

4.2 — Reasoning and Inference-Time Compute

Gavin · AI-assisted course design

Contents

1	An answer has a computation budget	3
2	Training can teach a model how to use extra time	3
3	Intermediate tokens give later predictions more context	4
4	There are several ways to spend inference compute	4
4.1	Continue one trajectory	4
4.2	Sample several complete candidates	5
4.3	Evaluate completed answers or intermediate steps	5
4.4	Search over partial solutions	5
5	A correct candidate is not a correctly selected answer	6
5.1	Add an imperfect verifier	7
5.2	Count the cost of the selection rule	7
6	Shared failures change the curve	8
7	Adaptive budgets ask where the next unit helps	8
8	Effort controls are product and model specific	9
8.1	Same checkpoint or a different system	10
9	Compare performance at a stated budget	10
10	Correctness and faithfulness answer different questions	11
11	Check your understanding	11
12	More Learning	12

1 An answer has a computation budget

Suppose a system produces an incorrect solution in two seconds. Giving it twenty seconds might help if it can explore another approach, check a calculation, or repair a mistake. It might also repeat the mistake in more words. Time is a resource; the procedure using that resource determines what it buys.

Lesson 3.2 distinguished changing learned weights from changing the computation performed with those weights. Lesson 4.1 introduced generation and its runtime costs. We now connect those ideas: **a fixed model can participate in several inference procedures with different costs and different success rates**. A larger budget is neither a weight update nor a correctness guarantee.

By the end, you should be able to:

- distinguish reasoning training, intermediate tokens, and neural computation;
- explain sampling, verification, search, and adaptive computation;
- calculate candidate availability and selection success under explicit assumptions;
- compare performance while accounting for latency, tokens, and failed attempts;
- interpret reasoning-effort controls without inventing their implementation;
- separate a correct answer, a good explanation, and a faithful causal account.

i Math to know / refresh

Needed now: probability, independence, complements, conditional probability, and the intuition that additional sequential steps take work.

Useful refresh: finite geometric sums, expected values, and diminishing marginal returns. We derive the formulas used below.

Side trail: sequential decision theory, search algorithms, calibrated uncertainty, and causal intervention. None is required for the core Lab.

2 Training can teach a model how to use extra time

Write a model as $\pi_{\theta}(y \mid x)$, with learned parameters θ . Reasoning-oriented training changes θ : demonstrations can teach useful solution procedures; reinforcement learning can favor attempts that pass an outcome check. Whether a response is long, includes corrections, or sounds reflective does not identify its training recipe.

The DeepSeek-R1 report provides a concrete example. Its R1-Zero procedure applied reinforcement learning to a base checkpoint without preliminary supervised fine-tuning, whereas the R1 procedure included cold-start demonstrations and further stages. The authors reported changes in generated solution behavior during training. That is evidence about their documented training runs, not evidence that every long answer received the same treatment. DeepSeek-AI, Sections 2.2–2.3

At ordinary fixed-checkpoint inference, the parameters remain unchanged while tokens, activations, caches, and application state change. Training can make additional inference steps more useful, but these remain separate axes. Teaching someone a checking procedure and allowing time to perform it are an analogy for the distinction, not a claim that neural networks reason like people.

A trained verifier adds another set of parameters. Training that verifier is a training expense; running it against ten candidates is an inference expense. If an experiment fine-tunes the generator and changes its sampling budget simultaneously, the performance difference cannot be attributed solely to additional inference compute.

3 Intermediate tokens give later predictions more context

For a conventional causal language model, generated tokens become part of the context for later tokens. An intermediate calculation can therefore change the distribution of a later answer. The model performs neural computation to produce every token, including tokens that merely repeat earlier text.

Consider an original arithmetic prompt: “A box contains 17 packets of 23 cards. How many cards?” A generated decomposition, $17 \times 20 + 17 \times 3 = 340 + 51 = 391$, supplies intermediate values that later predictions can use. The statement can also be checked independently. Writing it down does not prove that every neural operation corresponds to a multiplication in that statement.

Chain-of-thought prompting asks for, or demonstrates, intermediate solution text. The original study by Wei and colleagues measured improvements on selected arithmetic, commonsense, and symbolic tasks for the models and prompting setups it tested. It did not establish that requesting longer text improves every task or model. Wei et al.

Keep three objects separate:

1. **Intermediate tokens:** discrete symbols generated before or between answer segments.
2. **Neural computation:** activation transformations that produce token distributions, including attention and MLP operations.
3. **An explanation:** text presented to a reader, possibly generated, edited, or summarized separately.

An undisplayed token sequence is still a token sequence. It is not the entirety of a model’s hidden activation state. Conversely, a model producing a short answer still performs substantial computation. Token visibility and computational existence are different properties.

Appending arbitrary filler also adds token-generation work, but it need not perform useful checking or decomposition. A compute budget enables a procedure; it does not specify one.

4 There are several ways to spend inference compute

4.1 Continue one trajectory

A system can generate a longer intermediate solution, revise an answer in context, or run a tool and continue from the result. These steps are sequential: later work depends on earlier output. An early mistaken premise may persist through the whole trajectory.

A revision prompt changes the context while leaving the checkpoint fixed. A calculator changes the available evidence. Neither should be described as a weight update. If the revised answer is correct, ask which intervention made the difference.

4.2 Sample several complete candidates

A system can sample N responses and select one. **Best-of- N** usually means choosing the highest-scoring candidate under a specified scoring procedure. “Best” refers to that procedure, which can be wrong.

Self-consistency instead aggregates final answers from sampled solution paths, often by choosing the most frequent normalized answer. Wang and colleagues found improvements on the arithmetic and commonsense benchmarks in their study. The method relies on how probability mass is distributed across correct and incorrect answers, not on a theorem that agreement implies truth. Wang et al.

Ten answers that reproduce one mistaken assumption offer weaker evidence than ten genuinely different successful checks. Identical strings also require no mysterious coordination: a distribution can simply place substantial probability on the same wrong answer.

4.3 Evaluate completed answers or intermediate steps

An **outcome verifier** evaluates a completed response. A **process verifier** evaluates intermediate steps. A verifier might be a learned model, executable tests, a symbolic checker, or a human rubric. Tests can be exact for the property they check while still omitting important requirements.

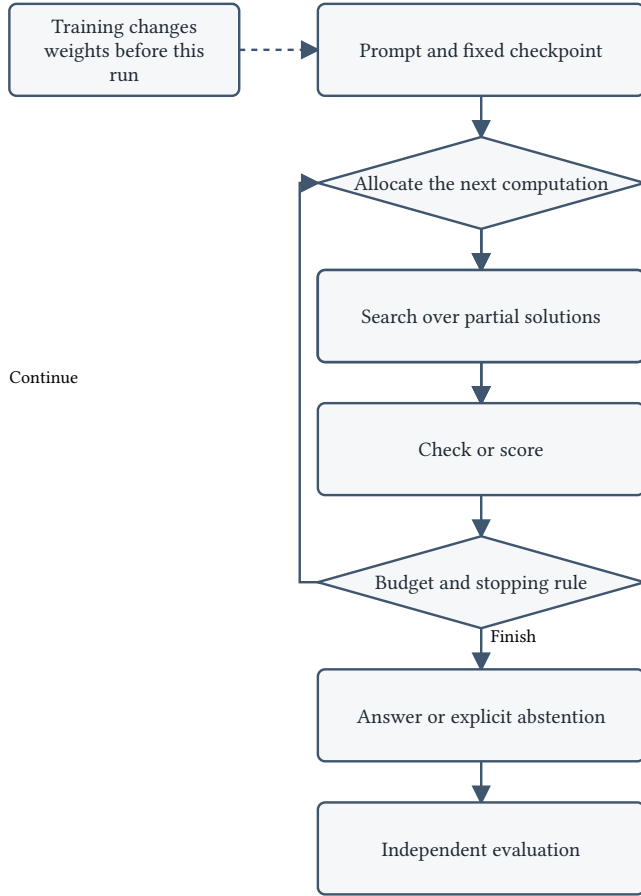
Lightman and colleagues compared outcome- and process-supervised reward models on mathematical solutions. Their evaluation used a fixed generator and selected among its sampled solutions; it was not reinforcement learning of that generator. Process supervision performed better in their reported MATH settings, with important differences between the large-scale and controlled smaller-scale comparisons. Lightman et al., Sections 2–4

A process score is still an estimate. Multiplying step scores, taking their minimum, or averaging them creates different selection rules. A proof checker, by contrast, can verify a formally specified statement without assessing whether that statement expresses the intended problem.

4.4 Search over partial solutions

A search procedure expands possible next steps, evaluates them, and chooses which branches to continue. It may backtrack. Branch width, search depth, stopping rules, and evaluator cost all affect the budget.

Tree of Thoughts demonstrated an explicit inference framework on Game of 24, creative writing, and mini-crosswords, using language-model proposals and evaluations within search. Its results concern those tasks and experimental configurations; they are not an implementation disclosure for a commercial reasoning slider. Yao et al.



Original course schematic. These are alternative or combinable procedures, not a claimed architecture for any particular product. A checking step and the independent evaluation have different jobs.

5 A correct candidate is not a correctly selected answer

Here is an original probability model, deliberately simpler than an LLM. Each candidate is correct with probability $p = 0.4$. Candidate correctness events are independent, with the same probability on every trial. These assumptions will matter.

The probability that all N candidates are wrong is $(1 - p)^N$. Therefore

$$P(\text{at least one correct}) = 1 - (1 - p)^N.$$

For four candidates, this is $1 - 0.6^4 = 0.8704$. That is **candidate availability**, not the accuracy of a system that must choose an answer. An oracle with access to correctness could exploit this availability. An actual selection procedure may miss the correct candidate.

The improvement from adding candidate $N + 1$ is

$$[1 - (1 - p)^{N+1}] - [1 - (1 - p)^N] = p(1 - p)^N.$$

At $p = 0.4$, the fifth candidate adds 0.05184 to availability, while the ninth adds about 0.00672. The benefit diminishes even in this favorable independent setting. This is an exact property of the invented probability model, not an empirical LLM scaling law.

5.1 Add an imperfect verifier

Let the verifier accept a correct candidate with probability $a = 0.9$ and accept a wrong candidate with probability $b = 0.1$. These are sensitivity and false-positive rate. Conditional on correctness, verifier decisions are independent across candidates. They do not see other candidates or adapt their thresholds.

Use this explicit policy: generate and check up to N candidates in a predetermined order; return the **first accepted** candidate; abstain if none is accepted. Equivalently, generate all N , assign binary scores, break accepted-candidate ties by order, and abstain when every score is zero. This is a restricted selection policy, not a model of every best-of- N system.

For one attempt:

$$P(\text{correct and accepted}) = pa = 0.36,$$

$$P(\text{wrong and accepted}) = (1 - p)b = 0.06.$$

Total acceptance probability is $q = 0.42$, and rejection probability is $r = 1 - q = 0.58$. A correct selected answer can occur at position one, or after one rejection, or after two, and so on:

$$P(\text{selected correct}) = pa \sum_{k=0}^{N-1} r^k = \frac{pa}{q}(1 - r^N).$$

Similarly,

$$P(\text{selected wrong}) = \frac{(1 - p)b}{q}(1 - r^N), \quad P(\text{abstain}) = r^N.$$

For $N = 4$, these are respectively 0.76014432, 0.12669072, and 0.11316496. They sum to one. The oracle availability was 0.8704, but actual selected-answer success is about 0.7601.

Among accepted answers, accuracy is $pa/q = 6/7 \approx 0.8571$, independent of N under these assumptions. Increasing N increases coverage, the fraction receiving an answer, while accepted-only accuracy stays fixed. Reporting only 85.71% would hide how many requests received nothing. Reporting only coverage would hide false acceptances.

5.2 Count the cost of the selection rule

Assign generation a cost of ten invented units and verification two units per candidate. These are dimensionless teaching units, not tokens, seconds, dollars, or measured FLOPs.

Generating and checking all four costs $4(10 + 2) = 48$ units per request. Sequentially stopping at the first acceptance uses an expected number of attempts

$$E[K] = \sum_{k=0}^{N-1} r^k = \frac{1 - r^N}{q}.$$

Why? Attempt two happens only if attempt one was rejected; attempt three requires two rejections. At $N = 4$, $E[K] = 2.111512$, so expected cost is 25.338144 units, with a worst-case cost of 48. Under this artificial model, both execution policies select the same answer from the same candidate stream.

Parallel execution could have shorter wall-clock latency even while spending more total work. The toy cost model does not quantify that trade-off. It also excludes caching, variable candidate lengths, batching, and expensive shared setup.

💡 Lab recommended here

Lab 13 — Test Compute Budgets derives these expectations, checks them with a seeded standard-library simulation, and breaks the independence assumption. The core requires no model, account, paid API, or GPU. An actual model-effort experiment is a separate optional extension.

6 Shared failures change the curve

Independently drawing random samples does not make a fixed success probability valid across a heterogeneous question set. A particular question may be much harder than another. Shared context can also anchor many attempts to the same error.

Change the toy population: half of requests are unsolvable by this generator, so every candidate is wrong. On the other half, candidates independently succeed with probability 0.8. Marginal single-candidate accuracy is still $0.5 \times 0.8 = 0.4$.

Nevertheless, four-candidate availability is now

$$0.5 [1 - (1 - 0.8)^4] = 0.4992,$$

far below 0.8704. It can never exceed one half. Candidates are independent conditional on the request type, but positively dependent after mixing request types. An average single-sample accuracy is insufficient to predict a multi-sample curve.

With an imperfect verifier, repeated attempts on the unsolvable half can eventually produce a false acceptance. As the sample cap rises, the accepted population includes more of those requests. Accepted-only accuracy can decline even while overall correct-answer probability rises. The Lab makes this selection effect visible.

Other failure modes arise when a learned scorer gives occasional wildly optimistic scores to wrong answers. Searching more candidates can locate those scoring mistakes. The lesson is to evaluate the selection rule against independent ground truth, rather than celebrate improvement in the score being optimized.

7 Adaptive budgets ask where the next unit helps

A fixed cap spends the same maximum budget on every request. An adaptive procedure might stop after a decisive check, allocate another attempt when evidence is weak, or choose a different search strategy for a difficult problem.

Snell and colleagues studied revision and verifier-guided search using PaLM 2 models fine-tuned for those roles on MATH. Their results depended on question difficulty and the method used. A difficulty-adapted allocation could improve efficiency relative to their best-of- N baseline; their largest-model comparisons were conditional on their FLOPs accounting and task regime. This supports studying allocation policies, not a universal exchange rate between parameters and reasoning time. Snell et al., Sections 4–7

Their predicted-difficulty experiment estimated each question’s difficulty using 2,048 samples and did not charge that estimation cost to the reported analysis. It was therefore not an end-to-end production-efficiency measurement. Snell et al., Section 3 and Appendix C

Difficulty is usually unknown before solving the request. A usable controller needs an observable signal, such as disagreement or a failed test. An experiment that consults the answer key to decide which questions deserve more work uses an oracle. Its allocation performance is an upper-bound-style comparison, not a deployable policy.

A sensible stopping decision compares expected benefit with incremental cost and deadline. “Try until correct” is not implementable unless correctness can actually be determined. “Try until the checker accepts” is implementable, but inherits the checker’s errors.

8 Effort controls are product and model specific

A label such as **low**, **medium**, or **high** usually expresses a relative operating choice within an interface. It does not universally specify a token count, search width, number of forward passes, or accuracy target.

Two documented examples, checked on **7 October 2026**:

- OpenAI’s **31 January 2025 o3-mini announcement** described low, medium, and high reasoning-effort options. This is a historical example, not a recommendation to choose that model now. Official announcement
- OpenAI’s model reference for the original **GPT-5** lists minimal, low, medium, and high. The extra label illustrates why an experiment must consult its exact model’s supported settings. Official model reference

OpenAI’s reasoning guide describes supported effort values and defaults as model-dependent. It also distinguishes unavailable raw reasoning tokens from optional reasoning summaries. Its token accounting includes non-visible reasoning in output usage and output limits; a limit can be reached before visible answer text appears. These are documented API behaviors, not evidence about another provider’s implementation. Official reasoning guide, checked 7 October 2026

A short displayed answer may therefore have consumed more generated tokens than a long-looking answer elsewhere. A summary is not a raw trace, and a raw token trace would still not expose all neural computation. Asking a product to reveal its hidden reasoning does not guarantee that the requested information is available or returned faithfully.

An effort setting also differs from an output-length preference, a sampling temperature, and a prompt asking for a detailed explanation. Each changes a different part of the setup. Do not substitute one for another without saying so.

8.1 Same checkpoint or a different system

In a controlled open-model experiment, one can hold weight files fixed and vary the inference algorithm. That directly supports a same-checkpoint comparison.

For a hosted API, a pinned model version and documented effort field support a narrower statement: the same exposed model version was requested with different settings. They do not independently prove the provider’s weight identity or reveal all serving decisions.

For a chat product, a mode switch might change a compute policy, route to another model, alter tools, or combine changes. If documentation does not resolve this, record the uncertainty. A more deliberate-looking answer alone cannot tell you which happened. Conversely, do not assert that routing occurred merely because it is possible.

9 Compare performance at a stated budget

Begin with the question you want to answer. “Does this system become more accurate when allowed more work?” permits different costs. “Which system is more efficient?” requires performance-versus-cost comparisons. “What does changing effort alone do?” requires other settings and the evaluation set to remain comparable.

Record at least:

- model/version, complete inputs, effort, decoding, tools, and selection policy;
- input tokens, provider-reported total output tokens, visible output, and separately reported reasoning tokens, without double-counting;
- generator, verifier, retry, and tool work;
- elapsed time, time to first visible answer, and timeout rules;
- correctness on all requests, abstention, incomplete output, and errors;
- uncertainty across requests and, where practical, repeated trials.

For example, OpenAI’s total output usage can include non-visible formatting as well as reasoning, so visible plus reasoning counts need not exhaust it. Preserve the provider’s usage fields. Official token-counting guide

Tokens are useful accounting units, but different tokenizers, architectures, context lengths, and hardware can make equal token counts unequal compute. Likewise, identical maximum-token limits need not produce equal usage. Report actual usage alongside caps.

Latency needs a clock boundary. Does measurement include queueing, network time, tool calls, and retries? Median latency describes a typical request; a high percentile helps expose slow cases. Throughput under concurrent load answers a different question from single-request delay.

Preserve failures. Dropping timed-out high-effort requests can manufacture a favorable accuracy estimate. Repeating only unsuccessful low-effort requests changes the policy and its cost. Selecting the best of many experiments after viewing the test set also introduces selection bias. Reserve evaluation data and state how settings were chosen.

10 Correctness and faithfulness answer different questions

A response may have a correct final answer, invalid intermediate arithmetic, and elegant prose. Score these separately. Even a mathematically valid explanation may be a reconstruction rather than the causal route that produced the answer.

Lanham and colleagues intervened on generated solution text, including truncation and inserted mistakes, and measured changes in answers. Dependence on that text varied across their models and tasks. These interventions probe particular faithfulness failures; they do not recover a complete neural mechanism or establish that all explanations are useless. Lanham et al., Sections 2–3

For practical use, ask for checkable support: equations, evidence, tests, assumptions, and uncertainty. Verify those artifacts. For a mechanistic claim, seek causal evidence about the computation. Fluency is evidence that text is fluent; it is not a substitute for either test.

11 Check your understanding

Record your answer before consulting the explanations.

1. A fixed checkpoint produces four candidates instead of one. Which state must change, and which need not?
2. Under independent $p = 0.4$ sampling, is 0.8704 the accuracy of every four-candidate selector?
3. In the imperfect-verifier example, what rises with N : coverage, accepted-only accuracy, or both?
4. Why can two populations with single-candidate accuracy 0.4 have different four-candidate availability?
5. A product's high setting returns twice as many displayed words. What does this establish about weights and hidden computation?
6. High effort scores better after all incomplete requests are discarded. What is missing?
7. A correct explanation changes the answer when edited. Does that establish a complete faithful mechanism?

i Answers and reasoning

1. More candidate-generation work and intermediate state are required; learned weights need not change.
2. No. It is availability under the stated assumptions. Selection success depends on the selector.
3. Coverage rises; accepted-only accuracy stays 6/7 in that specific independent model. This need not hold under shared failures.
4. Marginal accuracy hides request difficulty and dependence. The mixture example gives 0.4992 availability instead.
5. It establishes a displayed-length difference. Documentation and usage records are needed for stronger claims.
6. All-request outcomes, failed-run costs, and the predefined treatment of incomplete outputs. The retained subset may differ systematically.

7. No. It is evidence of dependence under that intervention, which is narrower than a complete causal account.

12 More Learning

1. Wei et al., Chain-of-Thought Prompting. Read the prompting comparison, then identify which claims are specific to the tested models and tasks.
2. Wang et al., Self-Consistency. Compare final-answer aggregation with a learned best-of- N scorer.
3. Lightman et al., Let's Verify Step by Step. Focus on the fixed-generator scope and the two supervision regimes.
4. Snell et al., Scaling LLM Test-Time Compute Optimally. Follow the difficulty-dependent allocation and compute-accounting assumptions.
5. OpenAI reasoning guide. A provider-specific reference for controls and usage accounting; recheck your chosen model before experimenting.
6. Lanham et al., Measuring Faithfulness. Examine what each intervention can rule out and what remains unresolved.