Natural Language and Speech Processing

Guest lecture: Reasoning in Large Language Models

Aivo Olev

What this lecture covers

- What "reasoning" denotes in modern LLM systems
- Why allocating extra compute at inference matters
- How reasoning models are trained — focus on RL with verifiable rewards (RLVR) and the DeepSeek-R1 recipe
- Practical demos: code, math, long context, planning
- Reasoning in audio language models and realtime voice

Goal: a precise mental model, not slogans. We will read the actual GRPO objective and a real R1 training trace.

Chain-of-Thought, then and now

- **2022**: prompt the model to generate intermediate steps; accuracy jumps on math/symbolic tasks [Wei+ 2022; Kojima+ 2022]
- **2024–2026**: the behavior is *trained in*, not prompted in. Models emit long CoTs by default and are graded on the final answer
- "Reasoning" today refers to the whole system: trained policy + inference procedure + tools + verifier + UI

Few-shot CoT prompt (Wei+ 2022)

```
Q: Roger has 5 tennis balls. He buys 2 more cans of 3 balls each. How many does he have?
A: Roger started with 5. 2 cans × 3 = 6. 5 + 6 = 11. The answer is 11.
```

Modern reasoning model (sketch)

<think>Let me set up: 5 + 2·3 = 11. Double-check: 2 cans, each with 3 balls, that is 6 new balls. 5+6=11. Correct.</think>
`<answer>11</answer>`

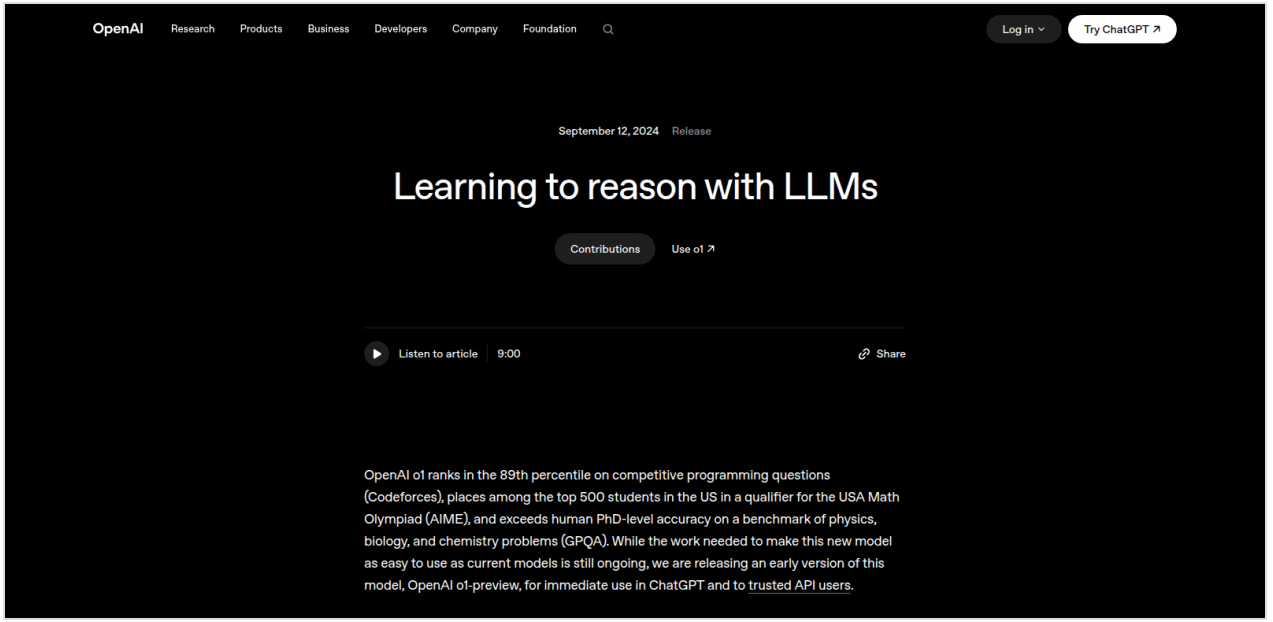
Provider surfaces in 2026

Provider	Reasoning model	Developer control	Visibility
OpenAI	o-series, gpt-5.5	`reasoning_effort` = minimal/low/medium/high/xhigh	Reasoning tokens counted & billed; not returned
Anthropic	Claude Opus 4.7	`thinking` block with `budget_tokens`; "adaptive thinking"	Signed/redacted thinking blocks preserved across turns
Google	Gemini 3.x Pro	`thinkingLevel`	Thought summaries (compressed)
DeepSeek	R1, V4 (open-weight)	Unified thinking / non-thinking switch	Full CoT exposed; enables distillation & research
Open-weight	Qwen3, gpt-oss-120b, R1-distill	Same template as above	Full traces visible — most academic study runs here

The underlying systems idea is converging: a learned policy that emits intermediate tokens, optionally calls tools, gated by a developer-controlled budget. The user-visible surface differs.

OpenAI: the o-series

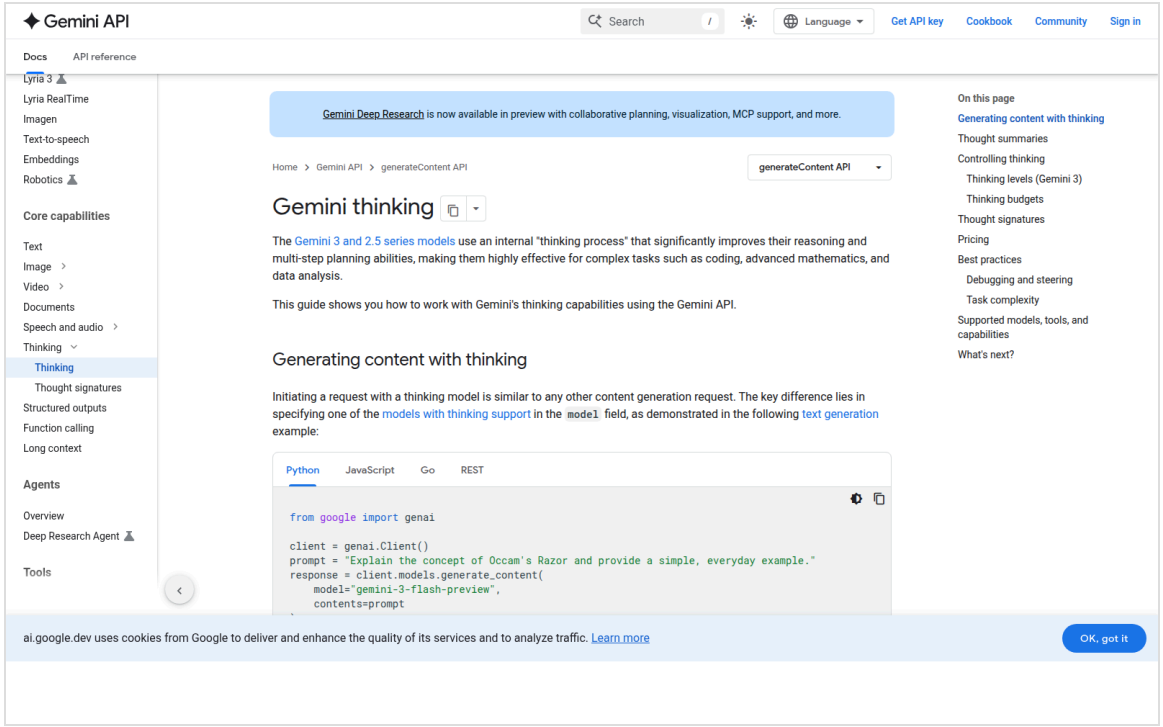
- Launched o1 in Sept 2024; first model framed explicitly as "trained with large-scale RL to reason"
- Reasoning tokens are part of the computation, counted in token usage but typically not returned verbatim
- `reasoning_effort` on gpt-5.5: minimal / low / medium / high / xhigh
- The detailed training recipe is undisclosed — most public technical detail about how this works actually comes from DeepSeek-R1 (which we will read)



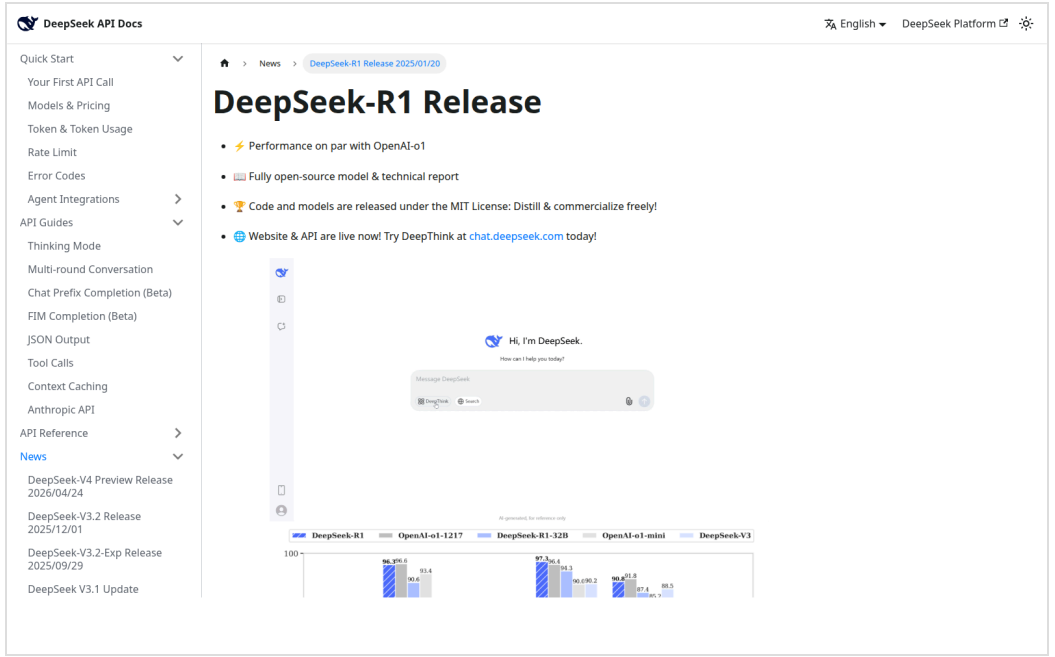
"OpenAI o1 ranks in the 89th percentile on Codeforces … exceeds human PhD-level accuracy on GPQA."

Source: openai.com/index/learning-to-reason-with-llms

Google & DeepSeek



Gemini 3 thinking docs — exposes `thinkingLevel` and thought summaries.



DeepSeek-R1 release notes — first frontier open-weight reasoning model (Jan 2025).

DeepSeek-R1 matters disproportionately for this course: it is the most fully documented reasoning recipe in the open literature, and the basis of nearly every academic replication.

Raw CoT is not the product surface

- CoT is **not guaranteed faithful** — models can produce plausible rationales that do not reflect the causal computation [Turpin+ 2023; Lanham+ 2023]
- Safety: leaked context, jailbreak amplification, prompt-injection targeting the trace itself
- Production stacks prefer summarised thinking, cited evidence, structured tool traces, verifier outcomes, constrained final answers
- This is why most APIs *do not* return the raw CoT, even when one exists internally
- Commercial reason on top of that: raw chains are training-grade data. R1-Distill showed a small base fine-tuned on harvested teacher chains recovers most of the parent's reasoning — publishing the chain publishes the recipe

Turpin+ 2023, simplified example

Add a biasing few-shot example where the "correct" answer is always (A). The model still emits a step-by-step CoT — but now justifies (A) regardless of the question.

Conclusion: CoT can be a post-hoc rationalisation, not an audit trail.

Cost model

$$\text{total\_cost} \approx N_{\text{in}} \cdot \text{p}_{\text{in}} \;+\; (N_{\text{reasoning}} + N_{\text{out}}) \cdot \text{p}_{\text{out}} \;+\; \Sigma\ \text{tool\_costs}$$

- Reasoning tokens often **dominate** on hard tasks — a single R1 response can be 10k+ tokens
- Latency moves from sub-second to tens of seconds, sometimes minutes for agents
- Reasoning **helps** on tasks that are compositional, verifiable, high-stakes, or tool-using
- Reasoning **hurts** on lookups, paraphrases, stylistic edits — empirically degrades accuracy at high effort

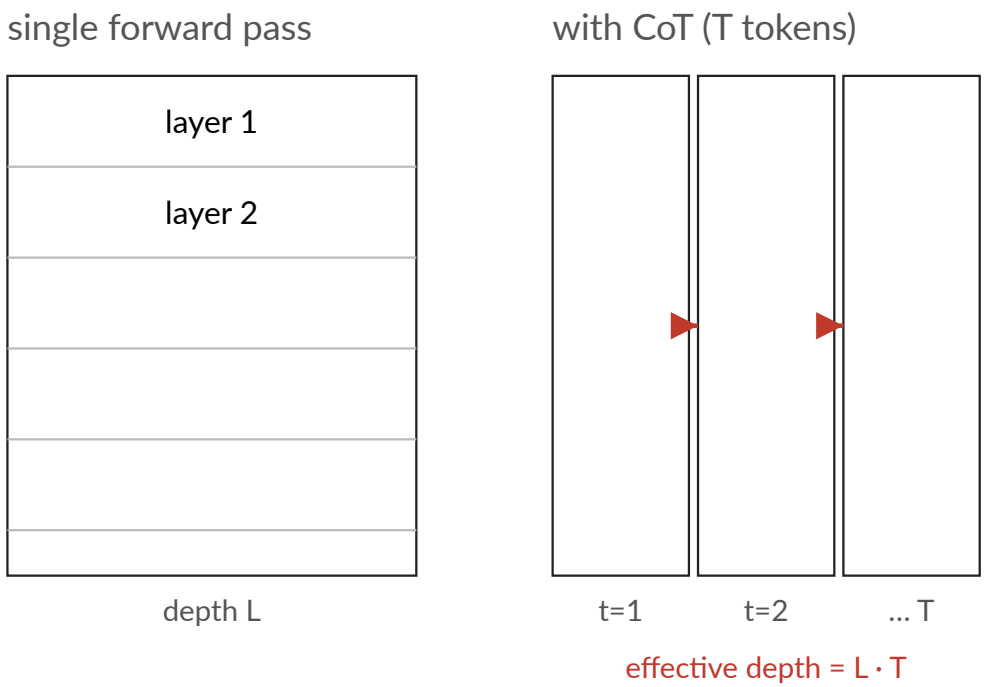
  [Sprague+ 2024; Liu+ 2024]

PART 2

Why extra inference-time computation matters

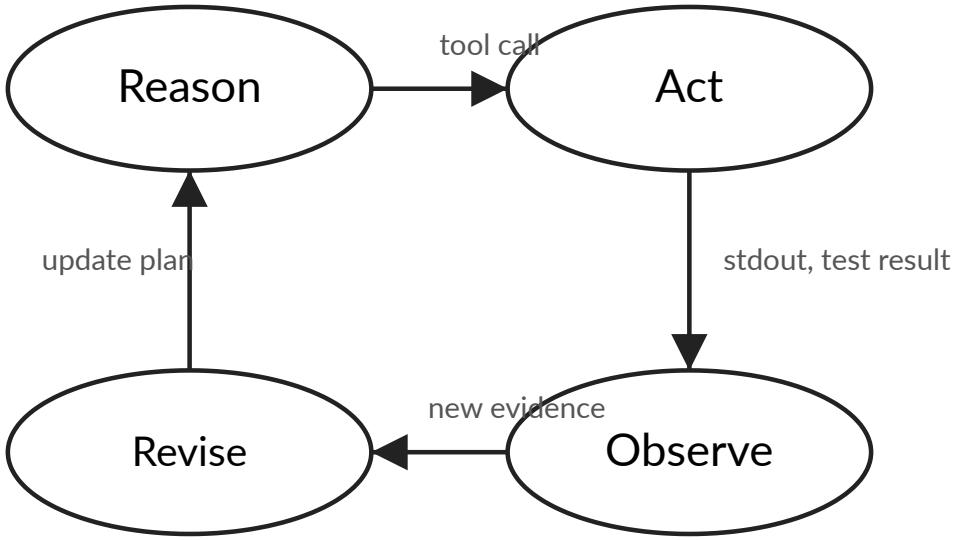
Expressivity: CoT changes what is computable

- A standard decoder with fixed depth L and bounded precision computes a fixed function class in **one forward pass**
- Autoregressive generation reuses the same parameters but adds a new layer of conditional computation per token
- Formal result [Merrill & Sabharwal 2024; Feng+ 2023]: with CoT, transformers can simulate algorithms that are **provably out of reach** of constant-depth, finite-precision single-pass models — sorting, modular arithmetic, graph traversal
- So CoT is not "thinking out loud" — it is depth via tokens



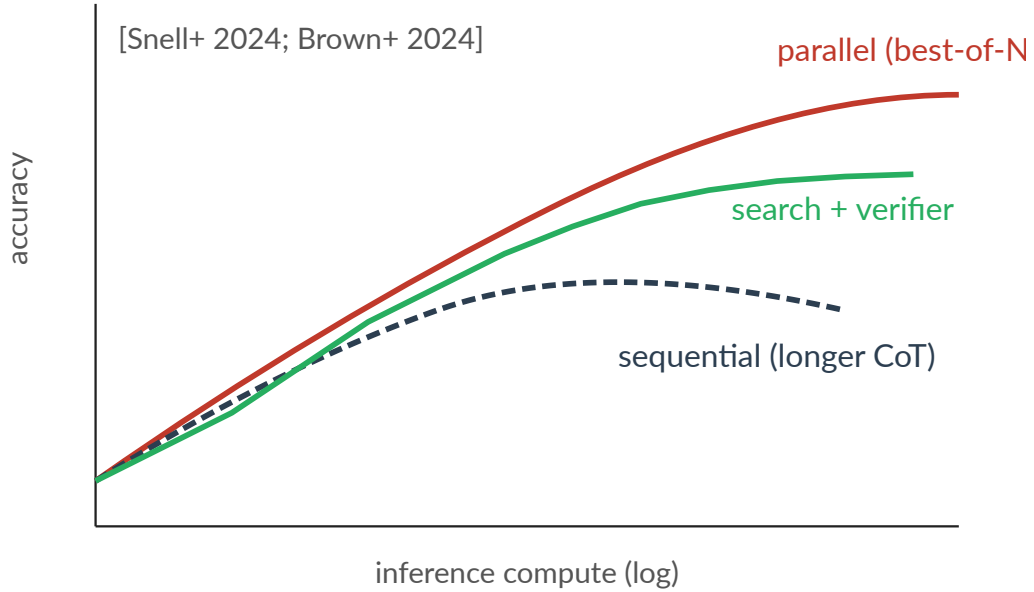
State + feedback: the ReAct loop

- Generated tokens, tool outputs, retrieved passages all enter the KV cache as *working memory*
- The model can re-read its own intermediate state, condition on tool feedback, and self-correct
- This pattern is the mechanism behind:
  - ReAct [Yao+ 2022]
  - Self-Refine [Madaan+ 2023]
  - Reflexion [Shinn+ 2023]
- Today every code-agent (SWE-agent, Claude Code, Cursor) is a variant of this loop



Test-time compute scaling

- **Snell+ 2024**: extra inference FLOPs (search, verifier, refinement) can beat scaling parameters at fixed compute
- **Brown+ 2024 ("Large Language Monkeys")**: pass@k scales near-log-linearly in k — given a verifier, even a weak base model can cover hard problems
- Two regimes:
  - **Sequential** — longer CoT, refinement chains; saturates and can reverse
  - **Parallel** — best-of-N, self-consistency; depends on verifier or majority validity
  - **Search** — Tree of Thoughts, MCTS, beam over partials



Tools turn reasoning into external computation

- Tools offload subproblems to systems that are **exact and verifiable**:
  - Python / shell — arithmetic, simulation, code execution
  - Retrieval — evidence and citations
  - Theorem provers (Lean, Coq) — formal proof
  - Browser / computer-use APIs — workflow tasks
- Toolformer [Schick+ 2023] and PAL [Gao+ 2022] established that LMs can interleave tool calls with text
- Verifier-driven loops (generate → check → repair) are the dominant pattern in modern code agents
- An interesting empirical fact: DeepSeek-R1 was trained *without* tools and still does well — but its successors explicitly add tool RL

Test-time compute is a routing problem

- The deployment question is rarely "use reasoning or not?" — it is "what is the minimum compute that meets the reliability target?"
- Heuristics:
  - Route by task type (chat vs. code vs. math)
  - Route by detected ambiguity or retrieval completeness
  - Route by explicit difficulty classifier
  - Route by user tier and SLA
- A small model + verifier + tools can beat a frontier model with no scaffolding on bounded workloads (see SWE-bench leaderboard)

PART 3

How reasoning models are built

The typical training pipeline

stage 3 is where most of the reasoning behaviour appears

- Stages 1–2 produce a competent next-token predictor. Stage 3 produces a **reasoning policy**.
- Stage 4 keeps the model useful and safe without destroying stage-3 capability — that is the hard part.

GRPO: the actual objective

For each prompt q, sample a group of G completions from the old policy. Score each with a (possibly rule-based) reward r_i. Compute the **within-group normalised advantage**:

$$A_i = (r_i - \text{mean}(r_{1..G})) \;/\; \text{std}(r_{1..G})$$

Maximise the clipped, KL-regularised group objective:

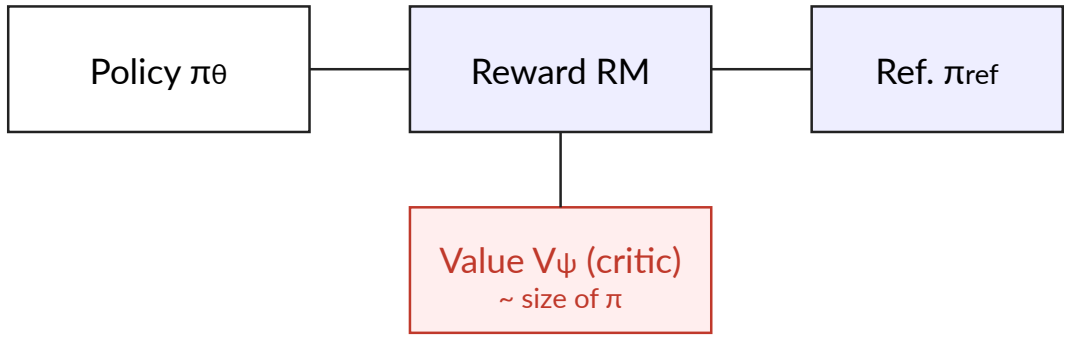
$$J_{GRPO}(\theta) = \mathbb{E}_{q,\,\{o_i\}} \quad (1/G)\;\Sigma_i\;[\;\min(\;\rho_i\;A_i,\quad \text{clip}(\rho_i,\,1-\varepsilon,\,1+\varepsilon)\;A_i\;)\;\;-\;\;\beta\cdot\mathbb{D}_{KL}(\pi_\theta\,\|\,\pi_{ref})\;]$$

$\rho_i = \pi_\theta(o_i|q) \;/\; \pi_{\theta_{old}}(o_i|q)$. Schulman's positive unbiased KL estimator is added directly to the loss, not folded into per-token rewards (as PPO does).

Shao et al., *DeepSeekMath*, 2024 — equation (3).

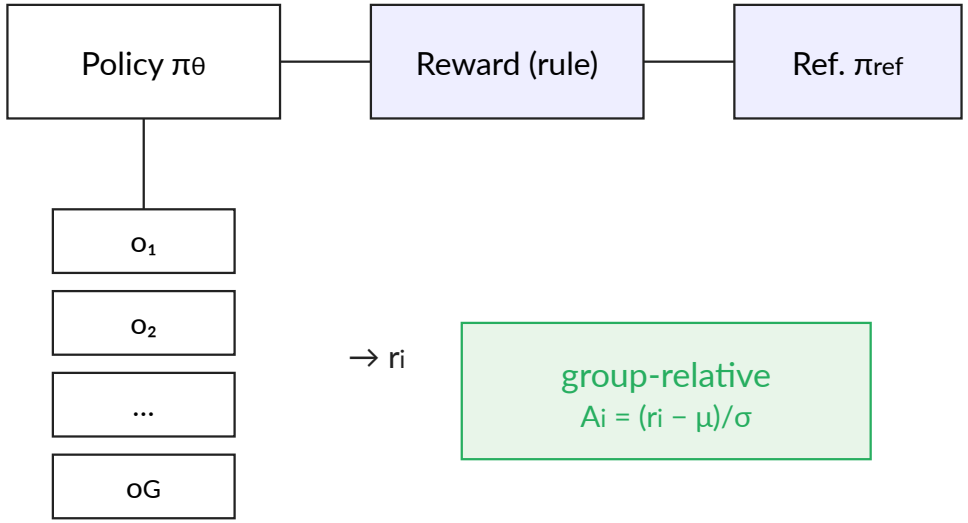
PPO vs GRPO

PPO



→ GAE advantage; KL added per-token to reward

GRPO

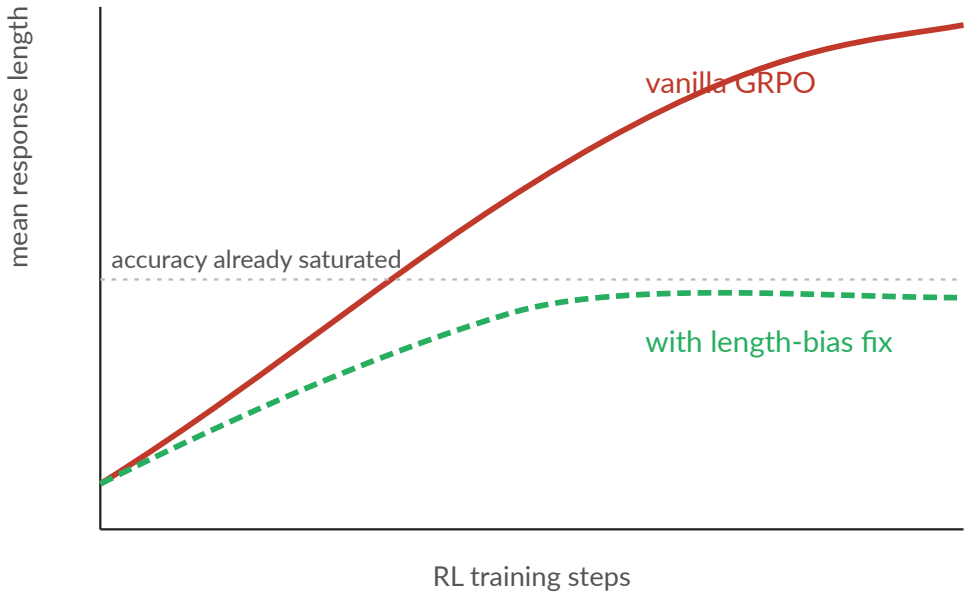


No critic. KL added directly to loss.
Halves the parameter memory of the RL stage.

Shao+ 2024; DeepSeek-R1 2025; figure adapted from DeepSeekMath Fig. 4.

The length-inflation pathology, and what followed

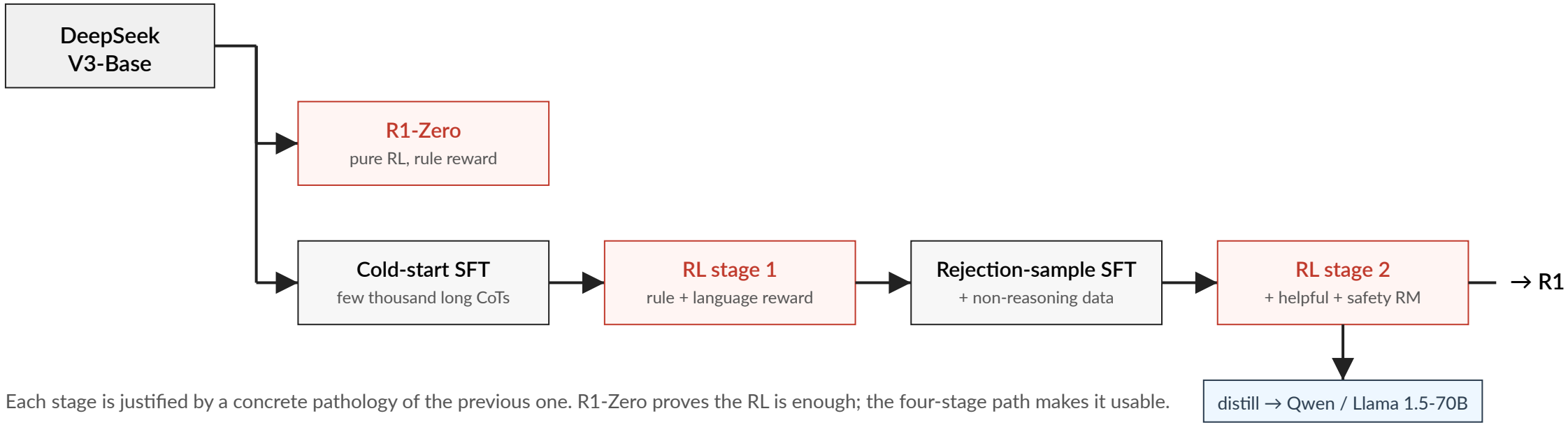
- Mean response length under vanilla GRPO keeps climbing well after benchmark accuracy plateaus — extra tokens, no extra correctness.
- The objective divides each rollout's contribution by its own length $1/|o_i|$. Under *negative* advantage that concentrates the penalty on short failures and spreads it thinly across long ones — the gradient prefers lengthening failures to fixing them.
- **DAPO** [Yu+ 2025]: one batch-level token count instead of per-rollout normalisation; every token contributes equally.
- **Dr. GRPO** [Liu+ 2025]: drop the length term entirely.
- After the fix: length of *correct* trajectories barely moves; *incorrect* ones get markedly shorter. Accuracy preserved, cost down.



Schematic: accuracy plateaus first; vanilla length keeps drifting up until the normaliser is corrected.

Yu et al., *DAPO*, 2025. Liu et al., *Dr. GRPO*, 2025.

DeepSeek-R1: the four-stage pipeline



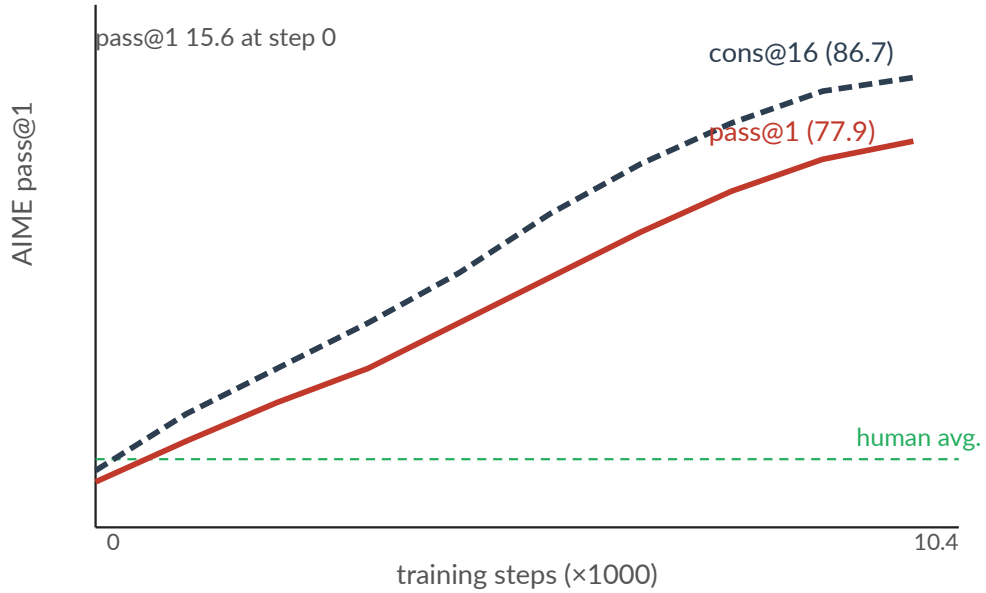
Each stage is justified by a concrete pathology of the previous one. R1-Zero proves the RL is enough; the four-stage path makes it usable.

- Stage-2 RL is scoped asymmetrically: helpfulness is graded only against the visible answer, while harmlessness is graded across the entire rollout — including the hidden chain. Disallowed content cannot hide inside the private monologue.
- "Distill →" here is plain SFT on chains sampled from R1; the student is not asked to match per-token distributions. At small parameter budgets this outperforms running RLVR on the small base directly.

DeepSeek-AI, *DeepSeek-R1*, Nature 2025 — Figure 2.

R1-Zero: pure RL, no SFT

- GRPO applied directly on V3-Base. Fixed template `<think>…</think><answer>…</answer>`
- Reward = `accuracy` + `format` only. Math: boxed-answer string match. Code: compiler + unit tests. No neural reward model.
- Hyperparams: lr 3e-6, β = 0.001, T = 1.0, G = 16 rollouts, 32k–65k context, 10.4k steps, batch 512 (~1.6 epochs)
- AIME 2024 pass@1: **15.6 → 77.9**; cons@16: **86.7** (above human-participant average)
- Two qualitative defects in the chains: occasional code-switching between natural languages within one thought, and stretches that read as syntactic noise. Verifier accuracy unaffected, but the trace is unfit as a product artefact — this is the empirical case for the cold-start SFT and language-consistency reward in the full R1 recipe



DeepSeek-R1 Figure 1 (redrawn). Reward-only training, no labelled CoTs.

The "aha moment" (verbatim trace)

Question: If $a > 1$, the sum of real solutions of $\sqrt{(a - \sqrt{(a + x)})} = x$ equals?

<think>

To solve the equation, let me square both sides… $(a - x^2)^2 = a + x$, … $x^4 - 2ax^2 - x + (a^2 - a) = 0$ …

Wait, wait. Wait. That's an aha moment I can flag here.

Let me re-evaluate this step by step. We started with $\sqrt{(a - \sqrt{(a + x)})} = x$. Square: $a - \sqrt{(a + x)} = x^2$; $\sqrt{(a + x)} = a - x^2$. Squaring again …

</think>

- Not prompted. Not in the SFT data (there isn't any in R1-Zero). Emerges **solely from the verifier-driven RL signal**.
- The frequency of the word "wait" rises sharply during a specific phase of training — Figure 9b of the R1 paper.

DeepSeek-R1, Table 2.

R1 final benchmark numbers

Benchmark	R1-Zero	R1-Dev2	R1	Notes
AIME 2024 (pass@1)	77.9	74.0	**79.8**	human participants ~40
MATH-500 (pass@1)	95.9	95.9	**97.3**	
CNMO 2024	88.1	73.9	78.8	Chinese math olympiad
Codeforces rating	1444	1687	**2029**	96.3 percentile
LiveCodeBench (pass@1-CoT)	50.0	63.5	**65.9**	
SWE-bench Verified	43.2	44.6	49.2	SWE RL was deliberately limited
GPQA Diamond (pass@1)	75.8	70.7	71.5	PhD-level physics/bio/chem
MMLU (EM)	88.5	91.2	90.8	
ArenaHard	53.6	73.2	**92.3**	+17% from stage-2 RL

DeepSeek-R1 Table 3 (selected rows). Bold marks the largest entry in each row.

What did *not* work in R1

- **Neural reward models for reasoning tasks** — both ORMs and PRMs were exploitable under large-scale RL. Authors fall back to rule-based rewards wherever a verifier exists.
- **Process reward models** — hard to scale, expensive to label, rewarded persuasive-looking but wrong steps.
- **Monte Carlo Tree Search at inference** — not used. R1 instead "dynamically allocates computational resources" by generating longer CoTs.
- **Few-shot prompting** — *consistently degrades* R1's performance. Zero-shot direct prompting recommended.
- **Tools (search, calculator)** — explicit limitation; deferred to successor models.
- **Software engineering RL at scale** — long eval latency limited the amount of SWE-RL applied; R1 is only modestly better than V3 on SWE-bench.

DeepSeek-R1, §6 "Conclusion, Limitation, and Future Work".

Outcome rewards vs process rewards

- **ORM** grades only the final answer.
  - Cheap, automatic for math/code/proof
  - Tolerates unusual but correct trajectories
  - Drives most practical progress (R1, o1, Qwen3)
- **PRM** grades individual reasoning steps.
  - PRM800K [Lightman+ 2023] showed PRMs beat ORMs on MATH *as a re-ranker at inference*
  - Labels expensive (human or model graded)
  - Reward-hackable: rewards "persuasive-looking" wrong steps [Setlur+ 2024]
- Current practice: ORMs/RLVR for training, PRMs (or LLM-as-judge) for inference selection and debugging

PRM example (one solution, six steps)

Step 1: define x. ✓

Step 2: substitute. ✓

Step 3: divide by x. ✗ ← here the trace plausibly drops a case

Step 4–6: continue confidently …

A PRM trained on this kind of labelled data can detect step 3 — but if it is itself an LLM, it can be fooled by a fluent step.

Inference techniques compound with trained reasoning

Technique	One-line	Best when
Self-consistency	sample N CoTs, majority-vote the answer	closed-form answers (math, MCQ)
Best-of-N + verifier	sample N, re-rank by learned verifier or PRM	code (tests), math (PRM/CAS)
Tree of Thoughts	expand a search tree of intermediate states	puzzles, planning
ReAct / tool loops	plan → act → observe → reason	any task with a useful tool
Self-Refine / Reflexion	generate → critique → revise	writing, code where errors are recoverable
Speculative decoding	draft with a small model, verify with the big one	orthogonal: just makes long CoTs cheaper
Budget forcing (S1)	splice continuation cues mid-stream to extend thinking, or termination cues to cut it off [Muennighoff+ 2025]	enforcing a token cap or pushing through premature stops
Latent reasoning (Coconut)	reason in a continuous latent space, not tokens	research; not yet standard

Pre-2024 these techniques carried most of the weight. After RLVR-trained reasoning models, they layer on top with diminishing but still real returns.

Trade-offs to remember

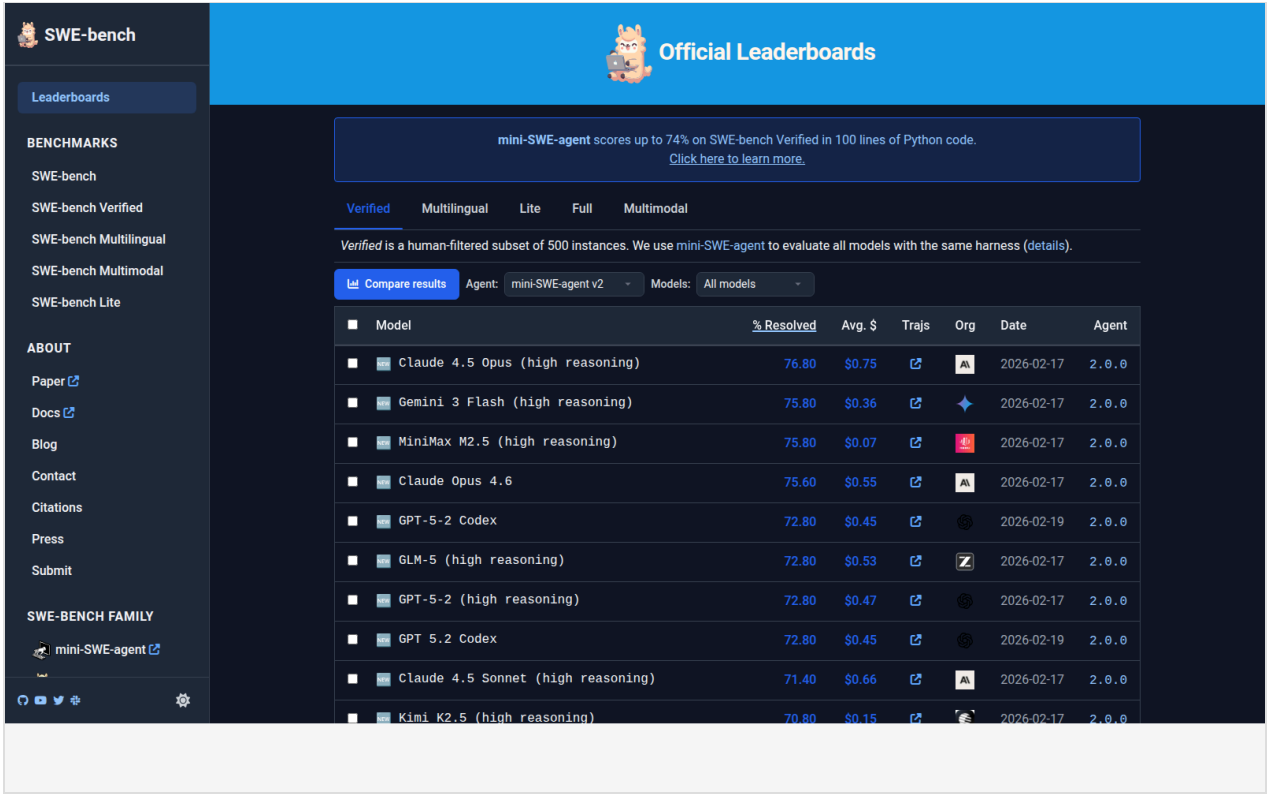
- Latency: sub-second → tens of seconds → minutes (agent runs)
- Hidden thinking tokens often dominate cost on hard tasks
- Faithfulness of CoT is not guaranteed — do not use it as an interpretability claim
- Overthinking: high effort hurts on simple tasks [Chen+ 2024, "Do NOT think that much for 2+3=?"]
- Strong instruction following amplifies prompt sensitivity — small wording changes can move accuracy several points [Mizrahi+ 2024]

PART 4

Practical demos & common failure modes

Demo 1 — Coding: failing test → patch

- Hand the model a small repo with one failing test
- Observable trace:
  - read the failing test & relevant source
  - form a hypothesis
  - patch minimally
  - run `pytest`
  - revise & summarise
- The verifier (test suite) is what makes RLVR and inference-time verification both work
- SWE-bench Verified scores went from <5% in 2023 to >60% in early 2026 — one of the biggest single-year capability shifts in NLP



SWE-bench Verified leaderboard. Each row is an agent + model + scaffold combination.

Demo 2 — Math with a verifier loop

- Pick a problem where the most-plausible first answer is wrong (e.g. an AIME geometry problem)
- Workflow:
  - solve the problem in CoT
  - have the model **write a Python checker** (or invoke a CAS)
  - run it; if mismatch, revise; loop
- Benchmarks: GSM8K, MATH, AIME 2024/25, FrontierMath, Putnam-AXIOM
- Frontier reasoning models exceed 90% on MATH and approach human-expert on AIME

````
<think>
Let r be the radius. Then r² + 7² = (r+3)²
=> r² + 49 = r² + 6r + 9
=> 6r = 40, r = 20/3.
Let me double-check with a script.
</think>

```python
r = 20/3
assert abs(r**2 + 49 - (r+3)**2) < 1e-9
print(r)  # 6.666...
```
<answer>r = 20/3</answer>
````

Solver writes its own checker — verifier-driven inference.

Demo 3 — Long-context document reasoning

- Task: "Can this customer get a refund, and which clause decides it?"
- Inputs: policy PDF, customer notes, receipt record (~30k tokens)
- Failure modes:
  - right clause, wrong date scoping
  - right answer, no citation
  - two clauses conflict — unresolved
- Better workflow: retrieve → quote → reconcile → abstain when underdetermined
- Long-context benchmarks: LongBench, Loong, RULER, FRAMES. Note Lost-in-the-Middle [Liu+ 2023]

Expected final answer shape

Eligible: Yes (partial refund, store credit).

Decided by: Clause §4.2 "purchase within 30 days, defect" — receipt dated 2026-04-22 (21 days ago).

Caveat: Clause §4.5 excludes electronics opened beyond 14 days. Cannot resolve without explicit category — *ask user*.

When reasoning hurts

- **Simple lookups** — extra latency, no accuracy gain
- **Missing-information prompts** — long CoT increases coherent fabrication
- **Subjective writing** — high effort often produces stiff, over-optimised prose
- **Real-time UX** — voice or autocomplete: the user is gone before you finish thinking

Operational rule: keep a default fast path, escalate to reasoning only when complexity, risk, or verification justifies it, and **measure the router with offline evals** before deploying.

Prompting reasoning models

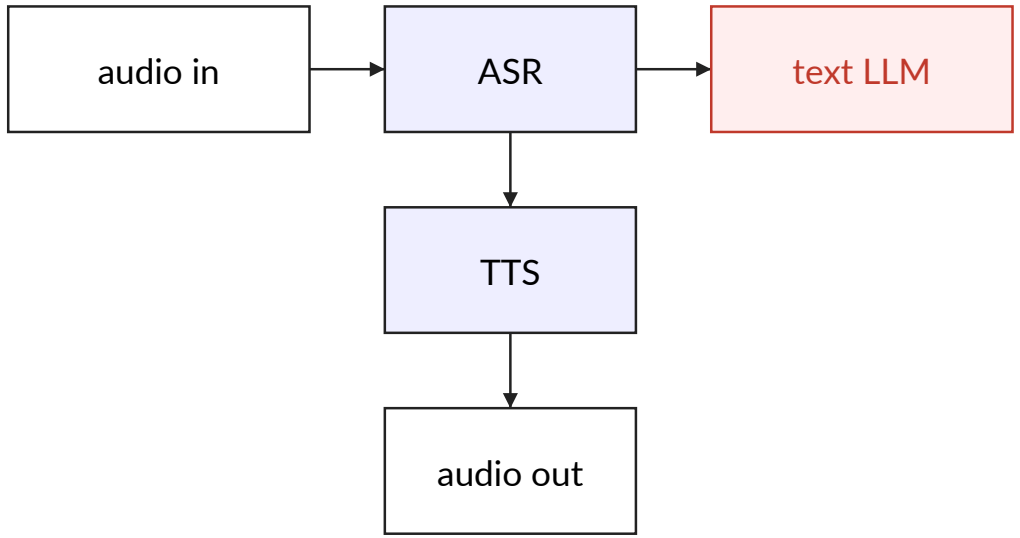
- Provide goals, constraints, data, and **explicit success criteria**. Criteria you can verify become reusable as offline evals.
- Prefer "verify against these tests/criteria" over "think step by step" — the latter is largely subsumed by trained behaviour [Sprague+ 2024]
- Use native effort controls instead of prompt-side hacks like "think harder"
- Return final answers with evidence / tool traces / uncertainty, not raw private reasoning
- R1-specific: **do not few-shot prompt R1**. Few-shot consistently degrades it. Zero-shot, direct, format-explicit prompts only.
- Log per-request: input + output + reasoning tokens, latency, tool calls, retries, verifier verdicts

PART 5

Reasoning in audio and realtime voice

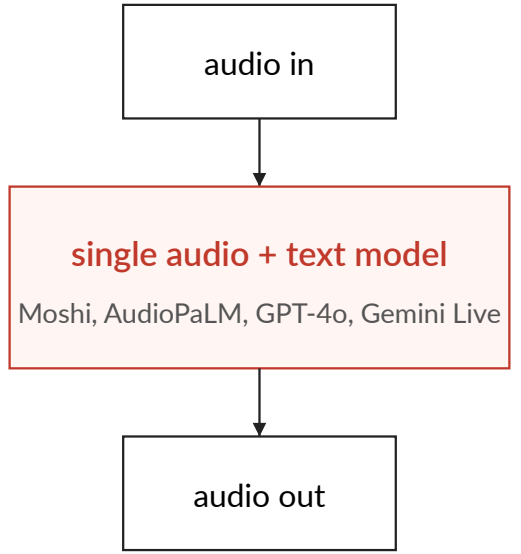
Two architectures for audio

Cascaded



+ modular, mature
− adds latency, loses prosody/emotion

Native (end-to-end)



+ low latency, paralinguistic, turn-taking
− text-RL recipes don't transfer cleanly

Audio-specific reasoning problems

- **Acoustic reasoning** — speaker identity, turn-taking, prosody/emotion, background sound, event localisation
- **Long-form audio** — meetings, lectures, calls: diarisation, segmentation, retrieval, longitudinal memory
- **Interruptions** — mid-CoT, the user revises a constraint; model must integrate without replay
  - Moshi [Defossez+ 2024] models user + model channels in parallel to interrupt itself
- **Latency budget** — natural turn-taking ≈ 200–500 ms; rules out long visible CoT

Voice demo (mid-turn interrupt)

User: 30 min, two vegetarians, one mushroom-hater, tiny kitchen.

Model: "Let me check — okay, I'd suggest a chickpea stew with—"

User (interrupting): "Actually one guest is gluten-free."

Model: "Got it. Same stew, swap the bulgur side for rice."

Good behaviour: integrate, do not replay, ask at most one clarifier.

Benchmarks: Big Bench Audio, MMAU, AIR-Bench, Audio-Reasoner-Bench. Less settled than text/code evals.

Three questions for any task

1. Is this task **reasoning-bound**, **knowledge-bound**, **retrieval-bound**, or **execution-bound**?
2. What **observable check** tells us the model succeeded?
3. What is the **cheapest reasoning path** that meets the reliability target?

A good production stack answers these for every request, automatically, before deciding which model and effort level to route to.

References (selected)

Core reasoning & CoT

- Wei et al. — *Chain-of-Thought Prompting*, NeurIPS 2022
- Kojima et al. — *Zero-Shot Reasoners*, NeurIPS 2022
- Wang et al. — *Self-Consistency*, ICLR 2023
- Yao et al. — *ReAct*, ICLR 2023 / *Tree of Thoughts*, NeurIPS 2023
- Madaan et al. — *Self-Refine*, NeurIPS 2023
- Shinn et al. — *Reflexion*, NeurIPS 2023

Theory & faithfulness

- Merrill & Sabharwal — *Expressive Power of Transformers with CoT*, ICLR 2024
- Turpin et al. — *Models Don't Always Say What They Think*, NeurIPS 2023
- Sprague et al. — *To CoT or Not to CoT*, 2024

Test-time compute & verifiers

- Cobbe et al. — *Training Verifiers (GSM8K)*, 2021

Training: RLHF, DPO, RLVR

- Ouyang et al. — *InstructGPT*, NeurIPS 2022
- Rafailov et al. — *DPO*, NeurIPS 2023
- Shao et al. — *DeepSeekMath* (introduces GRPO), 2024
- DeepSeek-AI — *DeepSeek-R1*, Nature 2025
- Yu et al. — *DAPO*, 2025 (GRPO length-bias fix)
- Liu et al. — *Dr. GRPO*, 2025 (drops length normaliser)
- Muennighoff et al. — *S1*, 2025 (budget forcing)
- OpenAI — *o1 system card*, 2024

Tools & agents

- Schick et al. — *Toolformer*, NeurIPS 2023
- Gao et al. — *PAL*, ICML 2023
- Jimenez et al. — *SWE-bench*, ICLR 2024

Audio & LALMs

- Rubenstein et al. — *AudioPaLM*, 2023