we extrapolate the predictions from our T^2 approaches and standard Chinchilla scaling beyond our scaling suite to 10^{25} FLOPs. Using the same fits, we visualize pretraining isoFLOP profiles for both approaches. We compare the standard single-pass setting (k=1) to the inference-corrected setting with $C_{\text{inf}} = 2 \times 10^9$ FLOPs and $k = \frac{C_{\text{inf}}}{2N}$. Each of the 12 isoFLOP curves traces out a fixed pretraining budget C_{train} by varying N and D subject to $C_{\text{train}} = 6ND$. We plot the Chinchilla optimal frontier in black and that of T^2 in red. Results are macro averaged across all eight tasks. Individual scaling fits for each task across different budgets can be found in Appendix B.

Results. Our results are shown in Figure 2 and Figure 3. Figure 2 shows that we can answer **RQ1** in the affirmative: both T^2 approaches forecast models that are **dramatically smaller and more overtrained** than what Chinchilla prescribes. We additionally confirm that the Chinchilla scaling fit is consistent with Hoffmann et al. (2022) by overlaying the 70B Chinchilla hero run model described in their paper, alongside the 20 tokens per parameter rule of thumb. Despite modeling fundamentally different quantities (NLL vs accuracy), both T^2 recommend extreme overtraining, with Approach 2 recommending more aggressive overtraining than Approach 1. Figure 3 shows isoFLOP curves under our T^2 approaches, how the overtraining trend develops within our scaling population. At every compute scale,

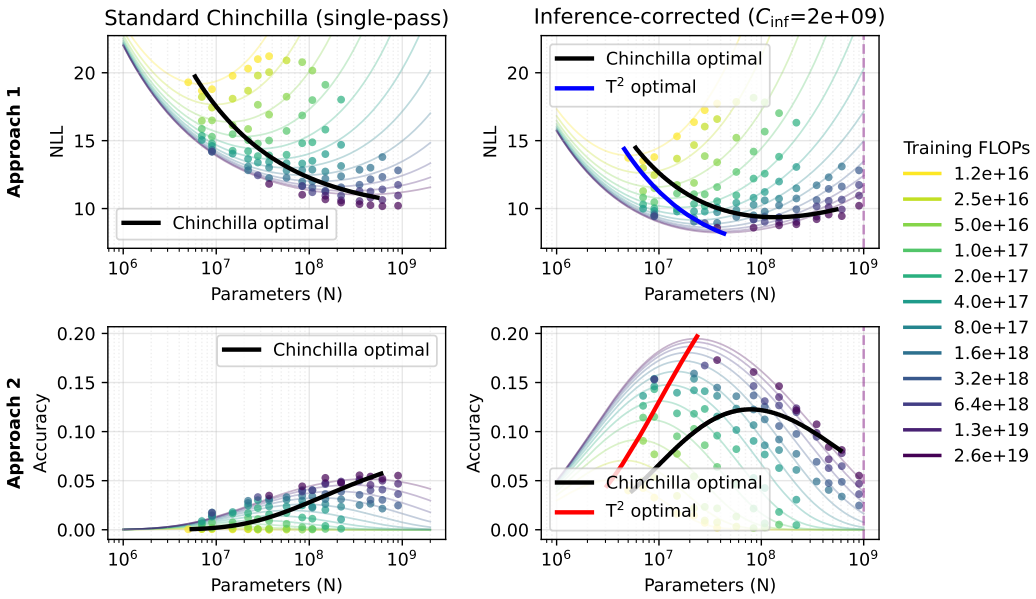

Figure 3: T^2 scaling across all of our evaluation tasks. Both approaches improve monotonically over Chinchilla scaling, while Chinchilla exhibits non-monotonic scaling in C_{train}.

the optimal frontier of both T^2 approaches shifts considerably toward smaller overtrained models with more repeated samples compared to the Chinchilla optimum. When inference-corrected, we see that the Chinchilla optimal frontier exhibits non-monotonic improvement in C_{train}. This is consistent with the findings of Snell et al. (2024), showing that smaller models with more test-time compute can outperform larger models. On the other hand, T^2 shows both stronger *and consistently monotonic* improvement, as we jointly model pretraining and test-time scaling. **These results confirm that if you know your test-time scaling budget, you should substantially overtrain compared to Chinchilla optimal pretraining.**

4.2 RQ2: Does T^2 Scaling Extrapolate to Overtrained Checkpoints?

Next, we evaluate **RQ2** by fitting both T^2 approaches to standard Chinchilla scaling checkpoints and measuring the performance of extrapolation to overtrained checkpoints.

Setup. We fit both of our T^2 approaches to a suite of 85 Chinchilla scaling checkpoints from Porian et al. (2024) (which stop short of the optimal overtraining regime that T^2 predicts) and measure the relative absolute error of extrapolating the predictions to 21 overtrained checkpoints that we train using an identical pretraining setup. We include training details and the exact checkpoint grid in Appendix C. We also compare the empirical best overtrained checkpoint (among our 21) in the inference-corrected regime and compare it to the empirical Chinchilla optimal checkpoint at a pretraining budget of $C_{\text{train}} = 2.56 \times 10^{19}$ across all eight tasks. We set $C_{\text{inf}} = 2 \times 10^9$ for all of the above.

Results. Our extrapolation results are shown in Figure 4 and empirical checkpoint pass@k results are shown in Table 1. Figure 4 shows that our T^2 approaches both extrapolate to the 21 new overtrained checkpoints. While both approaches somewhat overestimate performance, Approach 1 extrapolates better than Approach 2, with a relative error of 2.8% compared to 8.4%. Table 1 shows that our best small overtrained checkpoints always outperform the Chinchilla optimal checkpoints when inference corrected, across all eight tasks. **This confirms that T^2 extrapolates to real overtrained checkpoints, and that this phenomenon is not just an artifact of our T^2 approaches.**

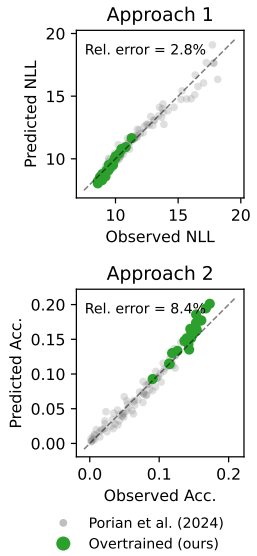

Figure 4: Extrapolating Porian et al. (2024) checkpoints to the overtraining regime.

		Best overtrained	Chinchilla opt.
Real	LAMBADA OpenAI	**49.90% (37M)**	27.30% (455M)
	OpenBookQA	**1.40% (37M)**	0.30% (901M)
	SciQ	**1.20% (37M)**	0.22% (611M)
	ARC-Easy	**0.14% (149M)**	0.07% (611M)
Synthetic	Simple Knowledge	**14.60% (84M)**	5.80% (901M)
	Simple Reasoning	**57.90% (37M)**	18.40% (901M)
	Commonsense Causal	**8.10% (37M)**	1.40% (901M)
	Spatial Reasoning	**6.00% (37M)**	1.10% (901M)

Table 1: Comparison of overtrained base models vs Chinchilla optimal pass@k, subject to $C_{\text{train}} = 2.56 \times 10^{19}$ and $C_{\text{inf}} = 2 \times 10^{9}$ FLOPs. Optimal model sizes are shown in parentheses.

		OpenBookQA	SciQ	ARC-Easy
FT	Best overtrained	**2.80% (37M)**	**56.10% (149M)**	**5.60% (149M)**
	Chinchilla opt.	0.45% (901M)	29.00% (901M)	1.50% (901M)
SFT	Best overtrained	**2.60% (37M)**	**66.80% (84M)**	**8.20% (37M)**
	Chinchilla opt.	0.38% (901M)	57.60% (347M)	3.40% (455M)

Table 2: Post-training comparison of overtraining vs Chinchilla optimal pass@k, subject to $C_{\text{train}} = 2.56 \times 10^{19}$ and $C_{\text{inf}} = 2 \times 10^{9}$ FLOPs. Optimal model sizes are shown in parentheses.

4.3 RQ3: Does T^2 Scaling Survive Post-Training?

Finally, we evaluate **RQ3** by showing that our findings persist after post-training.

Setup. We explore two canonical post-training techniques: standard fine-tuning (FT) and supervised fine-tuning (SFT), where we only fine-tune on the targets. We post-train on the three real tasks that have a standard training set: ARC-Easy, SciQ, and OpenBookQA, and report improved performance on the test sets for each of these. Additional post-training details can be found in Appendix D. We allocate the same number of training steps to each checkpoint, rather than scaling training based on FLOPs, since we ultimately train to convergence. After post-training, we fit both T^2 approaches to the FT and SFT checkpoints and evaluate their optimal tokens per parameter frontier compared to base models under T^2 scaling and the Chinchilla frontier. Finally, like in **RQ2**, we compare the best overtrained FT and SFT checkpoints to the Chinchilla optimal checkpoints for each task.

Results. Our results are shown in Figure 5 and Table 2. We see in Figure 5 that the optimal frontier continues to shift toward smaller overtrained models with more test-time samples across all three tasks and methods. Again, we find that these results are consistent between Approach 1 and Approach 2. On the other hand, we find that the optimal overtraining recommendation is somewhat subdued compared to T^2 on the base models alone, but not enough to shift it back to the original Chinchilla recommendation. The finding that it is subdued is consistent with prior work showing that overtrained models are harder to fine-tune (Springer et al., 2025). Finally, we see in Table 2 that our best overtrained checkpoints still outperform the Chinchilla optimal checkpoints after post-training, and that performance improves across the board compared to the same analysis on base models in Table 1. While we do not study subsequent RL stages (RLHF/RLVR), SFT is the post-training stage most likely to threaten our conclusion—the erosion mechanism of Springer et al. (2025) acts directly through supervised weight updates—and it subdues our prescription without reversing it. **This confirms that our findings with T^2 scaling persist after post-training.**

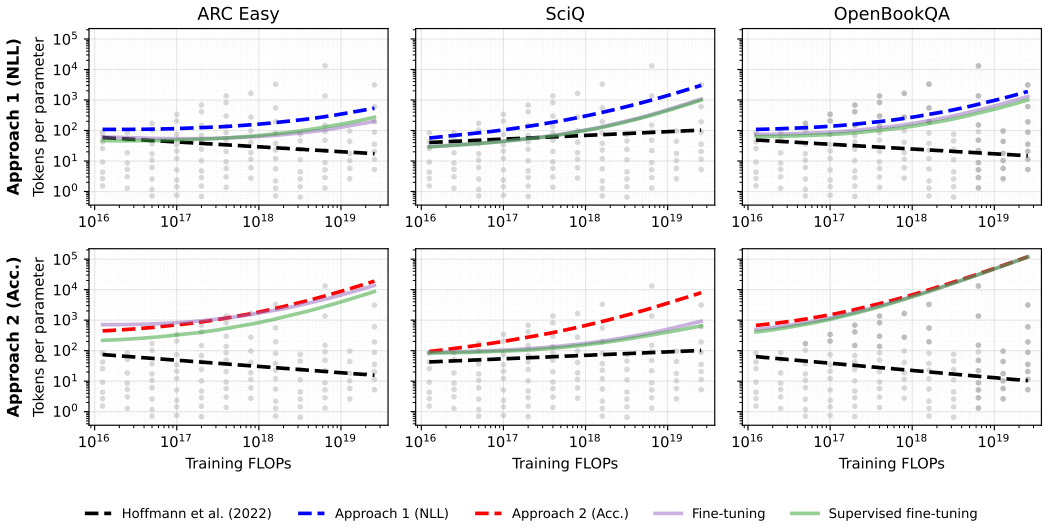

Figure 5: T^2 overtraining findings survive post-training. The optimal frontier is slightly subdued compared to base models but not reversed—consistent with Springer et al. (2025): overtrained models are harder to fine-tune, and this acts directly through the supervised weight updates of FT and SFT, making them the stages most likely to weaken our findings.

5 Conclusion

In this work, we have presented T^2 scaling laws that jointly optimize model size, training tokens, and the number of repeated samples at test-time under fixed pretraining and inference budgets. We find that when test-time compute via repeated sampling is accounted for during pretraining decisions, the optimal model is substantially smaller and more overtrained than what standard Chinchilla scaling prescribes. This finding is consistent across two complementary modeling approaches: Approach 1 which models the NLL, and Approach 2 which models the pass@k accuracy directly. We validated this across eight real and synthetic downstream tasks, validated that T^2 scaling extrapolates to the overtraining regime where its optima are predicted, and that our findings persist after post-training. **Based on our findings, we offer a recommendation to practitioners: if you know your test-time scaling budget with repeated sampling, you should train a smaller model for longer, and T^2 scaling offers the blueprint for doing so.**

Notably, this regime is already emerging in industrial-scale releases. For example, LFM2.5-350M (Liquid AI, 2026) was trained on 28T tokens—roughly 80,000 tokens per parameter, or $\sim$4,000$\times$ the ratio Chinchilla prescribes—despite being a non-Transformer hybrid architecture, supporting our view that this prescription reflects the compute-allocation structure of train-and-deploy scaling rather than an architecture-specific effect. Incidentally, LFM2.5-350M was released within a day of the initial submission of our manuscript. In future work, we plan to validate our prescribed overtraining recipes at larger scales, account for richer Transformer-specific inference cost models (e.g., prefill, memory, and KV-caching), and explicitly model the role of post-training in T^2 scaling, including RL-based post-training.

Acknowledgements

We are grateful for the support of the National Science Foundation (NSF) (CCF2106707), the Defense Advanced Research Projects Agency (DARPA Young Faculty Award and DARPA SAFRON), and the Wisconsin Alumni Research Foundation (WARF).

References

Akshita Bhagia, Jiacheng Liu, Alexander Wettig, David Heineman, Oyvind Tafjord, Ananya Harsh Jha, Luca Soldaini, Noah A Smith, Dirk Groeneveld, Pang Wei Koh, et al. Establishing task scaling laws via compute-efficient model ladders. *arXiv preprint arXiv:2412.04403*, 2024.

Song Bian, Minghao Yan, and Shivaram Venkataraman. Scaling inference-efficient language models. In *Forty-second International Conference on Machine Learning*, 2025.

Bradley Brown, Jordan Juravsky, Ryan Saul Ehrlich, Ronald Clark, Quoc V Le, Christopher Re, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling, 2025. URL https://openreview.net/forum?id=0xUEBQV54B.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

Ryan Ehrlich, Bradley Brown, Jordan Juravsky, Ronald Clark, Christopher Ré, and Azalia Mirhoseini. Codemonkeys: Scaling test-time compute for software engineering. *arXiv preprint arXiv:2501.14723*, 2025.

Samir Yitzhak Gadre, Georgios Smyrnis, Vaishaal Shankar, Suchin Gururangan, Mitchell Wortsman, Rulin Shao, Jean Mercat, Alex Fang, Jeffrey Li, Sedrick Keh, Rui Xin, Marianna Nezhurina, Igor Vasiljevic, Jenia Jitsev, Luca Soldaini, Alexandros G. Dimakis, Gabriel Ilharco, Pang Wei Koh, Shuran Song, Thomas Kollar, Yair Carmon, Achal Dave, Reinhard Heckel, Niklas Muennighoff, and Ludwig Schmidt. Language models scale reliably with over-training and on downstream tasks. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=SKfNXHJcCq.

Sachin Goyal, Pratyush Maini, Zachary C Lipton, Aditi Raghunathan, and J Zico Kolter. Scaling laws for data filtering–data curation cannot be compute agnostic. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22702–22711, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Dirk Groeneveld, Iz Beltagy, Evan Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, et al. Olmo: Accelerating the science of language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15789–15809, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, DDL Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 10, 2022.

Berivan Isik, Natalia Ponomareva, Hussein Hazimeh, Dimitris Paparas, Sergei Vassilvitskii, and Sanmi Koyejo. Scaling laws for downstream task performance of large language models. In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2024.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Matt Gardner Johannes Welbl, Nelson F. Liu. Crowdsourcing multiple choice science questions. 2017.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Joshua Kazdan, Rylan Schaeffer, Youssef Allouah, Colin Sullivan, Kyssen Yu, Noam Levi, and Sanmi Koyejo. Efficient prediction of pass@ k scaling in large language models. *arXiv preprint arXiv:2510.05197*, 2025.

Liquid AI. Lfm2.5-350m: No size left behind. *Liquid AI Blog*, 2026. www.liquid.ai/blog/lfm2-5-350m-no-size-left-behind.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*, 2018.

Gabriel Orlanski, Nicholas Roberts, Aws Albarghouthi, and Frederic Sala. Reward models enable scalable code verification by trading accuracy for throughput. *arXiv preprint arXiv:2506.10056*, 2025.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset, Aug 2016.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Hamza Alobeidli, Alessandro Cappelli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The refinedweb dataset for falcon LLM: Outperforming curated corpora with web data only. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=kM5eGcdCzq.

Tomer Porian, Mitchell Wortsman, Jenia Jitsev, Ludwig Schmidt, and Yair Carmon. Resolving discrepancies in compute-optimal scaling of language models. *Advances in Neural Information Processing Systems*, 37:100535–100570, 2024.

A Yang Qwen, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengpeng Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint*, 2024.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Nicholas Roberts, Niladri S Chatterji, Sharan Narang, Mike Lewis, and Dieuwke Hupkes. Compute optimal scaling of skills: Knowledge vs reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 13295–13316, 2025.

Jon Saad-Falcon, E Kelly Buchanan, Mayee F Chen, Tzu-Heng Huang, Brendan McLaughlin, Tanvir Bhathal, Shang Zhu, Ben Athiwaratkun, Frederic Sala, Scott Linderman, et al. Shrinking the generation-verification gap with weak verifiers. *arXiv preprint arXiv:2506.18203*, 2025.

Nikhil Sardana, Sasha Doubov, and Jonathan Frankle. Beyond chinchilla-optimal: Accounting for inference in language model scaling laws. In *International Conference on Machine Learning*, 2023. URL https://api.semanticscholar.org/CorpusID:266693796.

Rylan Schaeffer, Joshua Kazdan, John Hughes, Jordan Juravsky, Sara Price, Aengus Lynch, Erik Jones, Robert Kirk, Azalia Mirhoseini, and Sanmi Koyejo. How do large language monkeys get their power (laws)? In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=QqVZ28qems.

Rylan Schaeffer, Noam Itzhak Levi, Brando Miranda, and Sanmi Koyejo. Pretraining scaling laws for generative evaluations of language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=Ym33xJYINV.

Mustafa Shukor, Enrico Fini, Victor Guilherme Turrisi da Costa, Matthieu Cord, Joshua Susskind, and Alaaeldin El-Nouby. Scaling laws for native multimodal models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12–23, 2025.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Jacob Mitchell Springer, Sachin Goyal, Kaiyue Wen, Tanishq Kumar, Xiang Yue, Sadhika Malladi, Graham Neubig, and Aditi Raghunathan. Overtrained language models are harder to fine-tune. *arXiv preprint arXiv:2503.19206*, 2025.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024a.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024b.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Wenyue Hua, Haolun Wu, Zhihan Guo, Yufei Wang, Niklas Muennighoff, et al. A survey on test-time scaling in large language models: What, how, where, and how well? *arXiv preprint arXiv:2503.24235*, 2025.

Appendix Roadmap

Our appendix is structured as follows. We begin with related work in Appendix A, followed by Appendix B, which presents per-task scaling law analyses. We next turn to experimental details: Appendix C and Appendix D describe our pretraining and post-training setups, respectively, while Appendix E provides descriptions of all evaluation tasks employed in our study. Finally, Appendix F presents the details of our T^2 scaling fitting methodology.

A Related Work

Our work sits at the intersection of three research threads: (i) pretraining scaling laws, (ii) test-time scaling, and (iii) overtrained models.

A.1 Pretraining Scaling Laws

Kaplan et al. (2020) established that model loss follows predictable power laws as a function of model size and training data. Hoffmann et al. (2022) (Chinchilla) refined this into *compute-optimal training recipes*, prescribing how model size and token count should scale together under a fixed compute budget. Recent extensions have broadened the scope of scaling law modeling: studying data quality and quantity (Goyal et al., 2024), incorporating downstream task accuracy (Isik et al., 2024; Bhagia et al., 2024), decomposing scaling behaviors across knowledge and reasoning skills (Roberts et al., 2025), and extending to multimodal settings (Shukor et al., 2025). These frameworks, however, treat inference as an afterthought—optimizing for a model that is trained once and queried once. Sardana et al. (2023) take a step toward *deployment-aware* scaling by folding inference serving volume into the compute-optimal recipe, yet their analysis is limited to single-pass queries. Bian et al. (2025) also connect Chinchilla-style scaling to inference costs, but from the architecture perspective—searching for inference-efficient model shapes—without modeling repeated sampling. This direction is complementary to ours, which studies the interaction between pretraining and repeated sampling within a single architecture family; combining these orthogonal scaling axes is a challenging and fascinating direction for future work. We modernize this line of work, where the optimal training decisions must account for both the cost and the compounding performance gains of drawing multiple inference samples.

A.2 Test-Time Scaling

Beyond scaling pretraining compute, recent work has increasingly focused on investing computation at *inference time* (Snell et al., 2024; Zhang et al., 2025; Jaech et al., 2024; Orlanski et al., 2025). This test-time paradigm often focuses on the search for a correct reasoning path rather than the model's inherent knowledge and can broadly be categorized into three regimes: (i) *parallel scaling*, which uses consensus through self-consistency (Brown et al., 2025), or verification over multiple independent responses (Saad-Falcon et al., 2025); (ii) *sequential scaling*, which refines reasoning through iterative improvements or hierarchical pruning (Wei et al., 2022; Madaan et al., 2023); and (iii) *internal scaling*, which allows the model to dynamically adjust generation depth based on task difficulty (Jaech et al., 2024). In this work, we focus on *parallel repeated sampling*—the most common form of test-time scaling—and incorporate pretraining compute to jointly optimize allocation decisions.

A.3 Overtraining

Hoffmann et al. (2022) (Chinchilla) prescribes a compute-optimal ratio of roughly 20 training tokens per model parameter, yet modern models releases routinely deviate from this blueprint by *training smaller models on far more tokens than recommended*. This deliberate overtraining is motivated by *inference efficiency*: a smaller model costs less per query at deployment. Recent model families illustrate this trend—Llama-2-7B (Touvron et al., 2023) was trained on 2T tokens ($\sim$290$\times$ the recommended ratio); Google's Gemma-7B (Team et al., 2024a) was trained on 6T tokens ($\sim$857$\times$), and its successor Gemma 2-9B (Team et al., 2024b)

on 8T tokens (∼889×)—with OLMo (Groeneveld et al., 2024) following a similar philosophy. Most recently, LFM2.5-350M (Liquid AI, 2026)—an on-device model designed for verifiable settings such as tool use—pushed this trend to 28T tokens (∼80,000 tokens per parameter), landing deep within the overtrained regime that T^2 forecasts. Gadre et al. (2025) study the overtrained regime directly, showing that scaling laws remain reliable under overtraining and can forecast downstream task performance. However, they treat overtraining as a forecast target rather than an optimization variable, and do not consider test-time scaling; in contrast, we treat overtraining and test-time scaling as optimization variables, which reveals a compute-optimality-based justification for overtraining. Our work complements these findings by examining overtraining through a different lens: rather than studying its effect on post-training (Springer et al., 2025), we show that overtraining is actively *beneficial* when models are deployed with a repeated-sampling inference budget, and we provide a principled framework for determining how much to overtrain given a joint train-and-test compute allocation.

B Per-Task Analysis

We present isoFLOP profiles for each of the individual tasks in our evaluation suite in Figure 6 for Approach 1 and Figure 7 for Approach 2 . We find that overtraining predictions are relatively stable across inference budgets for both approaches.

C Pretraining Details

In this section, we provide details of our pretraining setup and scaling grid.

C.1 Checkpoint Scaling Grid

Figure 8 shows our checkpoint grid, comprising pretrained checkpoints from Porian et al. (2024) alongside additional overtrained checkpoints we pretrained in this work. Model sizes range from 5M to 901M parameters, and training FLOPs span 1.25×10^{16} to 2.56×10^{19}. Each cell reports the number of tokens per parameter, which characterizes the degree of overtraining. Typically, a suite of Chinchilla scaling checkpoints contains checkpoints at either side of the typical 20 tokens per parameter recommendation derived from Hoffmann et al. (2022). However, since T^2 suggests overtraining beyond the available set of checkpoints, we train additional checkpoints at higher tokens per parameter ratios. The overtrained checkpoints (shown in orange) are used to validate our forecasts in §4.2.

C.2 Hyperparameters

We train our overtrained checkpoints, shown in Figure 8, from scratch using the OpenLM framework with the same fixed hyperparameters used for the Chinchilla-optimal checkpoints from Porian et al. (2024). Specifically, we use their `hparams=base`, `warmup=short`, `decay=chinchilla` configuration. We use the AdamW optimizer with a learning rate of 3×10^{-3}, $\beta_1 = 0.9$, $\beta_2 = 0.95$, and a decoupled weight decay of 1×10^{-4}. Training uses a global batch size of 256 sequences of length 2048 tokens, cosine learning rate decay to zero matched to the token budget of each run, and a warmup period equal in tokens to the model's parameter count. We apply gradient clipping with a max norm of 1.0, QK-normalization, z-loss with coefficient 10^{-4}, and train in bfloat16 mixed precision. All hyperparameters are held fixed across model sizes, consistent with the base (untuned) configuration of Porian et al. (2024). We train on the RefinedWeb dataset with a vocabulary size of 50,432.

D Post-Training Details

We describe our post-training setup and configurations below. We employ two variants of post-training: **(i) standard fine-tuning** and **(ii) supervised fine-tuning (SFT)**. Standard

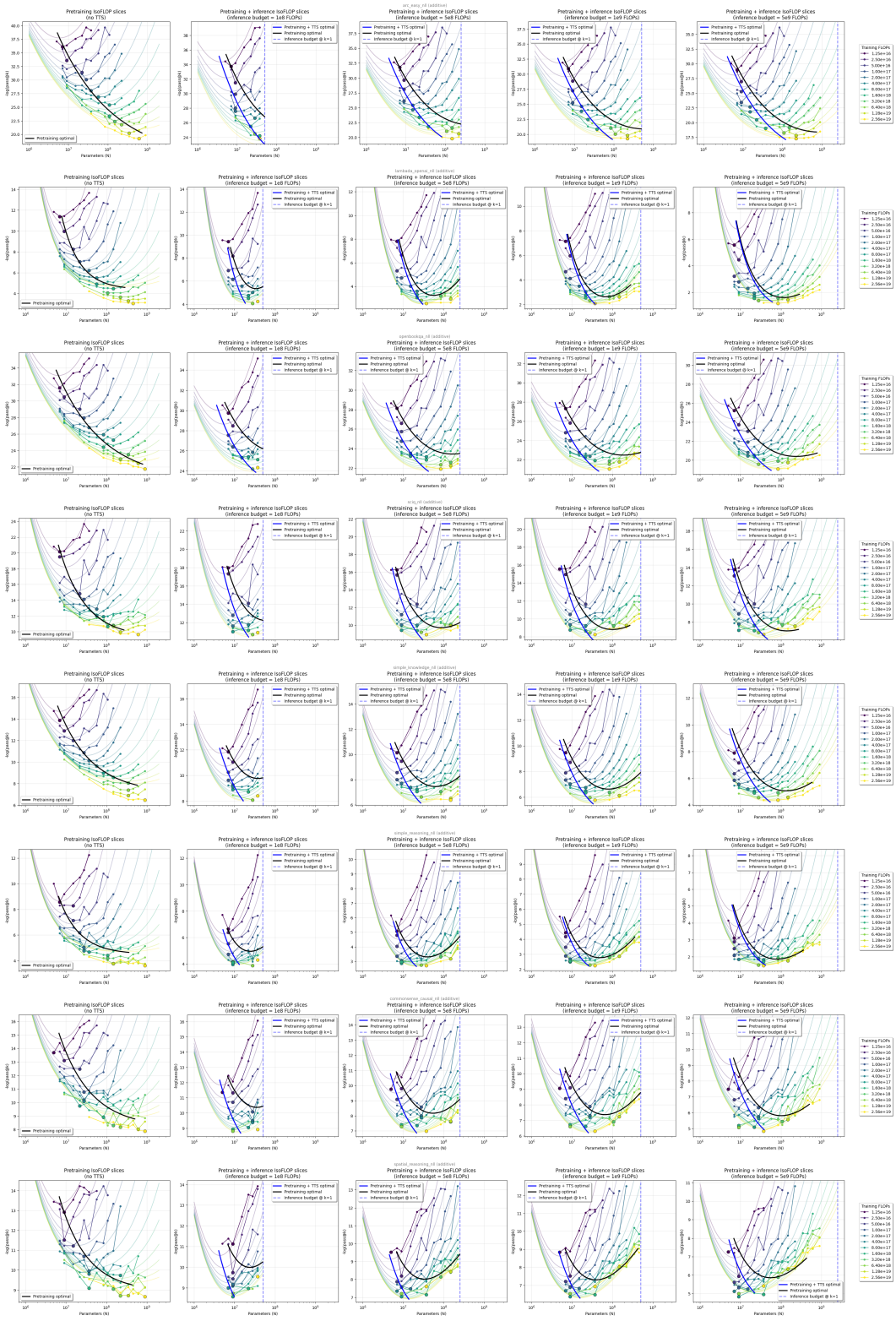

Figure 6: Approach 1 IsoFLOP profiles across different scaling budgets for all eight tasks.

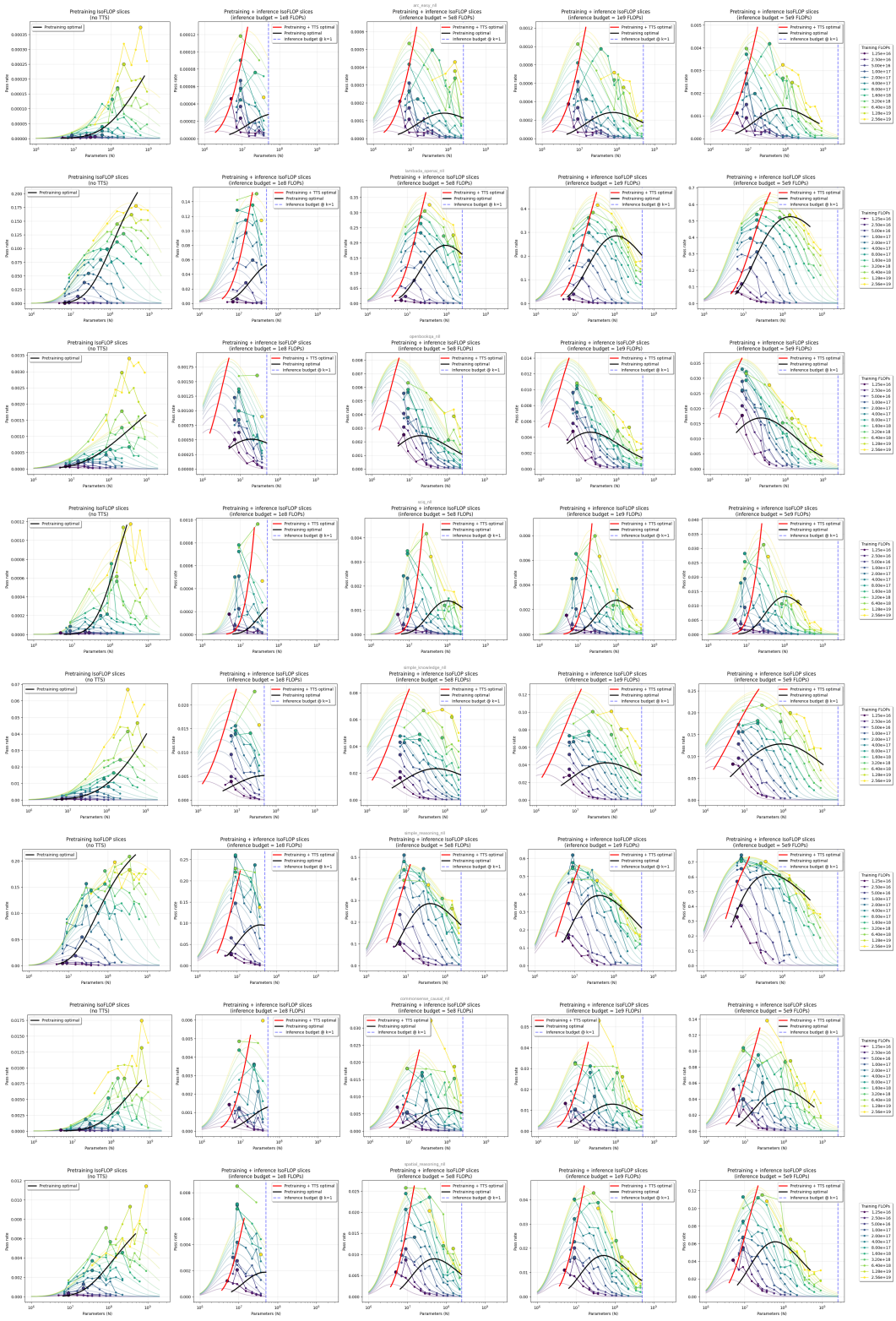

Figure 7: Approach 2 IsoFLOP profiles across different scaling budgets for all eight tasks.

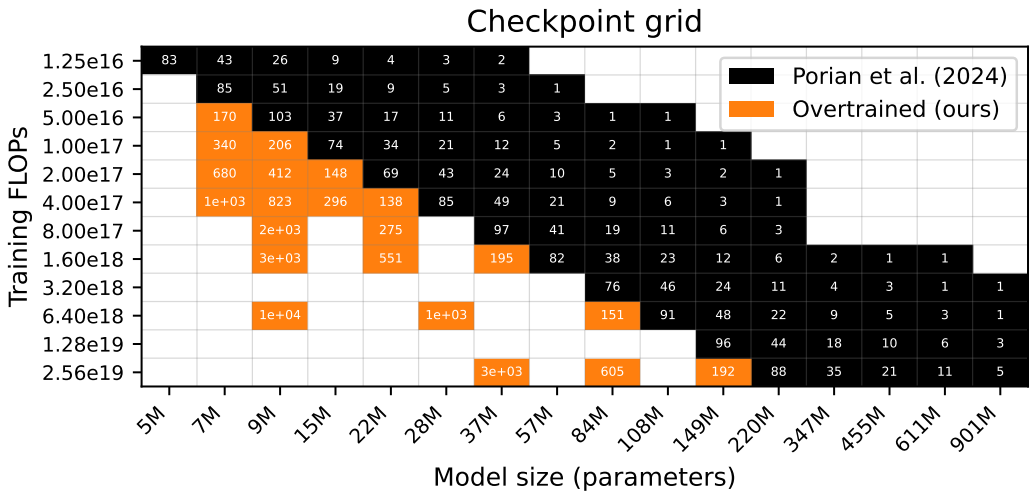

Figure 8: Overall checkpoint scaling grid. Each cell reports the number of tokens per parameter. Orange cells are overtrained checkpoints we created.

Box 1: Training Data Formats

Each format separates the prompt (plain) from the completion (highlighted), which is the only portion used in the SFT loss.

ARC Easy:

`Question: {question}\nAnswer: {answer}`

OpenBookQA:

`{question} {answer}`

SciQ:

`{support}\nQuestion: {question}\nAnswer: {answer}`

fine-tuning follows the conventional next-token prediction objective, computing loss over both the instruction (question) and completion (answer). SFT, in contrast, computes loss over the completion only, excluding instruction tokens from parameter updates.

We fine-tune on three tasks—ARC Easy (Clark et al., 2018), SciQ (Johannes Welbl, 2017), and OpenBookQA (Mihaylov et al., 2018)—covering the full population of pretrained checkpoints, including the overtrained ones. Each model is trained for 6 epochs until convergence using a batch size of 8 and a constant learning rate of 2×10^{-5}, after that we evaluate on the respective test set. All fine-tuning experiments are conducted on 4 NVIDIA A10 GPUs. Box D presents the training data format for each task, where the highlighted tokens indicate the completion portion used in the SFT loss computation. Their evaluation follows the same format: we measure negative log-likelihood over the correct answer placed in the highlighted placeholder.

Box 2: Commonsense Causal Reasoning

Example 1:

```
Grandparents tell stories to grandchildren. Teachers explain
concepts to students. Coaches demonstrate techniques to players
```

Example 2:

```
A mother comforts a crying baby. A teacher encourages a
struggling student. A coach motivates a discouraged player
```

Box 3: Simple Knowledge Recall

Example 1:

```
The capital of Egypt is Cairo
```

Example 2:

```
The fifth taste is umami
```

E Evaluation Tasks

Next, we describe the eight downstream tasks used to evaluate T^2 scaling, covering both real-world benchmarks and synthetic tasks. For all tasks, we measure the NLL of each model over the correct answer. Accuracy is always measured as exact match on generated sequences under pass@k, where the model must generate the highlighted answer span—not as multiple-choice accuracy (see §4 for a justification of this metric).

We evaluate on four real-world benchmarks.

1. **LAMBADA** (Paperno et al., 2016) (OpenAI variant): tests long-range language understanding, where the model must predict the final word of a passage given a broad context.
2. **ARC Easy** (Clark et al., 2018): consists of elementary-level science questions in a four-way multiple choice format, drawn from standardized tests.
3. **SciQ** (Johannes Welbl, 2017): contains science exam questions paired with supporting passages, presented in a multiple-choice format.
4. **OpenBookQA** (Mihaylov et al., 2018): requires multi-step reasoning by combining an open book of core science facts with broader common knowledge, presented as four-way multiple choice questions.

In addition to these four benchmarks, we incorporate four synthetic tasks spanning different domains. These tasks are designed to evaluate models on (i) simple knowledge recall, (ii) multi-step arithmetic reasoning, (iii) commonsense causal reasoning, and (iv) spatial reasoning. Each task consists of 1,000 fill-in-the-blank or short-completion questions, generated using GPT-5 and Claude Opus 4.6. Below, we present representative examples from each task along with their evaluation format. As in Box D, the token spans used to compute the NLL are highlighted in each example below.

F Fitting T^2 Scaling

In this section, we describe how each of our T^2 approaches are fit to empirical checkpoints.

Box 4: Multi-Step Arithmetic Reasoning

Example 1:

```
I have 5 toys. I give away 2 toys. Step 1: I started with 5
toys. Step 2: I gave away 2 toys. Step 3: 5 minus 2 equals 3
```

Example 2:

```
Pattern: 10, 20, 30, ... This adds 10 each time. After 30
comes 40
```

Box 5: Spatial Reasoning

Example 1:

```
The baby is in the crib. The crib is in the nursery. The
nursery is in the house. So the baby is in the house
```

Example 2:

```
The glasses are in the case. The case is in the handbag.
So the glasses are in the handbag
```

Fitting Approach 1. We fit the seven parameters $(\log A, \log B, \log E, \alpha, \beta, \log G, \gamma)$ of the additive model by minimizing the sum of squared errors (SSE) between predicted and empirical NLL values across all checkpoints and sampled values of k. We use the L-BFGS-B algorithm with 500 random restarts (each with up to 5,000 iterations and a tolerance of 10^{-15}) and we select the run with the lowest objective value.

Fitting Approach 2. We fit the model in two stages. First, we fit the standard Chinchilla scaling model $\widehat{L}(N, D) = E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}$ to the empirical NLL values of all checkpoints. We profile over a grid of 40 candidate E values spaced between $0.01 \cdot \min(\text{NLL})$ and $0.95 \cdot \min(\text{NLL})$; for each, we optimize the remaining four parameters $(\log A, \log B, \alpha, \beta)$ via L-BFGS-B with 50+ random restarts, using inverse-variance weighting across isoFLOP groups. Second, we fit the Beta regression parameters. The per-question success probability is modeled as $p \sim \text{Beta}(a_{N,D}, b_{N,D})$ where $\mu = a_{N,D}/(a_{N,D} + b_{N,D})$ is a scaled logit link and the concentration $\nu = a_{N,D} + b_{N,D}$ is parameterized as a log link function. Together, the five parameters $(\theta_0, \theta_1, \theta_2, \theta_3, \theta_4)$ are fit by minimizing SSE between predicted and empirical pass@k accuracy values over a grid of initializations seeded from a sigmoid baseline, again using L-BFGS-B.