REASONING MODELS

How LLMs Learn to Think Step-by-Step

Part 8

Large Language Models: From Foundations to Modern Trends

REASONING MODELS

"Learning to reason with reinforcement learning"

- How models can think step-by-step
- Multi-step reasoning mechanism
- Training with pure reinforcement learning
- Emergent behaviors and worked examples

What is a Reasoning Model?

DEFINITION

A Large Language Model trained (via Reinforcement Learning) to generate reasoning traces before producing the final answer.

Traditional LLM	Reasoning Model
Single-pass (System 1)	Multi-step deliberation (System 2)
Direct answer only	<think>reasoning</think> + answer
~100-1,000 tokens	~1,000-100,000 tokens
1-5 seconds latency	10s - 5+ minutes latency
SFT + RLHF training	Pure RL / RLVR training

Source: DeepSeek-R1, 2025

System 1 vs System 2 Thinking

SYSTEM 1

Fast & Intuitive

- Automatic, effortless
- Pattern recognition
- Quick decisions
- Prone to biases

Example: "What is 2+2?"

SYSTEM 2

Slow & Deliberate

- Conscious, effortful
- Logical reasoning
- Step-by-step analysis
- Accurate but slow

Example: "Solve 17×23"

KEY: Reasoning Models = LLMs that can engage System 2 thinking

Source: Kahneman, 2011

Evolution: From CoT to Reasoning Models

KEY INSIGHT

Traditional LLMs: Trained to IMITATE human reasoning
Reasoning Models: Trained to DISCOVER effective reasoning strategies via RL

Why Do We Need Reasoning Models?

THE PROBLEM

"This success [of CoT prompting] is heavily contingent upon extensive human-annotated demonstrations... by constraining models to replicate human thought processes, their performance is inherently capped by the human-provided exemplars."

Limitations of Traditional Approaches:

1. **Human-Capped**
   Performance limited by human examples
2. **Fixed Compute**
   Same thinking time for easy & hard problems
3. **No Self-Correction**
   Cannot backtrack or verify answers

THE SOLUTION: Pure RL Training

Let the model DISCOVER its own reasoning strategies through trial and error, rewarded only for correct answers.

Source: DeepSeek-R1, 2025

Chain-of-Thought vs Reasoning Models

Chain-of-Thought (2022)

- Prompt engineering technique
- "Let's think step by step"
- Manual, not learned
- Format depends on user
- Limited self-correction
- No length calibration
- Capped by human examples

VS

Reasoning Models (2024+)

- Trained behavior via RL
- Model learns WHEN to reason
- Automatic, learned
- Consistent <think> format
- Emergent self-correction
- Adaptive reasoning length
- Can exceed human strategies

Key Reasoning Models

OpenAI o1 (Sep 2024)

- First production reasoning model
- Hidden thinking (not visible)
- Closed-source, API only
- Strong math/code performance

DeepSeek R1 (Jan 2025)

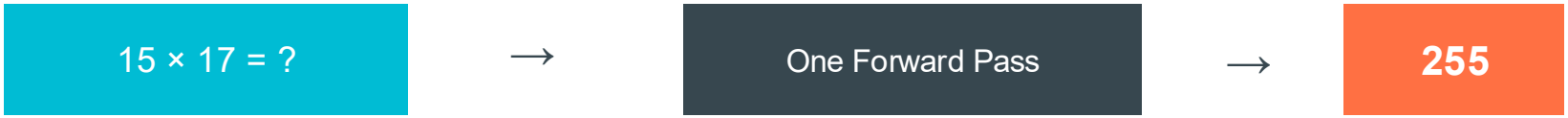
- Open-source with MIT license
- Visible thinking (<think> tags)
- Pure RL approach documented
- Matches/exceeds o1 on benchmarks

Performance Comparison (AIME 2024)

Source: DeepSeek-R1 Paper, 2025

Traditional LLM vs Reasoning Model

Traditional LLM:



Reasoning Model:

KEY DIFFERENCE

Traditional: Must compute everything in ONE forward pass (fixed compute)
Reasoning: Breaks down into MULTIPLE steps, each step can reference previous ones
→ **Thinking tokens act as WORKING MEMORY**

CORE MECHANISM

How Models Can "Think"

- The fundamental insight behind reasoning
- Autoregressive generation enables thinking
- Attention as working memory
- Test-time compute scaling

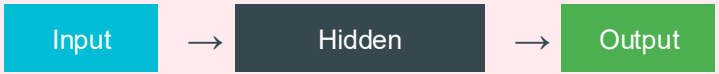
The Fundamental Insight

KEY REALIZATION

"Thinking" in LLMs is NOT magic — it's just TEXT GENERATION!

✗ INCORRECT VIEW

"Model thinks internally, then outputs result"



✓ CORRECT VIEW

"Thinking IS the generated text tokens"

Each token can "see" all previous tokens via Attention!

► THINKING = VISIBLE TEXT TOKENS

► MORE TOKENS = MORE "COMPUTE" = BETTER ANSWERS

Autoregressive Generation

$$P(\text{output}|\text{input}) = \prod P(\text{token}_t \mid \text{input}, \text{token}_1, \text{token}_2, \ldots, \text{token}_{t-1})$$

Each new token has ACCESS to ALL previous tokens → Has context

Generation Process for "15 × 17 = ?":

Step 1	`P("<think>" | question)`	*Model decides to start thinking*
Step 2	`P("Let" | question, "<think>")`	*Begins reasoning*
Step 3	`P("me" | question, "<think>", "let")`	*Continues*
Step N	`P("150" | question, "<think>", "Let me break this down:","15 × 17 = 15 × (10 + 7) = 15 × 10 + 15 × 7 = ")`	*Has context to compute 15×10*
Step N+1	`P("+" | ..., "150")`	*Continues with addition*
Step N+2	`P("105" | ..., "150", "+")`	*Continues with addition*
Step N+n	`P("255" | ..., "150 + 105=")`	*Attend to "150" and "105" to compute sum.*

KEY: Intermediate tokens ("150", "105") serve as WORKING MEMORY

Attention: The Mechanism Behind Thinking

$$\text{Attention}(Q, K, V) = \text{softmax}(QK^T / \sqrt{d_k}) \times V$$

In Reasoning Context:

Q (Query)	"What am I looking for?"	*Token "=" asking for sum result*
K (Key)	"What info is available?"	*"150" and "105" have relevant keys*
V (Value)	"Actual content"	*Numerical values 150 and 105*
Softmax	"Which tokens matter most?"	*High weights on operands*

Attention Weights when generating "255" after "150 + 105 =":

"15"	"×"	"17"	"="	"150"	"+"	"105"
0.02	0.01	0.02	0.04	**0.38**	0.12	**0.41**

Source: Vaswani et al., 2017

With vs Without Reasoning Tokens

WITHOUT Reasoning Tokens:

15 × 17 = ? → ONE pass → **255**

Problem: Must compute 15×17 entirely within ONE forward pass — no scratchpad!

WITH Reasoning Tokens:

15×17=? → <think> → 15×10 → =150 → +15×7 → =105 → sum= → </think> → 255

Each step can ATTEND to previous steps — builds on intermediate results!

ANALOGY

Without Reasoning = Mental Math

Hard for complex problems

With Reasoning = Paper & Pencil

Can solve much harder problems!

Why This Works: Computational View

Traditional LLM	Reasoning Model
Compute = FIXED (one forward pass) Easy: 2+2 → Uses full capacity Hard: ∫...dx → Same capacity!	**Compute = ADAPTIVE** (scales with thinking tokens) Easy: 2+2 → Few tokens → Fast Hard: ∫...dx → Many tokens

Mathematical Formulation:

Traditional	Reasoning
$O(L \times d^2)$ Fixed compute per forward pass	$O(N \times L \times d^2)$ N = thinking tokens (variable!)

TEST-TIME COMPUTE SCALING: More thinking = More computation = Better accuracy

Test-Time Compute Scaling

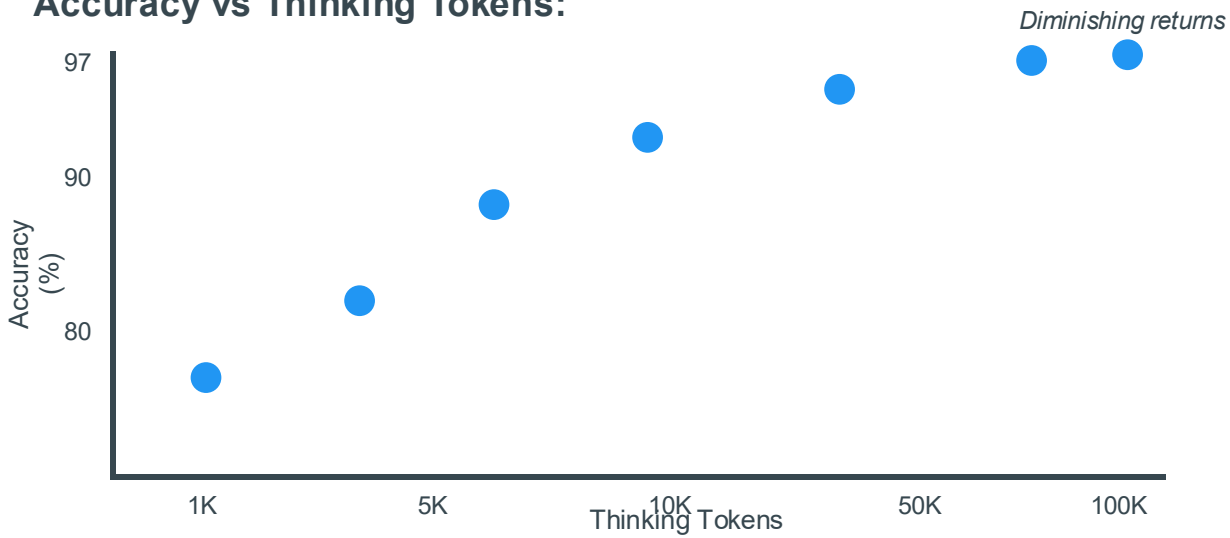
Traditional Scaling (Pre-2024)

More training compute → Better model

Test-Time Scaling (2024+)

Same model + More thinking → Better answers

Accuracy vs Thinking Tokens:



Key Observations:

- More tokens → Higher accuracy
- Diminishing returns after ~10-20K
- Model learns to calibrate length
- **This EMERGES from RL!**

Source: OpenAI o1 Blog, 2024

EMERGENCE

Multi-Step Reasoning

- How model learns to break down problems
- Discovery: Self-verification & backtracking
- Adaptive thinking length
- All emerges from RL training!

The Core Questions

Previous part explained WHY model CAN reason (via Attention mechanism)

This part answers HOW model LEARNS these behaviors:

1. How does model KNOW to break down problems?
2. How does model learn self-verification?
3. How does model learn backtracking ("Wait, that's wrong...")?
4. How does model know WHEN to stop thinking?

ANSWER: All these behaviors EMERGE from Reinforcement Learning!

The RL Training Loop (GRPO)

$$A_i = \frac{r_i - \text{mean}(r)}{\text{std}(r)}$$

Compare within GROUP: o_i correct & others wrong → $A_i >> 0$ (strong reward)

Example Responses for "What is 15 × 17?":

o_1: "255" (direct)	r=1	+0.2
o_2: "Let me think... 15×17=255"	r=1	+0.2
o_3: "15×17=245" (wrong)	r=0	-0.8
o_4: "<think>step-by-step...</think> 255"	r=1	+0.2

KEY: Model explores DIVERSE strategies → RL reinforces SUCCESSFUL ones

Source: DeepSeek R1, 2025

Discovery 1: Breaking Down Problems

Problem: "Find x if 3x + 7 = 22, then calculate 5x - 3"

Response A: Direct Approach

"The answer is 22"

Model attempts to compute
everything in ONE forward pass

Accuracy: ~40%

Response B: Step-by-Step

"<think>
3x + 7 = 22 → 3x = 15 → x = 5
5(5) - 3 = 25 - 3 = 22 ✓
</think> Answer: 22"

Accuracy: ~95%

RL Learning Signal:

- Response B consistently gets HIGHER rewards
- RL increases P("First", "solve for", "Then", "Calculate")
- Over time, model LEARNS to break down problems automatically
- **This is NOT programmed — it EMERGES from reward optimization!**

Discovery 2: Self-Verification

Problem: "A store has 15 apples. 8 sold. 12 more arrive. How many?"

Without Verification

"<think>15 - 8 + 12 = 17</think>"

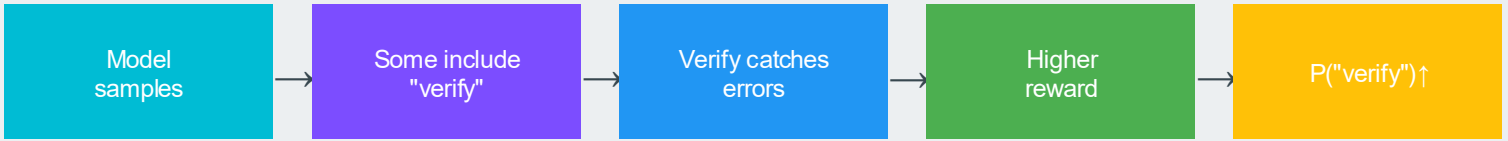
Sometimes: "15 - 8 = 6" (ERROR!)

Error Rate: ~15%

With Verification

"<think>
15 - 8 = 7, 7 + 12 = 19
Verify: 15→7→19 ✓
</think> 19 apples"
Error Rate: ~3%

How Verification Emerges:



Self-verification EMERGES as a learned strategy through reward optimization!

Discovery 3: Backtracking — "Wait, that's wrong"

One of the most FASCINATING emergent behaviors!

Real Example from DeepSeek-R1 Paper:

Problem: $\sqrt{a - \sqrt{(a+x)}} = x$, find sum of solutions

```
<think>
Square both sides... a − √(a + x) = x²
√(a + x) = a - x² → a + x = (a - x²)²
x⁴ - 2ax² - x + (a² - a) = 0
```

"Wait, wait. Wait. That's an aha moment..." ← BACKTRACKING!

Why Backtracking Works:

1. "Wait" signals reconsideration
2. Subsequent tokens correct error
3. Final answer correct → Reward=1
4. RL reinforces "Wait" pattern

Source: DeepSeek R1, 2025

Adaptive Thinking Length

Model learns to calibrate thinking length based on problem difficulty

EASY	MEDIUM	HARD (AIME)
What is 2 + 2?	Solve: 3x + 7 = 22	n^2-19n+99 = perfect sq?
`<think>2+2=4</think>`	`<think>3x=15, x=5, verify...</think>`	`<think>multiple approaches, backtrack...</think>`
~10 tokens	**~50-100 tokens**	**~1000-5000 tokens**

How Length Calibration Emerges:

- Easy + Long thinking → Correct but wasteful (same reward as short)
- Easy + Short thinking → Correct and efficient ✓
- Hard + Short thinking → Often WRONG (reward = 0) ✗
- Hard + Long thinking → More likely CORRECT (reward = 1) ✓

Summary: Emergent Behaviors from RL

"Rather than explicitly teaching the model how to solve a problem, we simply provide it with the right incentives, and it autonomously develops advanced problem-solving strategies."

Simple Reward

1 if correct
0 if wrong

→

Emergent Behaviors

- ✓ Breaking down into steps
- ✓ Alternative approaches
- ✓ Self-verification
- ✓ Adaptive length
- ✓ Backtracking
- ✓ Confidence calibration

KEY INSIGHT

With the right objective (accuracy) and enough exploration (diverse sampling), models DISCOVER effective strategies that humans never explicitly taught them.

What we provide	Simple binary reward + exploration
What model learns	Complex multi-step reasoning strategies
Key mechanism	RL + Group Relative Policy Optimization

TECHNICAL DEEP DIVE

Architecture & Training

- Model architecture
- Training approaches: Pure RL vs Multi-Stage
- GRPO algorithm details
- Reward design & training pipeline

Key Insight: Standard Architecture

IMPORTANT
Reasoning models do NOT have special architecture. They use STANDARD Transformer with SPECIAL TRAINING!

REASONING MODEL = BASE LLM + SPECIAL RL TRAINING

DeepSeek-R1 Specifications:

Total Parameters	671B	**Experts per Token**	8
Active per Token	37B (~5.5%)	**Layers**	61
Architecture	MoE (Mixture of Experts)	**Attention**	MLA (Multi-head Latent)
Experts	256 routed + 1 shared	**Context Length**	128K tokens
Context Length	128K tokens		

Source: DeepSeek R1, 2025

Two Training Approaches

Approach 1: Pure RL
(DeepSeek-R1-Zero)

DeepSeek-V3-Base

↓

GRPO + Rule Reward

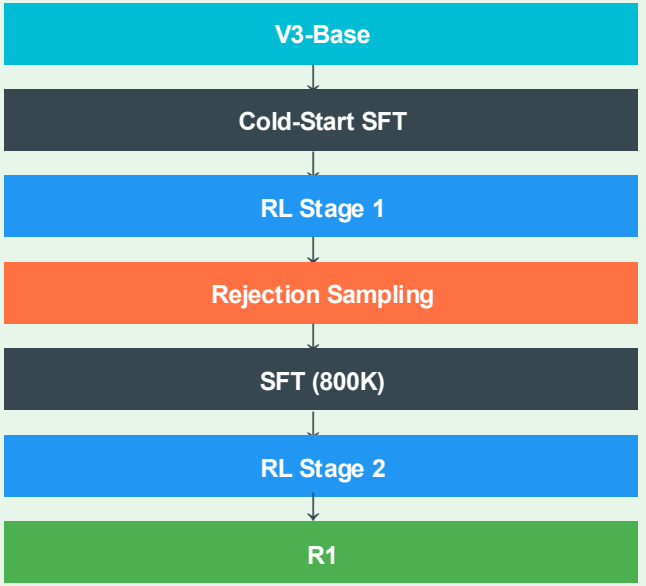
↓

R1-Zero

Issues:

- Poor readability
- Language mixing
- Repetitive patterns

Approach 2: Multi-Stage
(DeepSeek-R1)



Source: DeepSeek R1, 2025

GRPO vs PPO: Training Algorithm

PPO (Traditional)

q → Policy → o

↓

Reward Model → r

↓

Value Model → V(s)

↓

$A = r + \gamma V(s') - V(s)$

Requires: Policy + Value model

~400B parameters total

GRPO (DeepSeek)

q → Policy → $\{o_1 ... oG\}$

↓

Reward → $\{r_1 ... rG\}$

↓

Group Stats: mean, std

↓

$A_i = (r_i - \text{mean}) / \text{std}$

Requires: Policy model ONLY

~50% compute savings!

KEY INSIGHT

GRPO estimates baseline from GROUP comparisons instead of a separate Value model. Same output, same input → compare within group to determine advantage.

Source: DeepSeek R1, 2025

GRPO Objective Function

$$J(\theta) = \mathbb{E}[1/G\ \Sigma_i(\ \min(r_t \cdot A_i,\ clip(r_t, 1-\varepsilon, 1+\varepsilon) \cdot A_i) - \beta \cdot KL\ )]$$

where: $r_t = \pi_\theta(o_i|q) / \pi_{\theta\_old}(o_i|q)$ (probability ratio)

Formula Components:

$\mathbb{E}[...]$	Expectation over questions q and sampled outputs $\{o_i\}$
$1/G\ \Sigma_i$	Average over G samples in the group
$\min(...)$	Take MORE CONSERVATIVE update (stability)
$r_t \cdot A_i$	Standard policy gradient (unclipped)
$clip(r_t, ...)$	Clipped version (prevents large updates)

Key: ε=10 (higher for long outputs), β=0.001, G=16 samples per question

Source: DeepSeek R1, 2025

Reward Design

Rule-Based Rewards

$$\text{Reward} = R_{acc} + R_{format}$$

R_{acc}: 1 if correct, 0 if wrong
- Math: symbolic comparison
- Code: compiler + test cases

R_{format}: check structure
- Has <think>...</think> tags
- Answer in \boxed{}

Language Consistency

$$R_{lang} = \text{Words\_target} \;/\; \text{Words\_total}$$

Model-Based Reward

For general tasks:
DeepSeek-V3 as reward model
Evaluates helpfulness/safety

Combined (RL Stage 2):

$$\text{Reward} = R_{reasoning} + R_{general} + R_{language}$$

Training Data Breakdown:

Math	Code	STEM	Logic	General
26K (Rule)	17K (Compiler)	22K (Rule)	15K (Rule)	66K (Model)

Source: DeepSeek R1, 2025

Complete Training Pipeline

Pre-training	Cold-Start SFT	RL Stage 1	SFT	RL Stage 2
14.8T tokens	~K examples	Rule + Lang	800K samples	Rule + RM

Key Hyperparameters:

Parameter	R1-Zero	RL Stage 1	RL Stage 2
Learning Rate	3e-6	3e-6	3e-6
KL Coefficient (β)	0.001	0.001	varies
GRPO Clip (ε)	-	10	10
Temperature	1.0	1.0	0.7
Samples/Question (G)	16	16	16
Max Output Length	32K→64K	32K	32K

Full training uses ~2.8M GPU hours on NVIDIA H800 clusters

Source: DeepSeek R1, 2025

Distilled Models

Smaller models trained on R1 reasoning outputs

DeepSeek-R1 (671B) → generates reasoning data → **Smaller Models (SFT)**

Model Name	Base Model	Params	Use Case
R1-Distill-Qwen-1.5B	Qwen2.5-Math-1.5B	**1.5B**	Edge/Mobile
R1-Distill-Qwen-7B	Qwen2.5-Math-7B	**7B**	Local
R1-Distill-Qwen-14B	Qwen2.5-14B	**14B**	Balanced
R1-Distill-Qwen-32B	Qwen2.5-32B	**32B**	High Perf
R1-Distill-Llama-8B	Llama-3.1-8B	**8B**	Llama
R1-Distill-Llama-70B	Llama-3.3-70B	**70B**	Maximum

All distilled models have 32K context length

Source: DeepSeek R1, 2025

RUNTIME

Inference & Emergence

- Two-phase generation process
- Test-time compute scaling
- Emergent behaviors taxonomy
- Training dynamics visualization

Two-Phase Generation

User Input: "Solve: If 3x + 7 = 22, find 6x + 3"

PHASE 1: THINKING (inside <think> tags)

```
<think>
I need to solve this step by step.
First, find x: 3x + 7 = 22 → 3x = 15 → x = 5
Now calculate 6x + 3: 6(5) + 3 = 30 + 3 = 33
Verify: 3(5) + 7 = 22 ✓
</think>
```

PHASE 2: ANSWER

The answer is 33.

OpenAI o1: Thinking HIDDEN

DeepSeek R1: Thinking VISIBLE

Autoregressive Generation Process

Generation Flow:

At each step t:

1. Compute hidden state: h_t = Transformer(prompt, tok_1, ..., tok_{t-1})
2. Compute logits: $logits_t$ = LM_Head(h_t)
3. Apply temperature: $logits_t$ = $logits_t$ / T
4. Sample next token: tok_t ~ softmax($logits_t$)
5. Repeat until EOS or max_length

KEY: Model learns WHEN to generate </think> — calibrated to problem difficulty!

Test-Time Compute Scaling

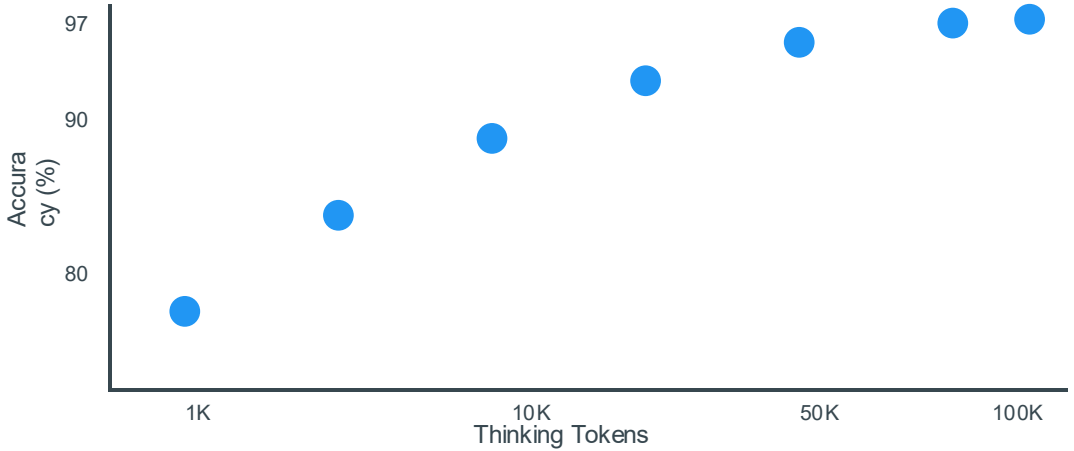
Traditional Scaling (Pre-2024)

More training compute → Better model
(Fixed inference cost)

Test-Time Scaling (2024+)

Same model + More thinking → Better answers
(Variable inference cost)

Accuracy vs Thinking Tokens:



Key Observations:

- More tokens → Higher accuracy
- Diminishing returns ~10-20K
- Model learns to calibrate
- **This EMERGES from RL!**

Source: OpenAI o1 Blog, 2024

Emergent Behaviors Taxonomy

These behaviors are NOT programmed — they EMERGE from RL optimization:

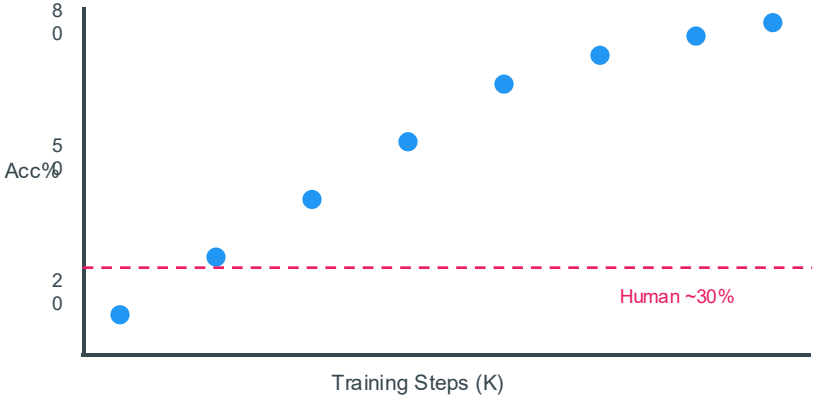
Self-Verification	*"Let me check my answer... ✓"*
Backtracking	*"Wait, that's not right... Let me reconsider..."*
Alternative Exploration	*"One approach is... Another approach is..."*
"Aha!" Moments	*"Oh wait, I see it now! Actually, I realize..."*
Progressive Refinement	*"Initial estimate was X, but after analysis..."*

"The self-evolution of DeepSeek-R1-Zero exemplifies how RL can autonomously enhance a model's reasoning capabilities."

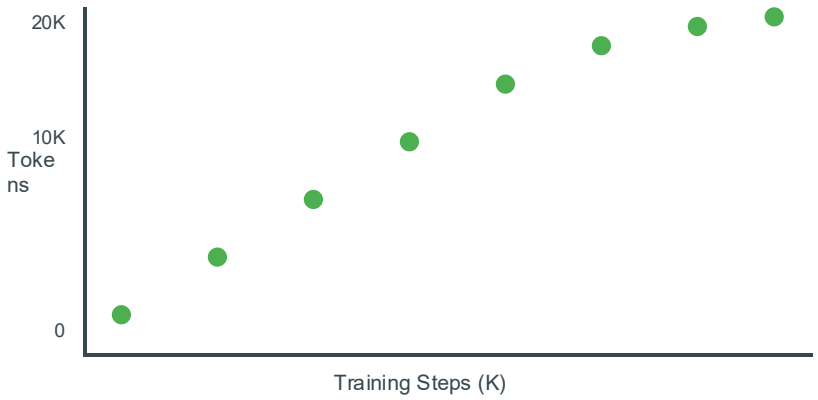
Source: DeepSeek R1, 2025

Training Dynamics of R1-Zero

(a) AIME Accuracy During Training:



(b) Response Length During Training:



Key Observations:

- Pass@1 increases from 15.6% to 77.9% over training
- Response length increases NATURALLY (model learns to think longer)
- Jump at step 8.2K due to max length increase (32K → 64K)
- **Length increase is NOT explicitly rewarded — it EMERGES!**

Source: DeepSeek R1, 2025

Inference Parameters

Parameter	Recommended	Purpose
Temperature	`0.6`	Balance creativity/accuracy
Top-p	`0.95`	Nucleus sampling
Max tokens	`32,768`	Maximum generation length
Presence penalty	`0.0`	Not typically needed
Frequency penalty	`0.0`	Not typically needed

Temperature Effect:

T < 1	T = 1	T > 1
More deterministic	Standard softmax	More random
Inference (consistent)	*Baseline*	*Training (explore)*

EVIDENCE

Benchmarks

- Model comparison on same problem
- Benchmark results analysis

Model Comparison: Same Problem

"A bat and ball cost $1.10. The bat costs $1 more than the ball. How much does the ball cost?"

Traditional LLM	CoT Prompted	Reasoning Model
"The ball costs $0.10"	*"Let me think step by step...* *x + (x+1) = 1.10* *→ x = $0.05"*	*"<think>Set up equation...* *Verify: $0.05+$1.05=$1.10 ✓* *$0.10 would give $1.20 ✗</think>"*
✖ WRONG (intuitive error)	✓ Correct (needs prompting)	✓ Correct + Verify (automatic)

Key Differences:

Traditional LLM	Falls for intuitive error, no reasoning shown
CoT-Prompted LLM	Correct with prompting, reasoning explicit but required prompt
Reasoning Model	AUTOMATICALLY thinks, includes verification, checks intuitive error

Benchmark Results Overview

Benchmark	R1-Zero	R1	o1-mini	o1	GPT-4o
MATH-500	95.9%	**97.3%**	90.0%	94.8%	74.6%
AIME 2024	77.9%	**79.8%**	63.6%	74.4%	9.3%
LiveCodeBench	50.0%	**65.9%**	53.8%	63.4%	33.4%
Codeforces	1444	**2029**	1820	1891	759
MMLU	88.8%	90.8%	85.2%	**92.3%**	87.2%
GPQA Diamond	**75.8%**	71.5%	60.0%	78.3%	53.6%

Key Insights:

- DeepSeek R1 achieves SOTA on math (97.3% MATH-500) and coding (2029 Codeforces)
- Pure RL (R1-Zero) already competitive, multi-stage training adds polish
- Reasoning models dramatically outperform traditional LLMs on complex tasks (AIME: 79.8% vs 9.3%)

Source: DeepSeek R1, 2025

Performance Across Training Stages

Stage	AIME	MATH-500	IF-Eval	ArenaHard
R1-Zero	77.9%	95.9%	46.6%	53.6%
Dev1 (Cold Start)	59.0% ↓	94.2%	71.7% ↑	77.0% ↑
Dev2 (RL Stage 1)	74.0% ↑	95.9%	72.0%	73.2%
Dev3 (SFT 800K)	78.1% ↑	95.4%	78.1% ↑	75.6%
R1 (Final)	**79.8%**	**97.3%**	**83.3%**	**92.3%**

Key Observations:

- Cold Start SFT: Decreases reasoning (AIME ↓) but increases instruction following (IF-Eval ↑)
- RL Stage 1: Recovers reasoning capabilities
- SFT on 800K: Boosts general capabilities and polish
- RL Stage 2: Final refinement → SOTA results

Source: DeepSeek R1, 2025

Distilled Model Performance

Smaller models trained on R1 reasoning outputs

Model	Size	AIME	MATH-500	vs Base
R1-Distill-Qwen-1.5B	1.5B	28.9%	83.9%	**+40%**
R1-Distill-Qwen-7B	7B	55.5%	92.8%	**+25%**
R1-Distill-Qwen-14B	14B	69.7%	93.9%	**+20%**
R1-Distill-Qwen-32B	32B	72.6%	94.3%	**+15%**
R1-Distill-Llama-70B	70B	79.8%	94.5%	**+12%**

Key Insights:

- Even 1.5B distilled model achieves 83.9% on MATH-500 (comparable to GPT-4 level)
- 7B model reaches 55.5% on AIME — far beyond original base model capability
- Distillation successfully transfers reasoning capabilities to smaller models
- Enables deployment on edge devices and local environments

Source: DeepSeek R1, 2025

Limitations & Challenges

Inference Cost

Long thinking = more tokens = higher cost & latency

Language Mixing

R1-Zero outputs mix languages mid-thought (addressed in R1)

Overthinking

May use excessive tokens for simple problems

Readability Issues

Raw RL outputs can be repetitive and hard to read

Safety Concerns

Extended thinking may include undesirable content

Domain Specificity

Best for math/code, less improvement on creative tasks

Mitigations in DeepSeek R1:

- Multi-stage training (cold start + SFT) improves readability
- Language consistency reward reduces mixing
- Model naturally learns to calibrate thinking length to problem difficulty
- Distillation enables cost-effective deployment

Future Directions

Longer Context	Extend from 128K to millions of tokens for complex reasoning
Multi-Modal	Apply reasoning to images, video, and audio
Agent Systems	Combine reasoning with tool use and environment interaction
Efficiency	Reduce inference cost while maintaining quality
Safety	Ensure safe reasoning without harmful intermediate thoughts
Interpretability	Better understand what happens during thinking

The paradigm shift: from training-time scaling to test-time scaling opens new frontiers

Key Takeaways

1	**Thinking = Text Generation**	Reasoning is visible token generation, not hidden internal process
2	**Attention = Working Memory**	Previous tokens become external scratchpad via attention mechanism
3	**Emergence from RL**	Self-verification, backtracking, and exploration emerge from simple rewards
4	**Test-Time Scaling**	More thinking tokens = more compute = better answers (new paradigm)
5	**Standard Architecture**	No special architecture — reasoning comes from training approach
6	**Practical Impact**	SOTA on math (97.3%) and coding (2029 Codeforces rating)

"Rather than explicitly teaching the model how to solve problems, we provide incentives, and it autonomously develops advanced problem-solving strategies."

Source: DeepSeek R1, 2025

References

1. DeepSeek-AI. DeepSeek-R1: Incentivizing Reasoning in LLMs via RL. arXiv:2501.12948, 2025
2. OpenAI. Learning to Reason with LLMs (o1 System Card). September 2024
3. Vaswani et al. Attention Is All You Need. NeurIPS 2017
4. DeepSeek-AI. DeepSeek-V3 Technical Report. December 2024
5. Schulman et al. Proximal Policy Optimization Algorithms. arXiv:1707.06347, 2017
6. Kaplan et al. Scaling Laws for Neural Language Models. arXiv:2001.08361, 2020
7. Wei et al. Chain-of-Thought Prompting. NeurIPS 2022
8. Ouyang et al. Training Language Models with Human Feedback. NeurIPS 2022

Thank You!

Reasoning Models: From Foundations to Frontiers

Part 8 of Large Language Models Course

Key Topics Covered:

- How models "think" through text generation
- Multi-step reasoning emergence from RL
- Architecture & training
- Benchmarks