LLM Reasoning

From Chain-of-Thought to Test-Time Compute

NLP Course – Lecture 3

Advanced Topics in Natural Language Processing

Why Do LLMs Struggle with Reasoning?

The Problem

- LLMs give instant responses
- No "working memory" for computation
- Multi-step problems require planning
- Direct answers often wrong

The Solution

- Chain-of-Thought: think step by step
- Test-time compute: think longer
- Process reward models: verify steps
- DeepSeek-R1: trained to reason

This lecture: Teaching LLMs to think before answering

Reasoning capabilities have dramatically improved since 2022.

Current Agent Limitations

Reliability Issues

- Agents get stuck in loops
- Wrong tool selection
- Hallucinated tool parameters
- Failure to know when to stop

Cost Accumulation

- Each step = API call
- Complex tasks = many calls
- Costs can spiral quickly

Security Concerns

- Tool access = system access
- Prompt injection attacks
- Unintended actions

What Works Today

- Well-defined, bounded tasks
- Human oversight/approval
- Retrieval-heavy workflows
- Single-domain expertise

"Agents are promising but not production-ready for autonomous operation." – 2024 consensus

---

Connection to reasoning: Better reasoning = more reliable agents. This leads us to Act II...

Act II: Reasoning in LLMs

Chain-of-Thought, Test-Time Compute, and DeepSeek-R1

The Surprising Discovery (2022)

The Experiment

Google researchers found something remarkable: Simply adding *"Let's think step by step"* to a prompt improved math accuracy by **40%+**

No model changes. No fine-tuning. Just a prompt.

Why?

- Creates "scratchpad" for computation
- Forces sequential reasoning
- Mirrors human problem-solving

Before CoT

Q: Roger has 5 tennis balls. He buys 2 cans of 3 balls each. How many does he have now?
A: 11 *(direct answer, sometimes wrong)*

After CoT

Q: [same question] *Let's think step by step.*
A: Roger starts with 5 balls.
He buys 2 cans $\times$ 3 balls $=$ 6 balls.
Total $= 5 + 6 =$ 11 balls
(reasoning chain makes answer verifiable)

Wei et al. (2022): "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models"

Chain-of-Thought: The Mathematics

Standard Prompting
Direct answer generation:

$$p(\text{answer}|\text{question})$$

The model jumps straight to the answer in a single forward pass.

Problem
Complex reasoning requires multiple "steps" – but each token is generated independently.

Chain-of-Thought Prompting
Decompose into two stages:

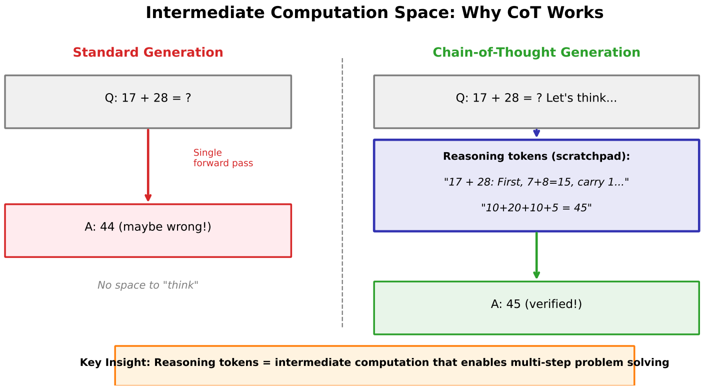
$$p(r|q) \cdot p(a|q, r)$$

Where:

- q = question
- r = reasoning chain
- a = final answer

Key Insight
The reasoning tokens r create intermediate computation space that the model can "use" to solve harder problems.

Connection to Week 9 (Decoding): CoT changes what the model generates, not how it decodes



Key insight: Reasoning tokens create a "scratchpad" that enables multi-step computation within a single generation

Chain-of-Thought Variants

Zero-Shot CoT
Just add: "Let's think step by step"
No examples needed. Works surprisingly well.

Few-Shot CoT
Provide examples with reasoning chains:
Example 1: [problem] [reasoning] [answer]
Example 2: [problem] [reasoning] [answer]
Your turn: [problem]

More reliable but requires good examples.

Self-Consistency
Generate N reasoning chains (with temperature > 0).
Take majority vote on final answer:

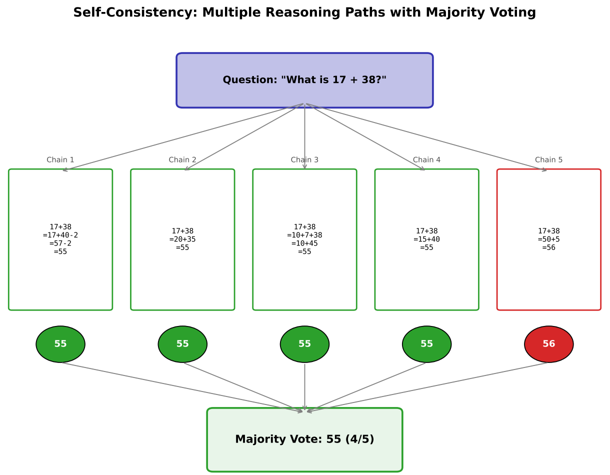
$$\hat{a} = \arg\max_a \sum_{i=1}^{N} \mathbf{1}[a_i = a]$$

Tree of Thoughts
Explore multiple reasoning *paths*, not just one chain.
Allows backtracking on dead ends.

Least-to-Most
Decompose into subproblems first, then solve.

CoT is the most powerful prompt engineering technique discovered so far

Self-Consistency: Sample and Vote



Self-consistency adds 5-10% accuracy on top of CoT by marginalizing over reasoning paths

Chain-of-Thought: Worked Examples

Example 1: Math Problem
Q: A farmer has 17 sheep. All but 9 die. How many are left?
Let's think step by step. "All but 9 die" means 9 survive. Answer: **9 sheep**

Example 2: Logic Puzzle
Q: If all roses are flowers and some flowers fade quickly, can we conclude that some roses fade quickly?
Let's analyze: (1) All roses $\subset$ flowers. (2) Some flowers fade quickly – but which ones? Could be non-rose flowers. (3) We cannot conclude roses fade quickly. Answer: **No, invalid inference**

Example 3: Code Debugging
Q: Why does `sum([1,2,3][:2])` return 3, not 6?
Let's trace: (1) `[1,2,3]` creates list. (2) `[:2]` slices indices 0,1 $\rightarrow$ `[1,2]`. (3) `sum([1,2])` $=$ **3**. The slice excludes index 2.

CoT works across domains: math, logic, code – wherever step-by-step reasoning helps

The Paradigm Shift: Test-Time Compute

Old Paradigm: Scale Training

$$\text{Performance} \propto \log(\text{Parameters})$$

Bigger models = Better performance

GPT-2 $\rightarrow$ GPT-3 $\rightarrow$ GPT-4

Problem
Training cost grows exponentially.
Diminishing returns at scale.
One-size-fits-all computation.

New Paradigm: Scale Inference

$$\text{Performance} \propto \log(\text{Test-Time Compute})$$

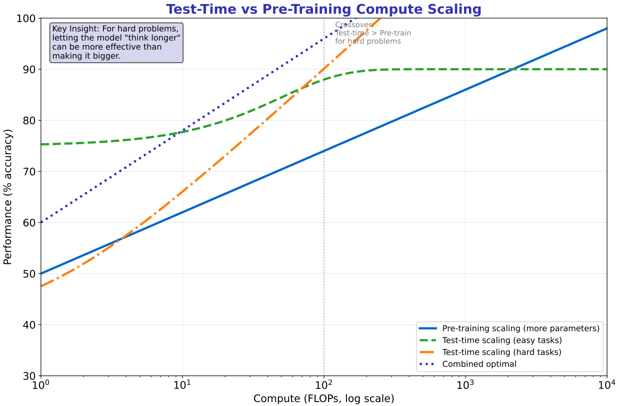
Same model, more "thinking" = Better answers

Key Insight
Not all questions need the same compute.
Hard problems deserve more thinking time.
Let the model allocate compute adaptively.

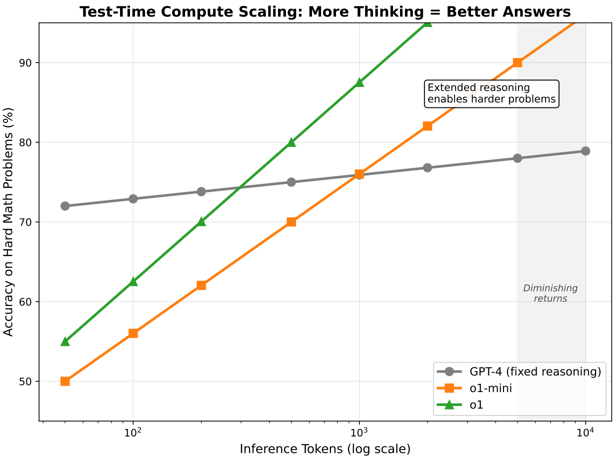
This is revolutionary!

Snell et al. (2024): "Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters"

Test-Time vs Pre-Training Scaling



For hard problems, test-time compute scaling can outperform pre-training scaling at equivalent FLOPs



Key insight: Reasoning models show log-linear improvement with inference tokens; standard models plateau

Two Mechanisms for Test-Time Scaling

1. Best-of-N with Verifiers
Generate N candidate solutions.
Score each with a verifier (PRM).
Select the best one.

$$\hat{y} = \arg\max_{y \in \{y_1, \ldots, y_N\}} r_{\text{PRM}}(y)$$

Process Reward Models (PRMs)
Score *each step* of reasoning:

$$r_{\text{PRM}}(s_1, \ldots, s_T) = \prod_{t=1}^{T} p(\text{correct} | s_1, \ldots, s_t)$$

More compute = more candidates = better selection.

2. Extended Reasoning
Let the model think for more tokens.
Longer reasoning = better answers.
How o1 Does It
Hidden "thinking" tokens before answering.
Model trained to use this space productively.

Adaptive Allocation
Easy questions: Short reasoning
Hard questions: Long reasoning
The model learns to allocate compute based on difficulty.

Both mechanisms: more compute at inference = better results (with diminishing returns)

The Cost-Quality Tradeoff

Model	Tokens Generated	Accuracy	Relative Cost
GPT-4 (direct)	$\sim$50	78%	1x
GPT-4 (CoT prompt)	$\sim$150	89%	3x
o1-mini	$\sim$500	95%	10x
o1	$\sim$2000	97%	40x
o1-pro	$\sim$5000+	99%	100x+

Practical Implication

You can choose your accuracy/cost tradeoff:

- Simple queries: Use fast, cheap model
- Complex reasoning: Invest in more compute
- Critical decisions: Use maximum reasoning

This is like choosing car vs. plane – different tools for different journeys

January 2025: DeepSeek-R1

The Announcement
DeepSeek (Chinese lab) releases R1:

- Matches OpenAI o1 performance
- Fully open-source (weights + paper)
- Fraction of training cost
- Multiple distilled sizes available

Why It Matters
Demonstrated that reasoning can be achieved with:

- Open research
- Smaller budgets
- Novel training approaches

Key Results
AIME 2024 (math olympiad):
15.6% $\rightarrow$ **71.0%** (pass@1)
Matches o1-1217 on most benchmarks.

Available Models
DeepSeek-R1-Distill-Qwen:
1.5B, 7B, 14B, 32B
DeepSeek-R1-Distill-Llama:
8B, 70B
All on HuggingFace, open weights.

DeepSeek-AI (2025): "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning"

DeepSeek-R1-Zero: The Revolutionary Finding

The Experiment
What if we train reasoning with *pure RL*, no supervised fine-tuning?

DeepSeek-R1-Zero

- Start from base model
- Apply RL directly
- Reward only final answer correctness
- No human demonstrations of reasoning

Result
The model *spontaneously* learned to:

- Generate reasoning chains
- Self-verify answers
- Reflect on mistakes
- Allocate more tokens to hard problems

Why This Is Shocking
"Reasoning" emerged from the objective alone.
No one told the model *how* to reason – just rewarded correct answers.

Emergent Behaviors
Self-verification:
"Let me check: $3 \times 5 = 15$, correct."
Reflection:
"Wait, I made an error. Let me reconsider..."
Extended thinking:
Hard problems $\rightarrow$ longer reasoning traces

Implication
Reasoning might be more fundamental than we thought – it emerges when you optimize for correctness.

This suggests reasoning is an "attractor" in the optimization landscape, not a special trick

GRPO: Group Relative Policy Optimization

Standard RL (PPO)
Requires:

- Critic network (value function)
- Reward model
- Complex optimization

GRPO Simplification
No critic network needed!
Compute advantage relative to group:

$$A(x, y) = r(y) - \frac{1}{|G|} \sum_{y' \in G} r(y')$$

For each prompt, generate multiple outputs, compare to each other.

Rule-Based Rewards
No neural reward model either!
Accuracy reward:

$$r_{\text{acc}} = \mathbf{1}[\text{answer correct}]$$

Format reward:

$$r_{\text{fmt}} = \mathbf{1}[\text{¡think¿...¡/think¿ tags present}]$$

Why This Works
For math/code: correctness is verifiable.
No need to learn "what humans prefer."

GRPO: Simpler than PPO, no reward model, no critic – yet achieves state-of-the-art reasoning

DeepSeek-R1 Full Pipeline

Stage 1: Cold Start (Optional)
Small amount of SFT on reasoning examples.
Teaches the format: <think>...</think>
Not strictly necessary (R1-Zero skips this).

Stage 2: Reasoning RL
Pure RL with GRPO.
Reward: correctness + format.
Model learns to reason.

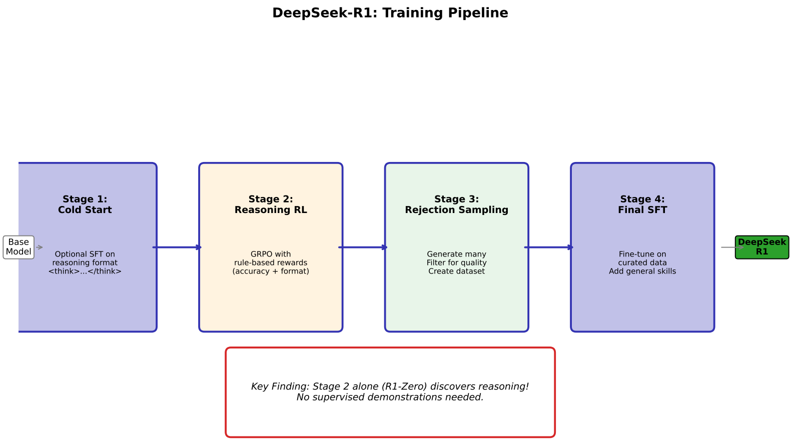
Stage 3: Rejection Sampling
Generate many responses from RL model.
Filter for correct + well-formatted.
Creates high-quality reasoning dataset.

Stage 4: Final SFT
Fine-tune on curated reasoning data.
Adds general capabilities back.
Balances reasoning with helpfulness.

Result: DeepSeek-R1

Key insight: RL discovers reasoning, then SFT polishes and generalizes it

DeepSeek-R1: The Four-Stage Training Pipeline



DeepSeek-R1 shows that open-source reasoning can match proprietary models with clever training

o1 vs DeepSeek-R1: What We Know

OpenAI o1

- Closed source, proprietary
- Hidden "thinking" tokens (not shown to user)
- Likely uses process supervision
- Rumored to use search/planning
- Available via API only

Strengths
Polish, reliability, integration with OpenAI ecosystem.

DeepSeek-R1

- Open source (weights + paper)
- Visible reasoning traces
- Pure RL approach documented
- Distilled to many sizes
- Run locally or via API

Strengths
Transparency, customizability, research value.

Performance
Comparable on most benchmarks.

The gap between closed and open reasoning models is narrowing rapidly

Key Takeaways: LLM Reasoning

1. **Chain-of-Thought** dramatically improves reasoning (+40% on math)
2. **Intermediate tokens** serve as computational scratchpad
3. **Test-time compute** is the new scaling paradigm
4. **DeepSeek-R1** showed pure RL can develop reasoning
5. **Process reward models** enable verification of reasoning steps

Key Insight: "Let the model think longer" is often more effective than making models bigger.

Reasoning capabilities define the frontier of AI capabilities in 2025.

Further Reading: LLM Reasoning

Foundational Papers:

- Wei et al. (2022) - "Chain-of-Thought Prompting"
- Wang et al. (2023) - "Self-Consistency"
- DeepSeek (2025) - "DeepSeek-R1"
- OpenAI (2024) - "o1 System Card"

Key Concepts:

- Test-time compute scaling
- Process Reward Models (PRMs)
- GRPO (Group Relative Policy Optimization)

Repository: github.com/Digital-AI-Finance/Natural-Language-Processing