Frontier — Agents · Reasoning · Interpretability

Lecture 24 · ES 667: Deep Learning

Prof. Nipun Batra
IIT Gandhinagar · Aug 2026

Learning outcomes

By the end of this lecture you will be able to:

1. Describe **agents** as the perceive-think-act loop with external tools.
2. Understand **test-time compute** scaling in reasoning models (o1, Claude).
3. Read the **residual stream view** of a Transformer.
4. Recognize **induction heads** and sparse-autoencoder features.
5. Articulate the **open problems** of DL (continual learning, data efficiency, grounding).
6. Identify **2026-2030 research directions** worth pursuing.

The final lecture

23 lectures of theory + practice. Today — three threads that define the 2024–2026 frontier, plus a course recap.

> REFERENCE
>
> Today's reading is a collection of blogs and papers · Yao 2022 (ReAct), Wei 2022 (CoT), Anthropic interp blog, OpenAI o1 blog.

Four questions:

1. What are **agents** and what changed in 2024?
2. What are **reasoning models** (o1, Claude thinking)?
3. What is **mechanistic interpretability**?
4. What are the open problems — and where does this course take you next?

PART 1

Agents & tool use

2024's big shift

Brain in a jar · why agents matter

KEY IDEA

Think of a standard LLM as a **brilliant brain in a jar**. It can talk and write, but it can't *do* anything in the world.

Agents are the revolution that gives this brain **hands and eyes** · web browsers, code editors, file systems, calculators, APIs.

The LLM keeps its strengths (knowledge, language fluency) and gains the missing piece · **action**. With tools, the same model that can describe how to book a flight can now actually book one.

From chatbot to agent

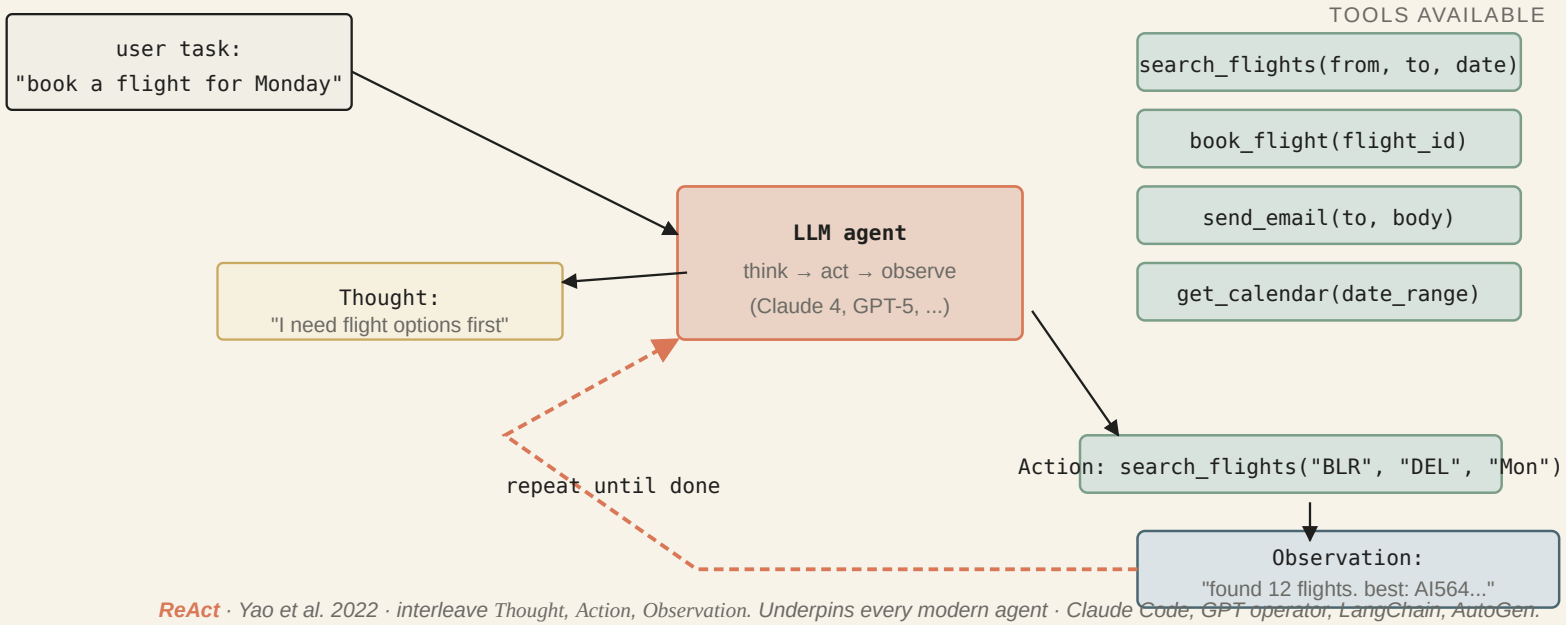
An LLM chatbot gives you text. An LLM **agent** can *act* — call tools, browse the web, write code, click buttons on a screen.

KEY IDEA

The difference: **closing the perceive-think-act loop** with external tools. The LLM becomes the reasoning layer of a larger system.

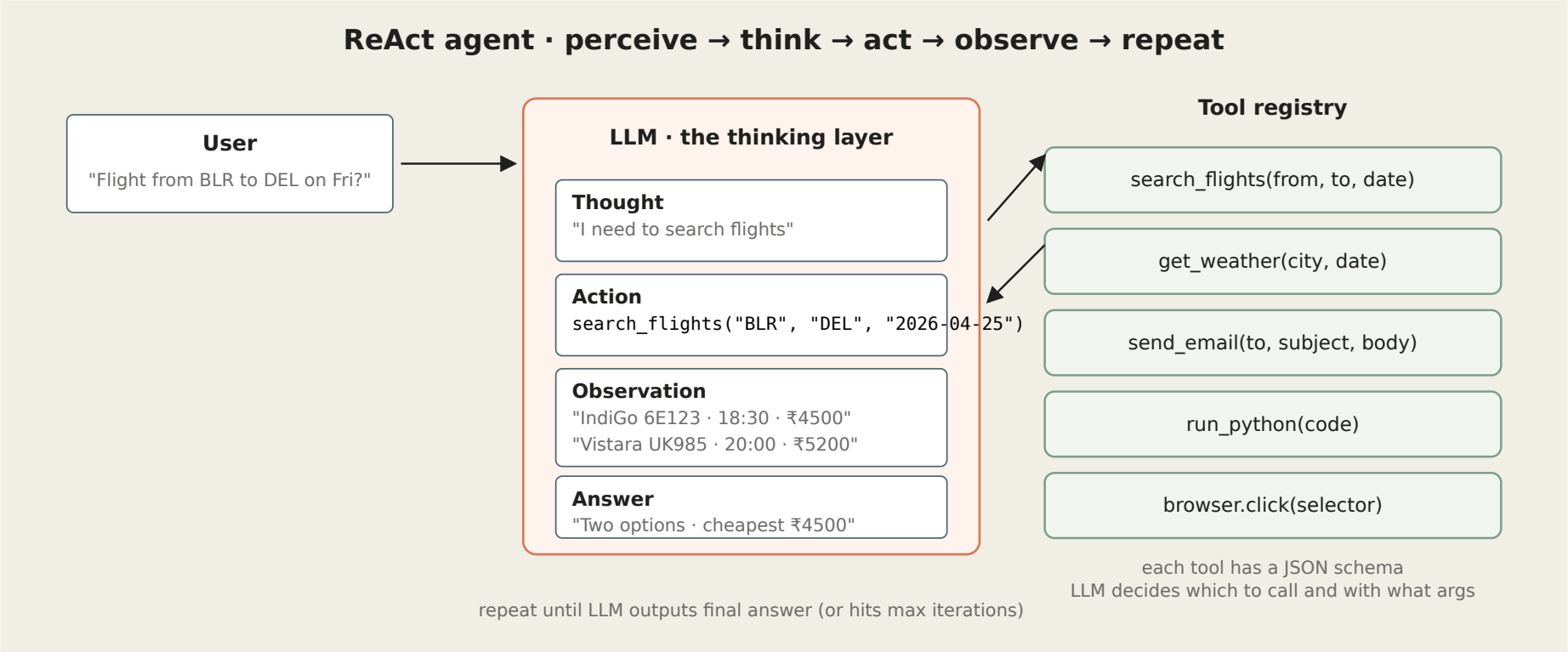
The ReAct loop

ReAct loop — LLM thinks, acts, observes, repeats until done



***ReAct** · Yao et al. 2022 · interleave Thought, Action, Observation. Underpins every modern agent · Claude Code, GPT operator, LangChain, AutoGen.*

ReAct · annotated with tools



Function calling · how agents work

Modern APIs (Claude, OpenAI, Gemini) support **structured tool calling**:

```
tools = [
    {
        "name": "search_flights",
        "description": "Find flights between two cities on a date",
        "input_schema": {
            "type": "object",
            "properties": {
                "from": {"type": "string"},
                "to":   {"type": "string"},
                "date": {"type": "string", "format": "date"}
            }
        }
    }
]
response = client.messages.create(model="claude-4", tools=tools, messages=[...])
```

The model returns a structured call · you execute the function · feed back the result · loop until done.

Computer use · agents controlling UIs

Claude Computer Use (2024) · the model sees a **screenshot**, outputs **mouse + keyboard actions**.

```
Screenshot → Claude → "move_mouse(320, 450) ; click() ; type('hello')" → screenshot
```

This is RL-like but trained mostly supervised on human demonstrations and synthetic examples. Opens up:

- Browser automation
- Desktop task completion
- Form filling
- Legacy-app bridging

> IN PRACTICE
>
> This course itself was built largely by Claude Code — an agent loop over bash / edit / read tools. Agents are the **application layer** of 2026 AI.

PART 2

Reasoning models

Test-time compute as a new axis

Chain-of-thought · the 2022 discovery

> REFERENCE
>
> Wei et al. 2022 · prompting an LLM to "think step by step" dramatically improves multi-step reasoning at scale.

```
Q: John has 5 apples. He gives 2 to Mary and buys 4 more. How many?
A: He starts with 5. Gives 2 → 3. Buys 4 → 7. Final: 7.
```

CoT emerges at scale (~60B params). Below that, adding "think step by step" doesn't help.

Reasoning models · outcome vs process rewards

INTUITION

Teaching a student to solve a math problem.

- **Outcome reward** · *"the final answer is 7"*. Easy to check; no info on **how** to get there.
- **Process reward** · *"this step is correct, but that step has a mistake."* Much richer signal.

Same idea for LLMs. 2024's big idea · don't just prompt for CoT — **train** for it with RL using both signals.

Reasoning · the training recipe

o1 (OpenAI 2024), **Claude extended thinking**, **DeepSeek R1** (2025):

1. **Generate** many candidate chains of thought for a hard question.
2. **Reward** each chain · outcome (final answer correct?) and process (each step logical?).
3. **Fine-tune via RL** to make high-reward chains more likely.
4. **At inference** · spend 10×–100× more compute, let the model "think."

Worked example · rewarding chains of thought

Question. A bat and a ball cost $1.10 total. The bat costs $1.00 *more than* the ball. How much does the ball cost?

Candidate 1 (greedy).

1. Bat is $1.00. 2. Ball = $1.10 - 1.00 = \$0.10$. **Answer · 10¢.**

- Outcome reward · **0** (wrong — common-sense trap).
- Process reward · low (step 2 based on a wrong assumption in step 1).

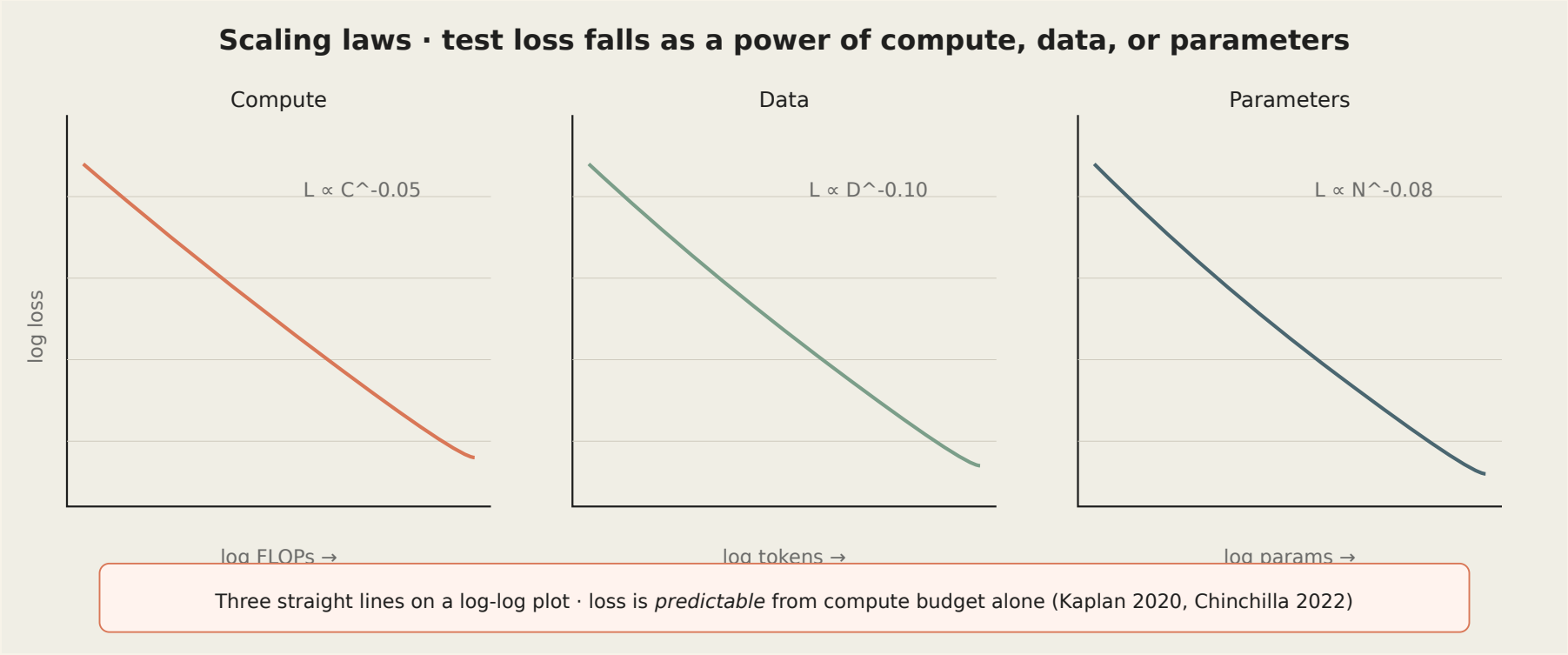
Candidate 2 (algebra).

1. $B + L = 1.10$, $B = L + 1$. → $2L + 1 = 1.10$. → $L = 0.05$. **Answer · 5¢.**

- Outcome reward · **+1**.
- Process reward · **high** — every step logical.

The model is trained to make traces **like Candidate 2** more likely.

The scaling laws recap · in one figure



Three straight lines on a log-log plot · loss is *predictable* from compute budget alone (Kaplan 2020, Chinchilla 2022)

Test-time compute · the college-exam analogy

KEY IDEA

Compare model training to going to college.

Training compute is the years spent in college learning general knowledge.

After college, on a hard exam question, you don't just answer instantly. You **pause, sketch on scratch paper, double-check**. That's **test-time compute**.

Reasoning models (o1, Claude thinking) are exactly this · same base capability as a "college graduate" model · but allowed to use scratch paper before responding. The scratch paper is internal chain-of-thought that the user never sees.

A new scaling axis

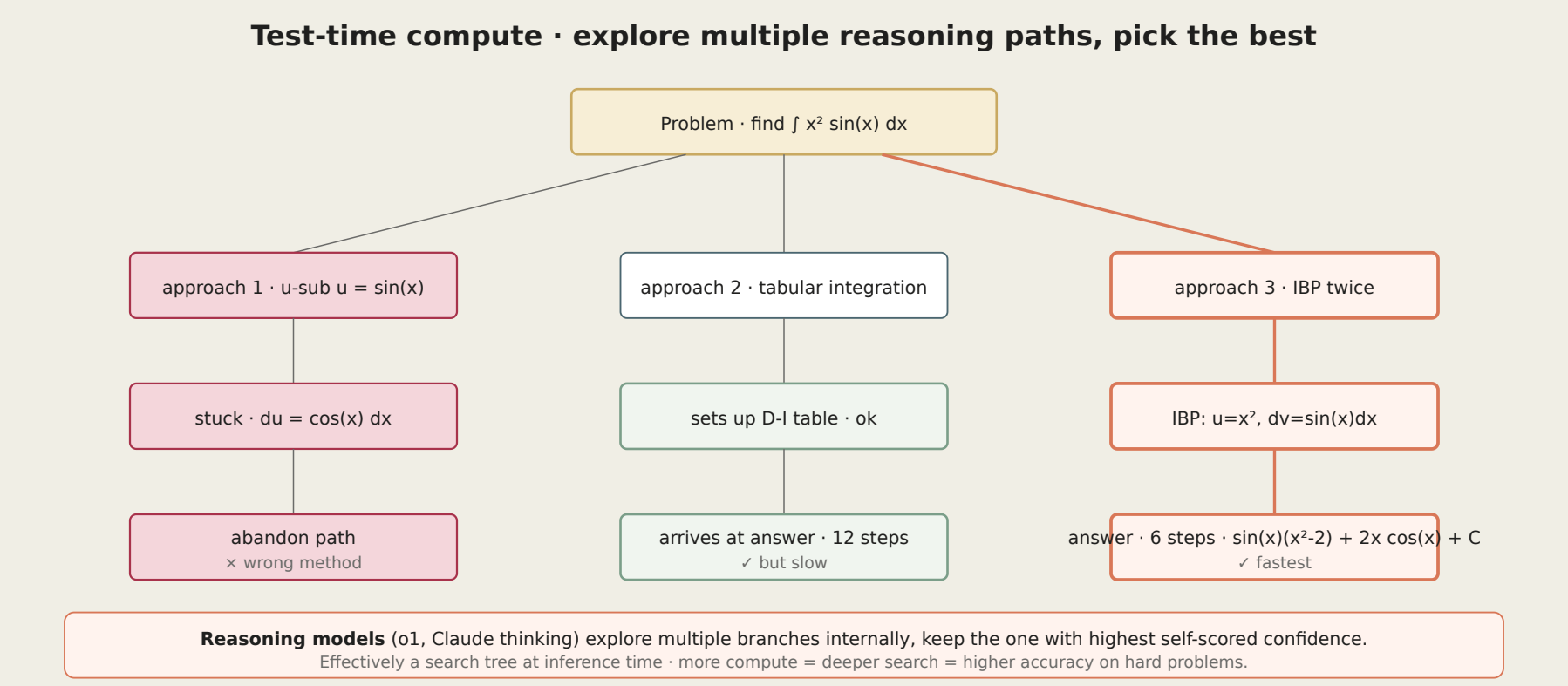
Until 2024, the only scaling axis was **training compute**. In 2024+ we added **test-time compute**.

KEY IDEA

Two knobs now: (a) spend more on pretraining → better general capability; (b) spend more on per-query reasoning → better on hard problems.

Both pay off. OpenAI reported that o1's performance on math benchmarks scales smoothly with inference-time compute budget.

Reasoning · tree search at inference



Reasoning models · benchmarks

MODEL	AIME 2024 (MATH)	CODEFORCES
GPT-4	12%	~800 Elo
o1	74%	~1800 Elo
o3	97%	~2700 Elo (grandmaster)

Comparable gains on HumanEval, MATH, GPQA. This is an entirely new capability curve.

IN PRACTICE

Practical rule · if the problem needs multi-step reasoning, use a reasoning model. If it needs fast response or is simple, use a regular model.

PART 3

Mechanistic interpretability

What is the model actually doing?

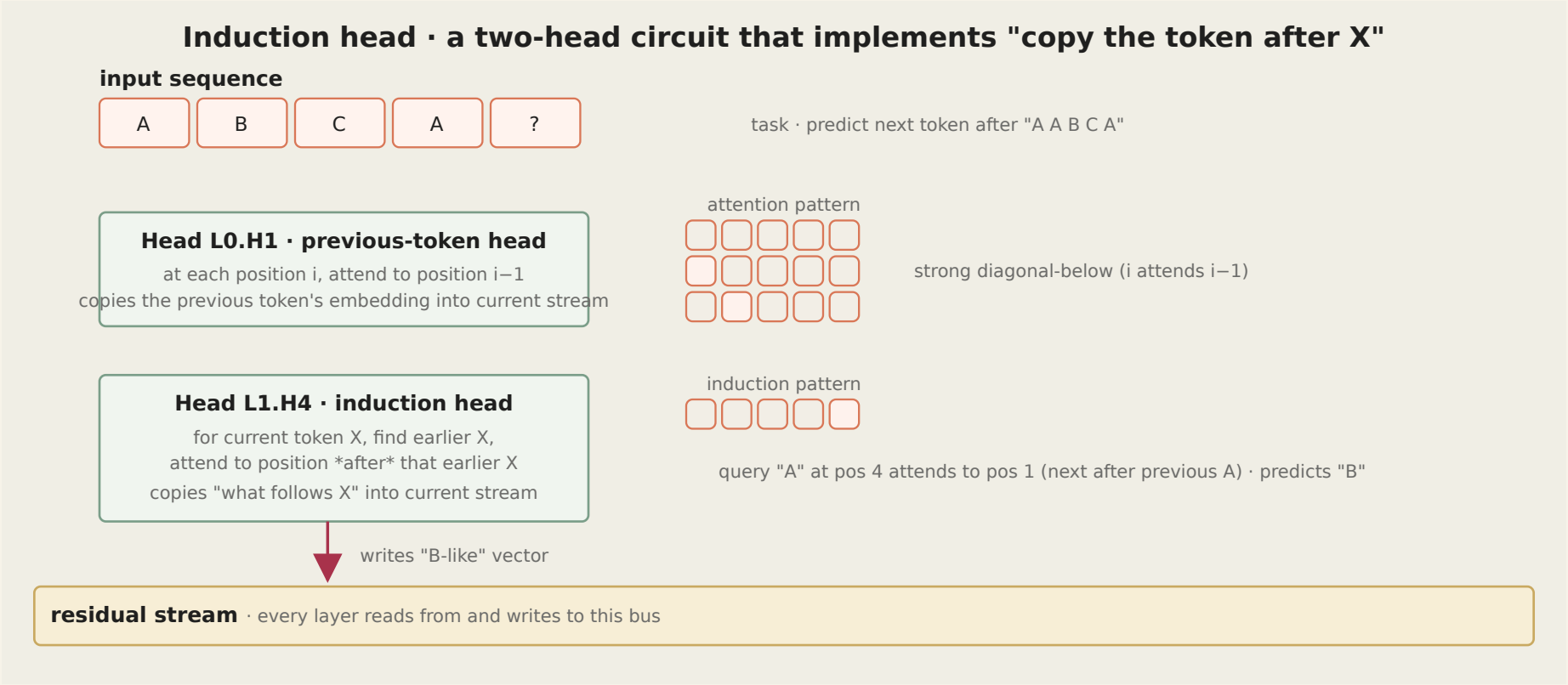
The problem

A 70B Llama has 70 billion parameters. We trained it, and it works. But if it produces a wrong answer — or a dangerous one — **we can't read the weights and know why**.

Mechanistic interpretability (**mech interp**) tries to reverse-engineer specific computations inside trained networks.

> REFERENCE
>
> Anthropic's interpretability team · Olah et al. circuits research 2020+, sparse autoencoders 2023+, dictionary learning 2024+.

Induction head · a discovered circuit



Residual stream · the shared-whiteboard analogy

INTUITION

A Transformer layer = experts working on a shared **whiteboard**.

1. Initial document (input) is on the board → the **residual stream**.
2. Attention expert reads, writes a sticky note ("this word relates to that word"), **adds** it to the board (doesn't erase).
3. FFN expert reads board + sticky, writes another note, adds it.
4. Updated board → input to next layer.

The Transformer equation just says: **read from the bus → add a contribution → write back.**

Residual stream · worked numeric

Toy model, $d = 4$. Token enters layer 5 with state:
$h_5 = [0.2, 0.5, 0.1, -0.9]$

Step 1 · attention update. Attention reads h_5, looks at other tokens, returns:
$\text{attn} = [0.0, 0.3, 0.4, 0.0]$
(boost features 2 and 3 from related tokens)

Step 2 · FFN update. FFN processes h_5 alone, returns:
$\text{ffn} = [0.1, -0.1, 0.0, 0.2]$
(slightly boost feature 1, decrease feature 2, clarify feature 4)

Step 3 · sum (the residual update).
$h_6 = h_5 + \text{attn} + \text{ffn}$
$h_6 = [0.2 + 0.0 + 0.1,\ 0.5 + 0.3 - 0.1,\ 0.1 + 0.4 + 0.0,\ \ -0.9 + 0.0 + 0.2]$
$h_6 = \mathbf{[0.3,\ 0.7,\ 0.5,\ \ -0.7]}$

Same equation: $h_{l+1} = h_l + \text{attn}_l(h_l) + \text{ffn}_l(h_l)$. The "bus" perspective makes circuits findable — induction heads, IOI task, etc.

Sparse autoencoders · the dictionary analogy

KEY IDEA

Inside a model, a single neuron must represent many ideas at once · "bank" means river bank AND financial bank, plus context-dependent shades. This is **superposition** · confusing for analysis.

A sparse autoencoder forces the model to use a **giant explicit dictionary**. Instead of one ambiguous "bank" neuron, distinct **river_bank** and **financial_institution** features fire.

The dictionary is much wider than the residual stream (e.g., 100k features for a 12k-dim residual). Most features are 0 for any input · the *active* ones become human-interpretable concepts.

Sparse autoencoders · the inverted bottleneck

INTUITION

A normal autoencoder · narrow bottleneck. *"Compress 100 words to 5."*

A **sparse autoencoder (SAE)** · *inverted* bottleneck. *"Have a 100,000-word dictionary, but only allowed to use 5 of them."* You must pick **extremely precise** words.

SAE · how it disentangles superposition

Problem · superposition. A single neuron in the residual stream might fire for *"Golden Gate Bridge"*, *"the colour red"*, AND *"Python syntax errors"*. Confusing.

SAE recipe:

1. **Input** · residual-stream vector x (e.g. $d = 12{,}288$).
2. **Encoder** · expand to a much wider sparse vector f ($d_{\text{sae}} = 100{,}000$). $f = \text{ReLU}(W_{\text{enc}} x + b)$, with a **sparsity penalty** forcing most entries to 0 (typically only 10–50 non-zero per input).
3. **Decoder** · reconstruct $x' = W_{\text{dec}} f + b'$.
4. **Train** to minimize $\|x - x'\|^2$ + sparsity penalty.

Worked numeric (toy). $x = [0.9, 0.8, -0.7, 0.1]$ (meaningless — superposition). Trained encoder maps to: $f = [0, 0, 0, 0, 0, \mathbf{0.95}, 0, 0, 0, 0]$

Researchers find **all inputs** that activate feature 6 → all about the **Golden Gate Bridge**. Label: *"Feature 6 = Golden Gate Bridge"*.

Anthropic 2024 · SAE on Claude 3 Sonnet → millions of human-readable features. Clamp features → control behavior ("Golden Gate Claude" demo).

Why mech interp matters

Two reasons:

1. **Safety** · identify and edit circuits that produce harmful outputs.
2. **Science** · understand what representations language models actually learn, and how.

> INTUITION
>
> Still a young field. Most interp results are about small models or narrow circuits. Scaling interp to frontier-level models is a 2026+ research agenda.

PART 4

Open problems

What the next decade of DL research looks like

Safety · why alignment matters

As models become more capable, the cost of misalignment grows:

DERIVATION

2018	2024	2030 (?)
Misclassify image	Give wrong factual answer	Autonomously execute bad plan
Cost: annoy user	Cost: spread misinformation	Cost: catastrophic

KEY IDEA

Claude, GPT, Gemini all ship with elaborate safety stacks · constitutional AI, RL from safety feedback, red-teaming, classifier filters, refusal training. Safety is not a layer; it's the product.

What you've learned · a final recap

DERIVATION

MODULE	COVERED
Foundations (L1-L2)	why DL, UAT, depth vs width, residuals
Training craft (L3-L6)	recipe, SGD / Adam, schedules, regularization
Vision (L7-L9)	CNN mechanics, ResNet family, detection, SAM
Sequences → Transformers (L10-L14)	RNN/LSTM/GRU, Seq2Seq, attention, Transformer, tokenization
LLMs (L15-L16)	scaling laws, RoPE, GQA, LoRA, RLHF, DPO
Self-supervision + VLMs (L17-L18)	SimCLR, MAE, CLIP, LLaVA
Generative (L19-L22)	VAE, GAN, DDPM, CFG, latent diffusion
Systems + frontier (L23-L24)	KV-cache, quantization, agents, reasoning, interp

Open problems · the short list

1. **Reasoning reliability** · even o3 still hallucinates; trust calibration is unsolved.
2. **Continual learning** · models can't easily update with new facts without retraining.
3. **Data efficiency** · humans learn from 10^3 examples; models need 10^{12}. Why?
4. **Alignment** · how do we ensure advanced systems remain beneficial?
5. **Interpretability at scale** · we barely understand what's inside a 70B model.
6. **Multi-modal reasoning** · image+text+video+action reasoning jointly is still weak.
7. **Grounding** · LLMs know about the world through text; they don't have embodied experience.
8. **Energy & cost** · a GPT-4 query costs ~pennies but society-scale deployment is energy-intensive.

Each is a PhD worth of work. Pick one.

Where's the field going?

Predictions (take with salt):

- **Reasoning compute** will scale 10× per year for a few years — expect ~1000× by 2028.
- **Agentic AI** will move from demos to production workflows in 2026.
- **Multimodal** will absorb audio, video, 3D into the same models.
- **Open-weight models** will continue closing the frontier gap (Llama, Mistral, DeepSeek).
- **Domain-specific** small models (medical, legal, scientific) will win niche deployments.
- **Safety & alignment** research will become mainstream.

PART 5

Course recap

What you learned

The 24-lecture arc

MODULE	LECTURES	BIG IDEAS
1 Foundations	L1–L3	MLP, ResNets, training recipe
2 Optimization	L4–L5	SGD, momentum, Adam, schedules
3 Regularization	L6	double descent, augmentation, norm, dropout
4 CNNs	L7–L9	architecture evolution, detection, SAM
5 Sequences	L10–L11	LSTM, Seq2Seq, bottleneck
6 Transformers	L12–L14	attention, nanoGPT, tokenization
7 LLMs	L15–L16	scaling laws, LoRA, RLHF, DPO
8 SSL + VLM	L17–L18	SimCLR, CLIP, LLaVA
9 Generative	L19–L22	VAE, GAN, DDPM, Stable Diffusion
10 Frontier	L23–L24	inference, agents, interp

What you can now do

1. **Read any 2026 ML paper** and understand the architecture.
2. **Implement a Transformer, a diffusion model, a LoRA fine-tune** from scratch.
3. **Know when to use what** — CNN vs ViT, SGD vs AdamW, RLHF vs DPO.
4. **Debug training failures** using the ladder and error analysis.
5. **Estimate compute + memory** for any training or inference setup.
6. **Build an agent** that uses tools to accomplish tasks.

This is the current skill floor for a DL engineer or research student.

What to read next

- **Bishop & Bishop** · *Deep Learning: Foundations and Concepts* — for more mathematical rigor.
- **Karpathy's Zero to Hero** — keep going beyond what we covered.
- **Prince's UDL** — revisit the chapters you skimmed.
- **Recent papers** — set up arXiv alerts for your interest area.
- **Blog posts** · Lil'Log (Lilian Weng), Simon Willison, Chip Huyen, Anthropic engineering blog.

What to do next

- **Replicate a paper** · picking one from NeurIPS / ICML / ICLR 2025 and implementing it end-to-end teaches you more than any course.
- **Contribute to open source** · HuggingFace, vLLM, PyTorch. Start with issues labeled "good first issue."
- **Build a small project** · fine-tune an LLM on your domain; train a tiny diffusion model on a toy dataset; ship an agent.
- **Read safety & alignment work** · if you care about where this technology is going.

Summary · Lecture 24 — summary

- **Agents** · LLM + tools in a ReAct loop. 2024's biggest application-layer shift.
- **Reasoning models** · train for long chains of thought with RL; spend more test-time compute.
- **Mech interp** · residual-stream view + sparse autoencoders; progress but still early.
- **Open problems** · reliability, continual learning, alignment, interpretability, grounding.
- **What you have now** · end-to-end understanding of every major DL system in 2026.

Thank you.

ES 667 · Deep Learning · Aug 2026

Prof. Nipun Batra
IIT Gandhinagar

Go build something.

NOTEBOOK

Final project · apply a technique from any lecture to a real problem · 3-week timeline · pitch week after endsem.