What is Train-to-Test Scaling? How to optimize AI costs from training to inference.

Learn about Train-to-Test scaling – a method that optimizes the entire cost of AI, from training to inference, helping smaller models achieve higher performance.

For many years, the principles of building large language models (LLMs) focused primarily on optimizing training costs. However, in real-world applications, inference costs have become the long-term dominant factor—especially as modern systems increasingly rely on 'inference sampling' to improve accuracy.

This gap has spurred the emergence of a new approach called **Train-to-Test (T2) scaling** — a framework that optimizes the entire computing budget from the training to the deployment phase.

When traditional scaling rules are no longer valid.

In the world of LLM, scaling rules serve as a guiding principle. On the one hand, they guide how resources are allocated during model training; on the other hand, they shape how resources are used during deployment, such as allowing the model to 'think longer' or generate multiple answers to select the best result.

The problem is that these two sets of rules were developed almost independently, even though they are actually closely related.

For example, the famous **Chinchilla** rule suggests an optimal ratio between the number of parameters and training data (around 20 tokens per parameter). However, in practice, many modern models like Llama or Gemma intentionally 'overtrain' small models with huge amounts of data—going against this rule.

The reason is simple: when the cost per inference becomes expensive (due to the large model size), iterating many times to increase accuracy becomes impractical. Conversely, small but well-trained models can allow for many iterations of inference at a much lower cost.

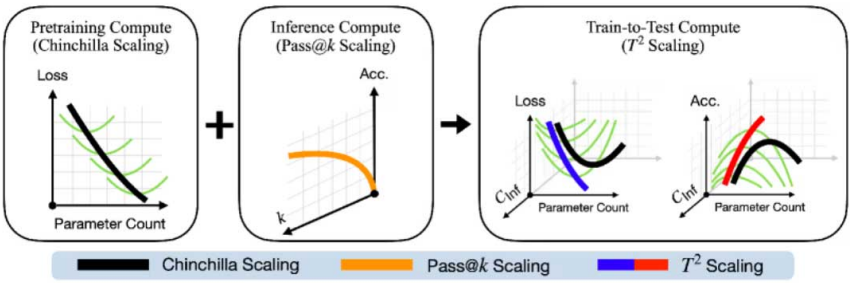
Train-to-Test Scaling: Connecting Training and Reasoning

To address this discrepancy, researchers have proposed **Train-to-Test (T2) scaling laws** , a model that combines all three key factors into a single equation:

1. Model size
2. Training data volume
3. Number of inference sampling cycles during deployment

Instead of optimizing individual components, T2 views the entire process as a unified system. It not only calculates training costs but also the iterative costs of inference—a factor often overlooked previously.

Interestingly, this framework can predict model performance based on the probability of success after multiple attempts (pass@k), rather than solely on the training error rate. This more accurately reflects how AI is used in real-world applications.



Smaller but more powerful

When applying T2 scaling to experiments with over 100 different models, the results revealed something quite surprising:

In many cases, the optimal strategy is not to build a larger model, but rather to use a smaller-scale model, train it with more data, and leverage the cost savings to increase the number of inferences.

This approach allows small but well-trained models to outperform large models in many problems—especially those requiring complex reasoning.

In other words, performance no longer depends solely on model size, but rather on how intelligently you allocate your computing budget.

However, not every application benefits from T2 scaling. This framework is particularly effective for tasks requiring multi-step reasoning, such as programming, solving mathematical problems, or complex workflow agents.

Conversely, for knowledge-based applications like conventional chatbots, the benefits may not be as apparent.

This establishes an important principle: **the choice of scaling strategy must be based on the nature of the problem, not the general trend of the industry** .

		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)

Practical implications for developers and businesses

One notable point is that implementing T2 scaling doesn't require overly complex technology. Techniques like **KV caching** (cache of processed context) can significantly reduce the cost of iterative inference.

However, overtraining also comes with trade-offs. Overtrained models can be more difficult to fine-tune and require very large amounts of data—even risking hitting the 'data limit' when high-quality data sources are no longer sufficient.

However, in the context of the increasing costs of frontier models, T2 scaling offers a new, 'democratizing' approach: allowing smaller organizations to build powerful AI systems without massive budgets.

Train-to-Test scaling is not just a technical improvement, but a new way of looking at how to build AI.

Instead of focusing intensely on the big picture, this approach emphasizes **intelligently allocating resources across the entire AI lifecycle** —from training to inference.

In the future, as AI systems become increasingly reliant on reasoning and agent workflows, strategies like T2 could become the new standard, replacing the 'bigger is better' mindset that has dominated the industry for years.