



Hochschule Aalen
UNIVERSITY OF APPLIED SCIENCES

Lecture 6

STEERING THE DISTRIBUTION

Two questions, two blocks

Block A — Why the mechanism fails

What the model actually is after pre- and post-training, and the ways it fails: hallucination, CLAUDE.md compliance drift, lost-in-the-middle. Every failure named in last week's vocabulary — never called a bug.

§2–8 · THE DIAGNOSIS

Block B — Engineering the distribution

The deliberate controls: temperature, in-context learning, few-shot, chain-of-thought, self-consistency, role priming. Real theory, with citations.

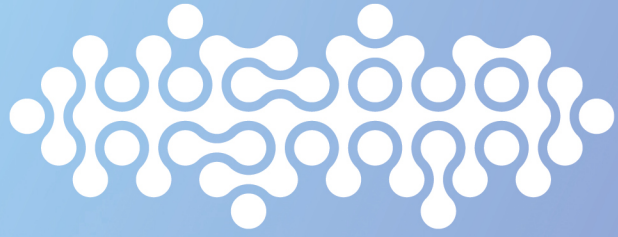
§9–16 · THE CONTROLS

THE DELIVERABLE

Five named patterns — **zero-shot** · **few-shot** · **chain-of-thought** · **self-consistency** · **role priming** — each a deliberate use of a Lecture 5 mechanism.

Every prompting pattern in this lecture — zero-shot, few-shot, chain-of-thought, self-consistency, role priming — is a **deliberate, named use** of the mechanism we built last week.

there are no new ingredients · no folklore — a mechanism and a citation, every time



Hochschule Aalen
UNIVERSITY OF APPLIED SCIENCES

Why the Mechanism Fails

A

One training objective

PRE-TRAINING, IN ONE SENTENCE

For every position in a colossal corpus, predict the next token; compare to the truth; back-propagate the cross-entropy loss. Repeat trillions of times.

The result, after all that compute, is a model with **one operational behaviour**: given a sequence, emit a probability distribution over the next token. Nothing else.

THE LOAD-BEARING FACT

The model was **never trained to be correct** — only to predict the most likely continuation. Everything in Block A follows from this.

What the model does NOT have

No truth or correctness

It saw correct and incorrect text alike; nothing flags the wrong answer as wrong.

→ §4 HALLUCINATION

No goals or intent

It is not "trying to help." It emits a distribution; helpful-looking text was just abundant.

→ RLHF SHAPES, NOT COMMANDS

No external grounding

It cannot check an endpoint, run code, or read the docs.

→ FACTUAL HALLUCINATION

No memory across chats

Each forward pass sees only its current context window.

→ CLAUDE.MD IS CONTEXT, NOT CONFIG

READ EVERY FAILURE THIS WAY

Each "does not have" maps to a failure mode later in Block A. The model is a pattern-completion engine — spectacular where patterns are dense, predictably wrong where they are rare.

The base model: two continuations

Base model – end of pre-training, no post-training, no preference

prompt "What is the capital of France?"

continuation 1 "Paris."

Wikipedia, textbooks, educational sites

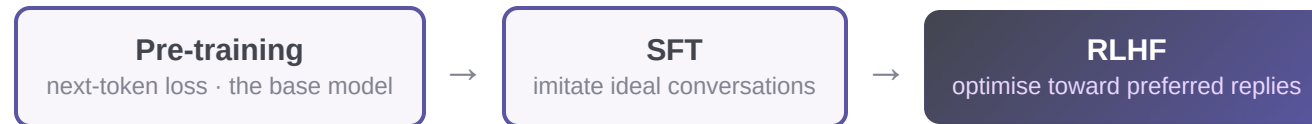
continuation 2 "What is the capital of Germany? What is the capital of Italy?"

quiz sheets, trivia lists – also abundant in the corpus

NO PREFERENCE

Both continuations have non-trivial probability. The base model is not choosing "helpful" over "unhelpful" — it is sampling a distribution that contains both. **SFT + RLHF is what makes the answer win.**

SFT → RLHF: the post-training pipeline



SFT — teaches the format

Train on curated (prompt, ideal-response) conversations with the *same* next-token loss. The model learns what an assistant *looks* like. Vicuna, Alpaca, LLaMA-Instruct stop here.

RLHF — teaches the preference

A reward model learns human rankings; the policy is optimised to score higher. The model learns which assistant reply humans *prefer*. PPO → DPO / RLAIF / Constitutional AI.

What post-training buys (and doesn't)

What post-training buys	What it does not buy
Instruction following — the distribution peaks on tokens that obey	Truth — it aligns to <i>preferred forms</i> , not to facts
A helpfulness <i>style</i> — confident, structured, caveated	Elimination of hallucination — it gilds it, not removes it
The [system] > [user] role hierarchy (a learned bias)	Reliable refusal — learned, not enforced; jailbreakable
Refusal of harmful / off-topic requests	New factual knowledge — that still comes from pre-training

RLHF does not teach facts

MISCONCEPTION, CORRECTED

RLHF shifts **style and behaviour**, not knowledge. The classical reward model scores the *form* of an answer — authoritative, well-structured, caveated — not its truth.

THE CONSEQUENCE

A post-trained model **hallucinates more politely, not less often**. The confident gloss that rankers preferred sits on top of the same base distribution — still no ground-truth check, so a falsehood can come out as a sharp, confident peak.

DeepSeek-R1 and the GRPO frontier

THE FRONTIER HAS MOVED

The classical recipe rewards *human preferences over response forms*. For reasoning tasks, the reward can be replaced by something stronger: **automatic verification of the answer**.

DEEPSEEK-R1 (GUO ET AL., 2025)

Pure RL against a correctness signal elicited complex reasoning — and the SFT stage was skipped entirely in their **R1-Zero** variant. A new RL algorithm, **GRPO**, replaced PPO. We return to this in chapter 13.

REGISTER THE SHIFT

Post-training is no longer a single canonical recipe but a fast-moving family of techniques. The principles of §3 still hold.

Hallucination is **not a bug**. It is the expected behaviour of next-token prediction when the context is insufficient to peak the distribution onto a correct answer.

the central diagnostic claim of Block A — §4

Peaked vs flat: the shape of the distribution

$P(\text{next} \mid \text{"the capital of France is ..."})$

PEAKED



One token (Paris) captures almost all the mass.

$P(\text{next} \mid \text{"the Stack Exchange API endpoint is ..."})$

FLAT



Many tokens share the mass — no clear winner.

Same softmax. Same forward pass. Only the prompt's specificity decides the distribution's shape — and the model always picks a token, even from the flat one.

Factual / structural / confident-nonsense

Factual

A specific fact, stated confidently, and wrong.
"The Stack Exchange endpoint is
`/api/v2/questions/monthly`." It does not
exist.

CLUSTERS ON PROPER NOUNS, VERSIONS, APIS

Structural

Code shaped correctly, but the method / signature
/ import does not exist. *locust*
`get_monthly_report()` .

CAUGHT BY THE RUNTIME, NOT BY READING

Confident-nonsense

A long, fluent, contextually-perfect explanation
that is end-to-end invented. *requests*
`DEFAULT_POOL_SIZE` .

NEEDS DOMAIN EXPERTISE TO DETECT

ONE MECHANISM, THREE FACES

All three are pattern-completion artefacts with no ground-truth check (flat OR confidently wrong). Telling them apart tells you *how* to engineer against each.

Per-category examples and mitigations

Category	Example (Act 1)	Root cause (L5)	Mitigation
Factual	invented API endpoint	no ground-truth check on a specific fact	retrieval — docs in context
Structural	hallucinated function signature	pattern shape $\neq$ actual library surface	grounding — API surface, type stubs
Confident-nonsense	fabricated requests internals	confidently-wrong sharp peak + RLHF style	verification — run it, test it, ask an expert

THE MOVE

"Better prompt" is rarely enough alone — each category needs a different layer. This foreshadows the three-layer model.

Checkpoint: the diagnostic foundation

HOLD THESE

1. One objective → no truth, no goals, no grounding, no memory. **2.** Post-training adds *style*, not *knowledge*. **3.** Hallucination = pattern completion, no ground-truth check (flat OR confidently wrong). **4.** Three categories, three mitigations: retrieval · grounding · verification.

Next: why CLAUDE.md sometimes feels like it doesn't exist.

AGENTS.md: the cross-tool standard

WHAT CLAUDE.MD ACTUALLY IS

Every modern AI coding tool reads an instruction file from the project root before answering. Across vendors the convention converged on one name: **AGENTS.md** — now governed by the Linux Foundation's Agentic AI Foundation, in 60,000+ repos.

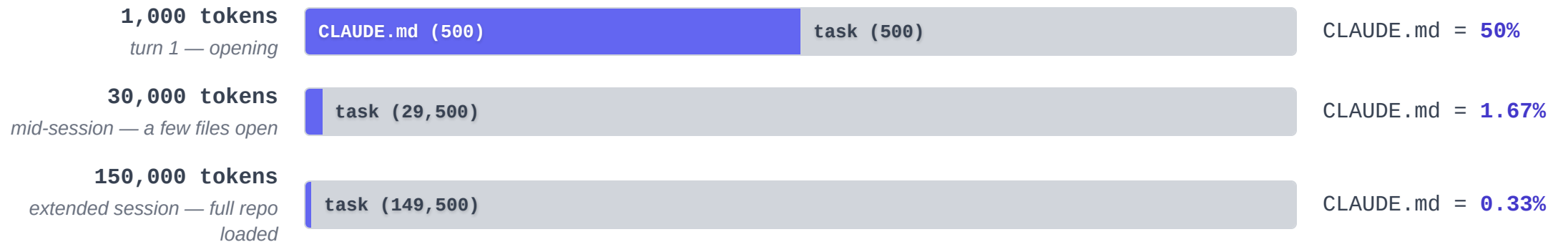
THE WRINKLE

Claude Code does **not** natively read AGENTS.md — it reads `CLAUDE.md`. The community fix: a one-line `@AGENTS.md` reference inside `CLAUDE.md`.

WHY IT MATTERS

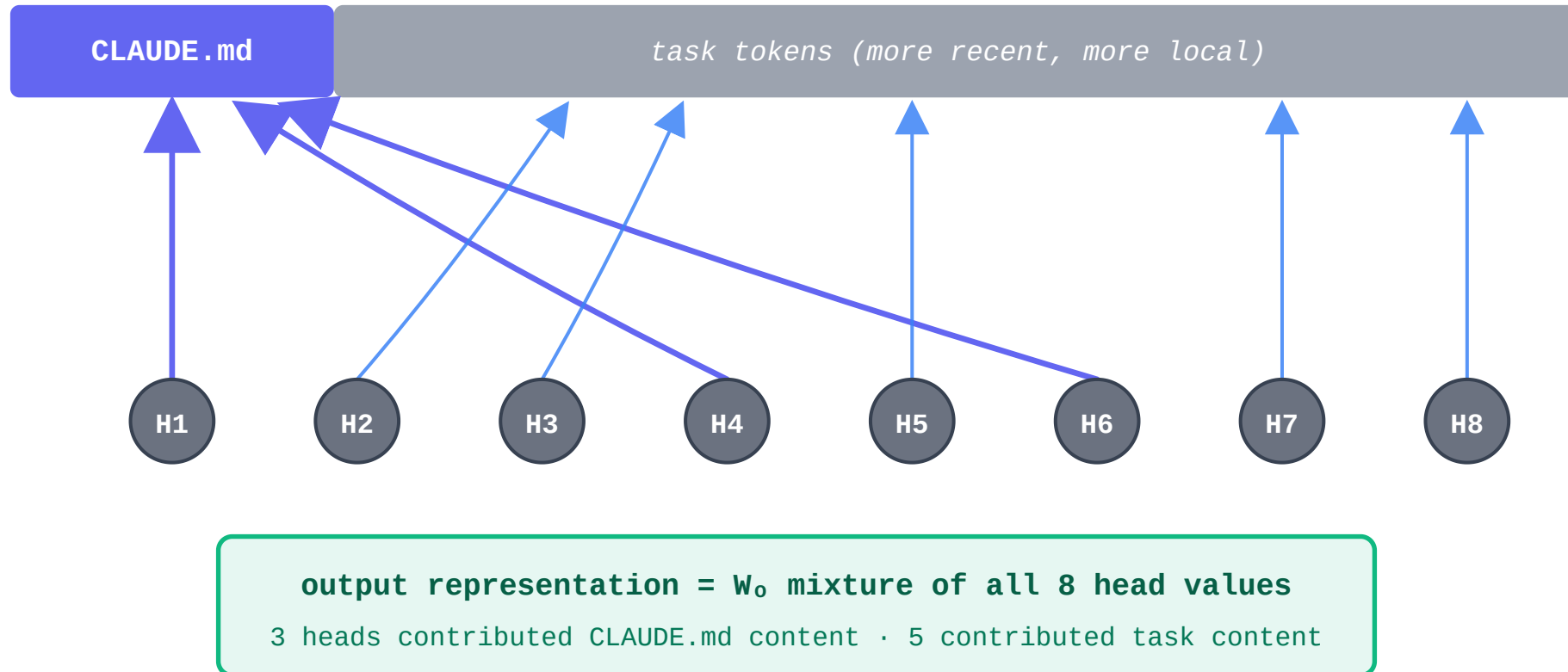
The mechanism we are about to explain is **identical** across AGENTS.md, CLAUDE.md, `.cursorrules`, copilot-instructions — they are all just project-root text the orchestration layer injects as tokens.

CLAUDE.md tokens compete for attention



Same 500-token CLAUDE.md across all three rows. Only the surrounding task tokens grow — the shrinking indigo sliver is its share of positions, not its share of attention: softmax is winner-take-most, so an important token can still win strongly from a tiny slice. But it now competes with far more newer, more local tokens.

Multi-head dilution



Each head routes to wherever its query/key match is strongest. Only some happen to route to CLAUDE.md — 3 of 8 in this caricature. The output is the W_O -weighted mix of all heads, so CLAUDE.md's effective influence is whatever weight that subset carries.

CLAUDE.md does not **command the model. It **influences** the distribution.**

influenced, not commanded — short, recent, aligned instructions influence strongly; long, distant, orthogonal ones influence weakly

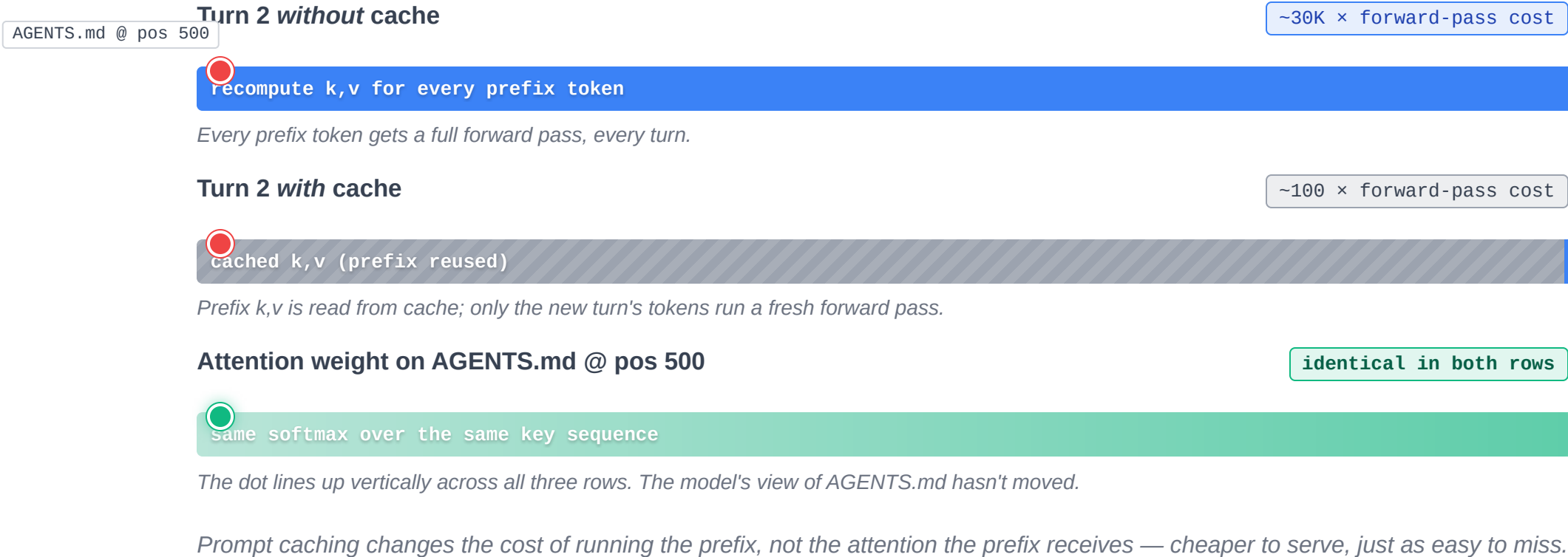
KV and prompt caching: the economics

	KV caching	Prompt caching
Scope	<i>within</i> one response	<i>across</i> requests
Caches	each token's k, v as it's generated	a stable prefix's k, v (system, AGENTS.md, history)
Opt-in?	automatic, intrinsic	sometimes explicit; prefix must be byte-identical
What it changes	recompute cost within a generation	the cost of re-running a long prefix every turn

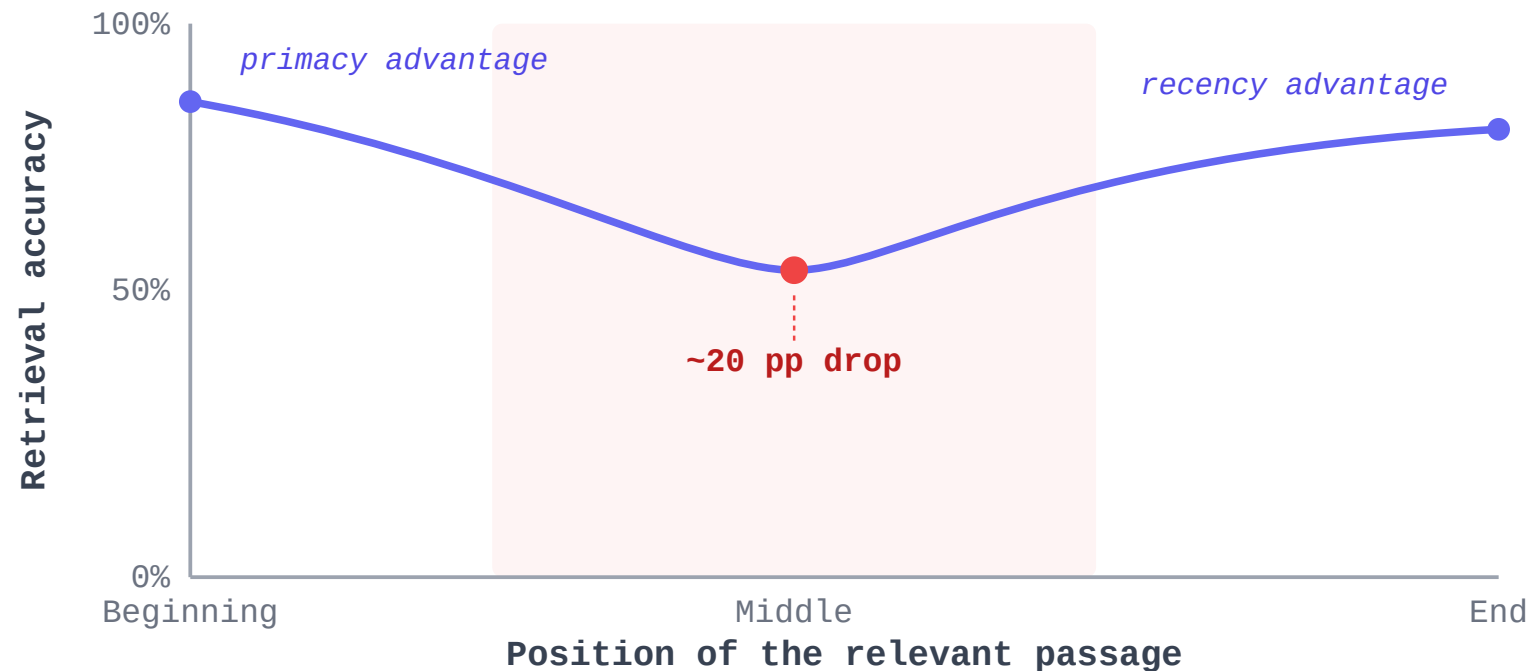
IT CHANGES COST, NOT INFLUENCE

A cached 50K-token AGENTS.md is cheap to keep — but every token still occupies a position and still competes for attention. Caching solves the *cost* of dilution, not dilution itself.

Cached vs uncached context



Lost in the middle: the U-shape



Liu et al. (2023): retrieval accuracy is high at the start (primacy) and end (recency), and drops sharply in the middle — a U-shape, not the flat line naive attention theory would predict.

Engineering rules for instruction files

FRONT-LOAD

The single most important rule belongs on **line 5, not line 350**. Early position has primacy advantage.

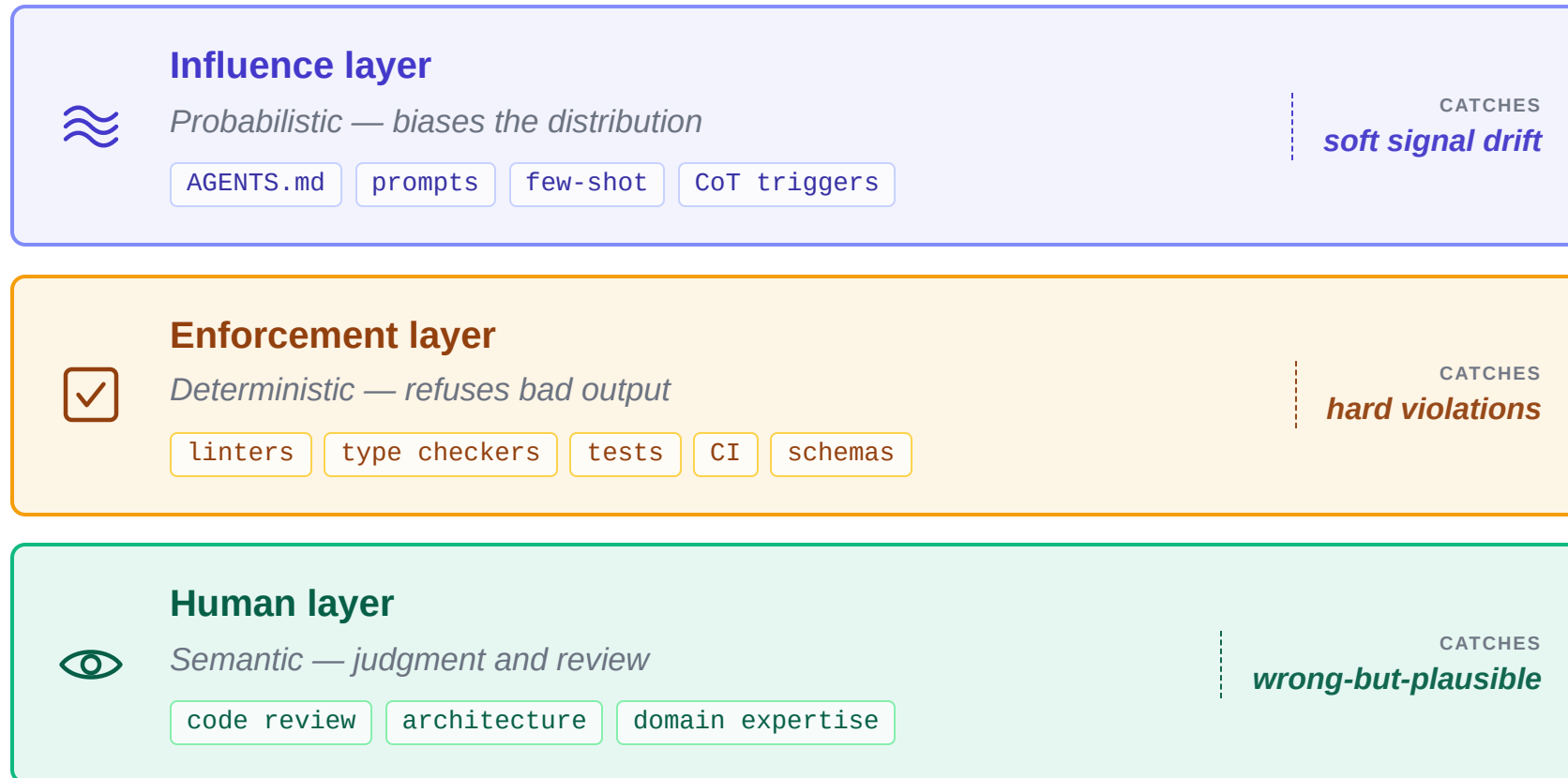
LONGER ≠ MORE RELIABLE

A 100-line CLAUDE.md is more reliable line-for-line than a 600-line one. Buried rules suffer middle-of-context decay.

REPEAT AT THE END

If a rule *must* hold for a critical task, restate it in the user prompt — recency advantage on top of the file load.

The three-layer model



Three parallel safety nets. Each layer catches what the other two miss — remove any one and the gaps line up.

Why all three layers matter

Layer	What it does	What it catches	Limitation
Influence	biases the distribution	soft signal drift	probabilistic, position-dependent, dilutable
Enforcement	refuses bad output deterministically	hard violations	can't tell if the code does the <i>right</i> thing
Human	semantic judgment	wrong-but-plausible	does not scale to every line

TEACHING-REPO ASYMMETRY

Your Locust repo is deliberately *thin* on enforcement — so the influence layer carries extra weight, and you feel exactly which failures it misses. In production the balance inverts.

From consequences to controls

THE PIVOT

Block A named the failures. **Block B names the controls.** Every pattern ahead is a deliberate use of an L5 mechanism to shift the distribution toward the correct continuation.

The question Block B asks: *what can you do at prompt-construction time to make the distribution sharper, more peaked, more aligned with correct answers?*

TAKE A BREATH

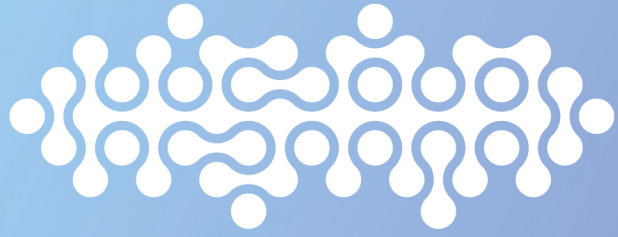
This is the natural break point in a 2×90 session. We switch from "why it fails" to "how to steer it."

Checkpoint: Block A synthesis

FIVE CLAIMS TO KEEP

1. Pre-training = one objective, no truth. **2.** Post-training = style, not knowledge. **3.** Hallucination = pattern completion, no ground-truth check (flat OR confidently wrong). **4.** CLAUDE.md = influence, not command. **5.** Reliability = influence + enforcement + human.

If you remember nothing else from Block A, remember these. Block B builds a control on top of each one.



Hochschule Aalen
UNIVERSITY OF APPLIED SCIENCES

Engineering the Distribution

B

Softmax with temperature

$$p_i = \frac{e^{\ell_i/T}}{\sum_j e^{\ell_j/T}}$$

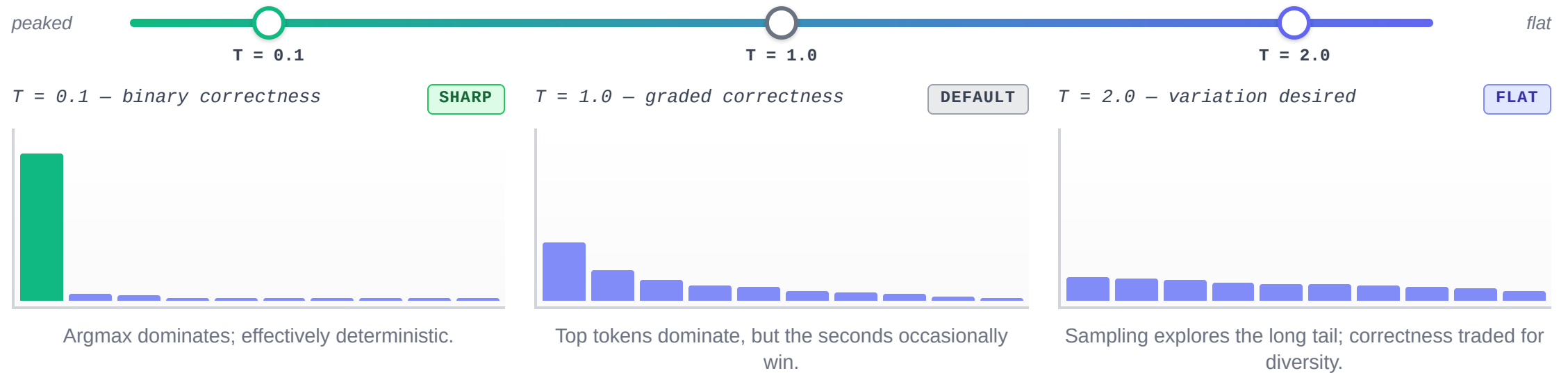
T < 1 — SHARPENS

Divide the logits by a small number → the winner takes more. $T \rightarrow 0$ becomes argmax.

T > 1 — FLATTENS

Divide by a large number → the long tail gains mass. More variation, less commitment.

Three regimes: $T = 0.1, 1.0, 2.0$



Temperature turns the same logits into different distributions. The model and its parameters are unchanged — only how its softmax is shaped at sampling time.

Temperature decision tree

Task type	Recommended T	Why
Binary correctness	$T = 0$	one right answer; commit, repeatable, debuggable
Graded correctness	$T \approx 0.3\text{--}0.7$	quality dimension, small variation fine — most code generation
Variation desired	$T \geq 1.0$	brainstorming, naming, alternatives — diversity on purpose

THE ONE-LINER

Match the sharpness of the output to the sharpness of the *task's correctness criterion*. There is no universal "right temperature" — only the right T for *this* task.

Worked example: same logits, three temperatures

Token	Logit ℓ	$p(T=0.1)$	$p(T=1.0)$	$p(T=2.0)$
glowing	1.8	≈ 0	0.058	0.107
potion	4.2	≈ 1.000	0.637	0.354
fire	3.1	1.7×10^{-5}	0.212	0.204
troll	0.9	≈ 0	0.024	0.068
smashes	1.1	≈ 0	0.029	0.075
(+5 more)	...	≈ 0	...	...

TEMPERATURE RE-WEIGHTS THE DICE — IT DOES NOT CHANGE WHICH FACE HAS THE MOST DOTS

The argmax never moves: **potion** wins at every T . At $T=0.1$ it carries ~all the mass; at $T=2.0$ the runner-ups become real candidates.

The first-token cascade

T = 0 IS DETERMINISTIC, NOT CORRECT

Greedy decoding picks the per-step argmax. If the step-1 token is wrong — a confident wrong answer that beat the genuinely-correct second-place token — every later token is conditioned on it. The cascade compounds.

WHY SOME ENGINEERS RUN CODE-GEN AT T = 0.3

A little stochasticity lets you, across a few samples, occasionally escape a wrong cascade. Greedy without CoT is the most cascade-vulnerable configuration — which is one reason chain-of-thought (ch 13) helps.

What temperature does not solve

IT CANNOT SUPPLY A MISSING ANSWER

If the model genuinely doesn't know, lowering T makes it commit *harder* to the modal-but-wrong answer. The ch-4 hallucinations (flat OR confidently wrong) don't vanish at $T=0$ — they become more *repeatable*. Temperature cannot add facts, and it cannot enforce constraints.

THE LEVER MOVES EARLIER

Temperature is the easiest knob, with the smallest effect. The rest of Block B shapes the distribution at **prompt-construction time** — far more leverage.

Boltzmann: where the name comes from

$$p_i = \frac{e^{-E_i/kT}}{Z}, \quad Z = \sum_j e^{-E_j/kT}$$

SOFTMAX-WITH-TEMPERATURE IS THE BOLTZMANN DISTRIBUTION

Identify logit ℓ_i with $-E_i / k$ and the two expressions coincide term-for-term. The T here is physical temperature, in kelvin — a 150-year-old object from statistical mechanics.

THE LIMITS TRANSFER FOR FREE

Hot system $\rightarrow$ near-uniform (high entropy). Cold system $\rightarrow$ concentrates on the ground state (argmax). $T \rightarrow 0$: only the lowest-energy state. $T \rightarrow \infty$: uniform.

What is in-context learning?

DEFINITION

In-context learning (ICL) is the phenomenon by which a model, *with no weight updates*, changes its behaviour in response to examples placed in its prompt. Three input → output pairs in; the fourth input gets an output that follows the pattern.

STRANGE, ON ITS FACE

No gradient descent. No fine-tuning. No training. Just a forward pass over a longer context that includes examples. ICL is *emergent* — it appears at scale, in models trained purely on next-token prediction.

GPT-3: the ICL headline

BROWN ET AL. (2020) — "LANGUAGE MODELS ARE FEW-SHOT LEARNERS"

A 175B-parameter model could perform tasks it was never explicitly trained on, simply by seeing a few examples in the prompt.

Few-shot performance scales sharply with model size; one-shot and zero-shot do not.

THE REAL HEADLINE

Not "we made a bigger model" — *"an unexpected new capability emerged at scale: it can learn from examples in its prompt, with no parameter updates."* That capability is what every pattern in this lecture exploits.

ICL mechanism, in L5 vocabulary

NO NEW ALGORITHM — THE SAME ATTENTION MACHINERY

The examples are **additional context tokens**. The forward pass — unchanged from Lecture 5 — builds Q, K, V, computes attention, mixes values, layer by layer. At the final position, heads route back over the example tokens; the values they pull in shift the representation, which shifts the unembedding logits, which shifts the output distribution.

NOT "TRAINING DATA"

The weights are unchanged. The behaviour shift is entirely a consequence of changing the conditioning context for the same fixed function. **ICL is attention-based conditional pattern matching.**

Induction heads: the routing pattern

OLSSON ET AL. (2022)

Trained transformers contain attention heads that implement one very specific routine — and these heads are tightly correlated with the model's ability to do in-context learning.



THE RULE, IN ONE LINE

If you saw `[A][B]` earlier, and now you see `[A]`, predict `[B]`. **Find the previous match → look at the next token → route there.**

PREDICT IT

apple → red · banana → yellow · cherry → ?

Call the next token before the reveal. What is the induction head about to do?

Induction head walkthrough



The current token **cherry** (pos 7) appeared at pos 5; the token right after that match is **red** (pos 6). The head routes from pos 7 back to pos 6 — so **red**'s value mixes into the current representation, biasing the next token toward **red**. Find previous match → look at the next token → route there.

Induction heads work across domains

def get_user(id: int . . . def get_order(→ id: int

Q: capital of France? A: Paris . . . Q: capital of Spain? → A: ...

SAME HEAD, ANY PAYLOAD

Code signatures, $Q \rightarrow A$ formats, translation pairs — the induction head doesn't care what the tokens *mean*. It matches the **shape** of "what followed this before" and continues it.

Min et al.: format matters more than content

MIN ET AL. (2022)

Take a classification task. Replace the demonstration labels with **random labels**. The expected result — accuracy collapses. The actual result: **only slightly lower than correct labels, and much higher than zero-shot**.

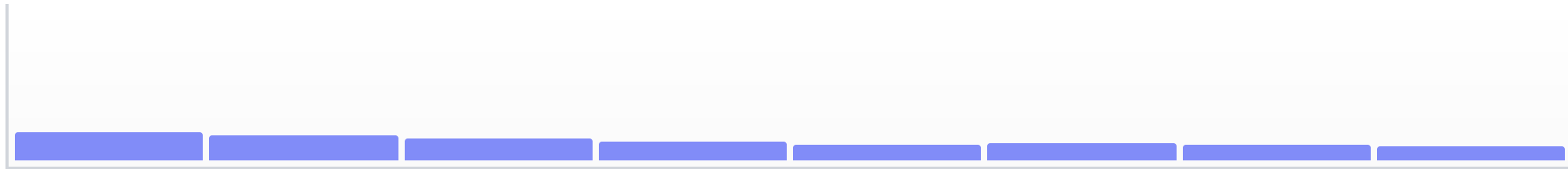
WHAT ACTUALLY TRANSFERS

The **format** (input on one line, output on the next), the **input distribution**, the **label space**, and the **task identity** — not primarily the correct answer. The demonstrations are *format and task-identity cues* first, answer guides second.

Logit trace: the zero-shot baseline

Zero-shot – no demonstrations

FLAT



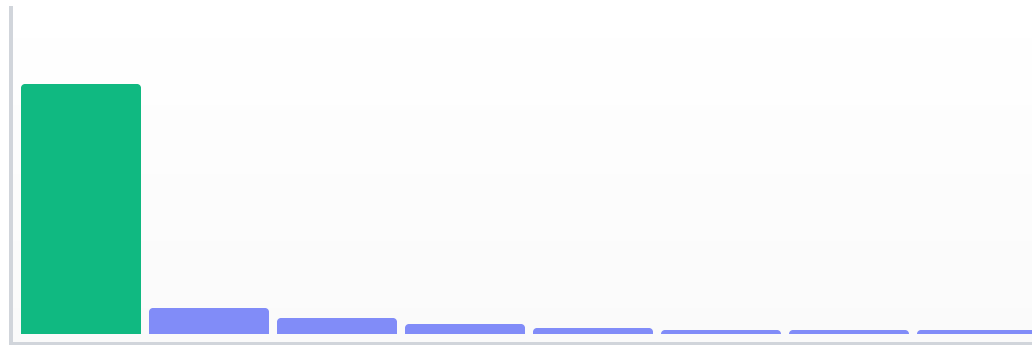
Mass spread across many continuations — the top-1 may not even be a label token (`positive ?` `mixed ?` `the ?`).

Query review: "The plot dragged for the first hour but the ending saved it." With no demonstrations, the distribution over the next token is flat — the model has no structural cue for the task.

Logit trace: few-shot correct vs random

Few-shot – correctly-labelled

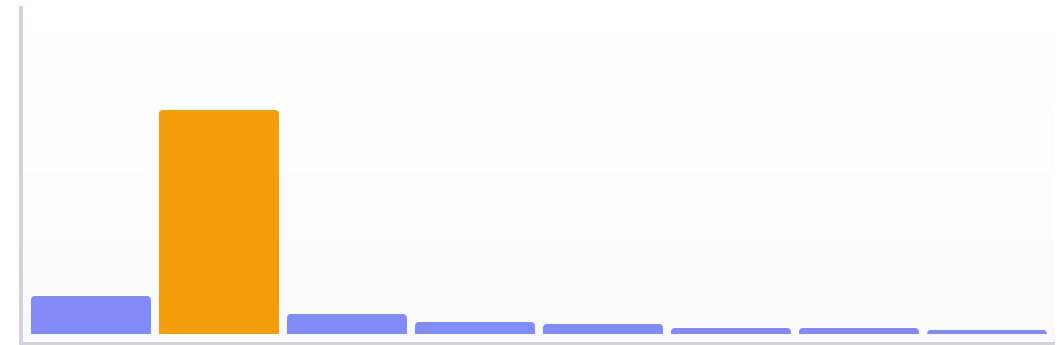
PEAKED ✓



Peaked on **positive** — the right label for this review.

Few-shot – randomly-relabelled

PEAKED, BUT WRONG



Still peaked, still on a *label* token — but **negative**. Format transferred; correctness didn't.

Demonstrations shape the peak; their content shapes which peak wins. The "wrong" peak on the right is the load-bearing observation behind Min et al.'s result.

For project-specific conventions, **examples beat instructions**. Format transfers;
content is secondary.

the bridge to few-shot in practice — show, don't tell

Checkpoint: ICL and induction heads

FOUR TO KEEP

1. ICL = attention on demonstrations, no weight update. 2. Induction heads: $[A][B] \dots [A] \rightarrow [B]$. 3. Format > content (Min et al.). 4. This is why few-shot works — and why it works even with imperfect examples.

§11 is the theoretical anchor for everything in §12–§14. Next: turning it into engineering.

Technique card · Zero-shot

Zero-shot	baseline — "just ask"
What it is	task description + input; no demos, no role, no reasoning trigger
Mechanism (L5)	relies on pre-training + RLHF; distribution shaped by training statistics alone
SE example	<code>normalize_email</code> — direct instruction, general Python knowledge
Evidence cue	a direct instruction; no examples, no role, no reasoning request
When it fails	conventions or coverage matter → task under-specified → model fills gaps from priors → hallucination

THE BASELINE

Every other pattern improves on this by *adding context that shifts the distribution*. Zero-shot is fine when the task is well-represented in training data.

Few-shot engineering rules

Rule	Detail	Source
k = 2–4 examples	the first 1–2 carry most of the lift; past ~8, diminishing returns	§12.1
Order matters	put your <i>best</i> example last — recency bias near the query	§12.2
Format > content	match input shape, output shape, delimiters	Min 2022
Avoid conflicts	conflicting demos → flatter distribution than <i>no</i> examples	§12.2
Known-good only	the model faithfully repeats bugs in your examples	§12.4

Technique card · Few-shot

Few-shot	show k input → output pairs
What it is	k = 2–4 complete input → output examples before the query
Mechanism (L5)	induction heads route from query to earlier outputs (Olsson 2022)
SE example	project-convention code gen — <code>get_user_by_id</code> (exercise Card 2)
Evidence cue	2–4 demonstration pairs; "follow this pattern"
When it fails	wrong / conflicting / off-distribution examples → garbage continuation

WATCH THE ORDER

Recency bias: the example nearest the query carries the most weight. Forward link to L7 — codified few-shot patterns become **skills**.

Examples beat instructions for style

FOR PROJECT-SPECIFIC CONVENTIONS, SHOW — DON'T TELL

Paste 1–2 existing repo functions before asking for a new one, and the model matches naming, ORM usage, return style, and type hints **in one shot**. A prose description of those same conventions rarely achieves the same consistency.

WHY

The induction heads route over *genuine examples* of the codebase's idioms. Instruction-style asks the model to *reconstruct* the pattern from a verbal description — many constraints, many ways to misread. Few-shot is indispensable for AI coding tools for exactly this reason.

Worked example: Stack Exchange (Act 1)

Stack Overflow Challenge (Act 1) – same task, three ways

zero-shot "...query the Stack Exchange API for monthly counts..."
→ **hallucinates** `/api/v2/questions/monthly`

few-shot + 2 real calls to `api.stackexchange.com/2.3/questions`
→ **endpoint hallucination collapses**

few-shot + CoT + "first list the APIs you'll call, then code"
→ **catches the hallucination in the plan, before code**

THE PROGRESSION

zero-shot → few-shot → few-shot + CoT — the decision tree we formalise in §16, anchored in your own Act 1 artefacts.

Chain-of-thought: headline and mechanism

WEI ET AL. 2022 · KOJIMA ET AL. 2022

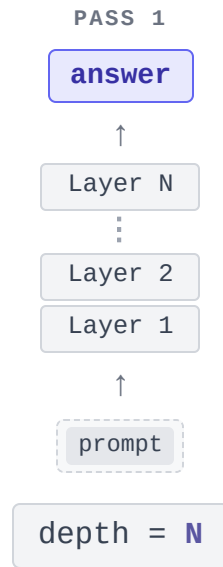
Prompt the model to *show its working* before the answer, and accuracy on reasoning-heavy tasks jumps **10–30 points** (it doubles on some). Kojima: most of the gain needs no examples — just "*Let's think step by step.*"

THE MECHANISM, IN ONE LINE

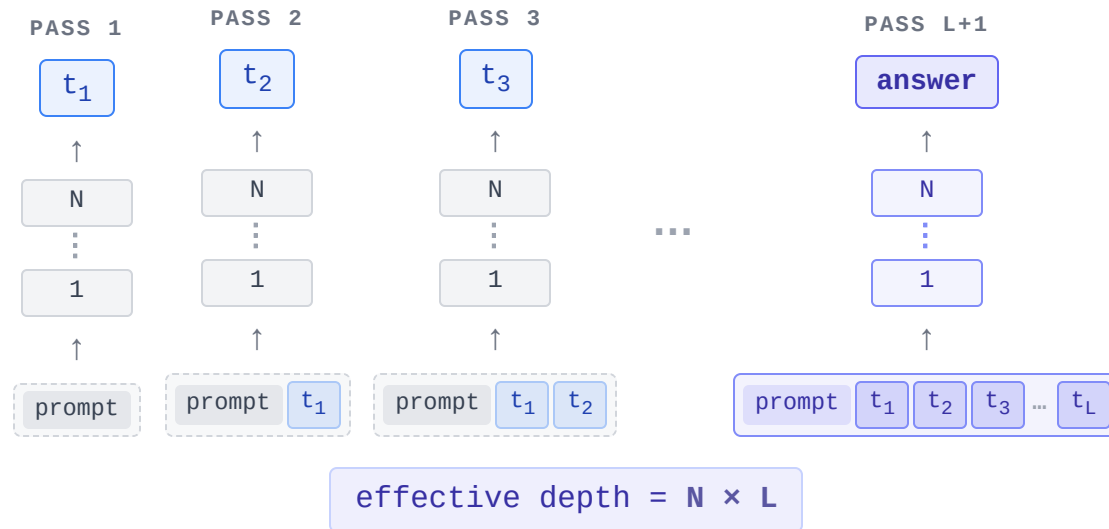
Intermediate reasoning tokens become attended-to context for later tokens. CoT **trades fixed depth for length** — the model writes notes to itself, then reads them back.

Depth for length: direct vs CoT

Direct generation — one forward pass



Chain-of-thought — L+1 passes; each reads the growing context



Each pass is one full N-layer forward pass emitting one token, appended to the next pass's input. By the time the answer pass runs, its input contains every prior intermediate token — each itself the output of N layers. Same architecture, same N; L-times the effective depth.

Technique card · Chain-of-thought

Chain-of-thought	reason before answering
What it is	ask for intermediate reasoning ("think step by step" / "plan, then code")
Mechanism (L5)	auto-regressive loop as scratchpad; trades depth for length (Wei/Kojima 2022)
SE example	"list the APIs you'll call, then write the code" — catches hallucinations
Evidence cue	"think step by step", "plan first", "explain your reasoning"
When it fails	simple lookups (pure overhead); cascades a wrong intermediate step

ROW 5 IS CRITICAL

CoT is not free — it costs tokens, can cascade errors, and the visible reasoning is **not necessarily faithful** (§15.3). Don't treat the chain as an audit trail.

SE chain-of-thought: plan, then code

THE PROMPT

"First list the external APIs you need. Then check: does each API actually exist? *Then* write the implementation."

WHY IT WORKS — BLOCK A MEETS BLOCK B

§4's API hallucination was a no-ground-truth-check problem. CoT adds intermediate tokens that force the model to **name the API explicitly** — where it can be checked against demonstrated patterns. The hallucination surfaces in the plan, not in the code.

When CoT hurts: four failure modes

Simple lookups

"Capital of France?" in ten times the tokens. Pure overhead — the answer was one forward pass away.

Cascade on reasoning

A wrong intermediate step conditions every step after it — first-token cascade, moved to the reasoning chain.

No knowledge to add

CoT extends depth, not knowledge. It can decompose a known task; it cannot summon a fact the model lacks.

Faithfulness illusion

You can read the reasoning — but it is not necessarily faithful to the internal computation. Verify with tests, not prose.

CoT expressivity: stated, not proved

FENG ET AL. 2023 · MERRILL & SABHARWAL 2023

There are problem classes *provably hard* for direct generation in a fixed-depth transformer that become tractable once the model may generate intermediate reasoning tokens. CoT moves the model into a strictly more expressive regime.

WHY IT MATTERS

The chain-of-thought gain is not an empirical accident. It is the model accessing more computation by spending more tokens — a circuit-complexity result we state but do not prove.

Reasoning models: productised CoT

O1 · DEEPSEEK-R1 · EXTENDED THINKING

Same attention, same FFN, same N layers, same loop — **no new mechanism**. They are trained to generate long internal reasoning traces before answering. The auto-regressive scratchpad is the entire story; the model just uses it aggressively.

WHAT CHANGES FOR PROMPTING

Prompt for the **goal, constraints, context, tools, output format, and verification criteria** — not for a visible chain-of-thought. Over-prompting "let's think step by step" on a reasoning model degrades quality.

Self-consistency: recipe and mechanism

Step	What happens	L5 mechanism
1. Sample N paths	N CoT chains at $T > 0$ (≈ 0.5)	non-zero temperature $\rightarrow$ diverse reasoning
2. Each reaches an answer	every chain ends in a final answer	auto-regressive scratchpad
3. Majority vote	aggregate the N final answers	distribution over <i>answers</i> is narrower than over <i>paths</i>
4. Return the mode	take the most common answer	variance reduction (Wang 2022)

WHY IT WORKS

Many reasoning routes; they often converge on the same answer. Trust where independent noisy votes agree — the first **test-time-compute scaling** result.

Technique card · Self-consistency

Self-consistency	sample many, vote
What it is	sample N CoT paths at $T > 0$; return the majority answer
Mechanism (L5)	variance reduction; paths broad, answers concentrate (Wang 2022)
SE example	hard one-off eval; pipelines where correctness > latency
Evidence cue	multiple independent generations; a voting / aggregation step
When it fails	the <i>wrong</i> answer is the mode; costly ($N \times$ inference)

PRODUCTION REALITY

You rarely implement this by hand — reasoning models internalise it. But "many paths, majority vote" is the first test-time-compute-scaling idea. When you see "extended thinking," this is the ancestor.

Variance reduction and the consensus signal

THE FORMAL CORE

If any single path is correct with probability $p > 0.5$, the *majority* of N independent samples is correct with probability much higher than p — the law of large numbers, applied to reasoning.

CONSENSUS AS A DIAGNOSTIC

38/40 agree → high confidence. 12/40 split → the task is ambiguous; **escalate to human review**. In practice this is what self-consistency is most used for — detecting when the model is hedging, not winning an extra accuracy point.

Checkpoint: prompting patterns

FIVE CLAIMS TO KEEP

1. Zero-shot = baseline, relies on RLHF. 2. Few-shot = induction heads; format > content. 3. CoT = auto-regressive scratchpad; depth for length. 4. Self-consistency = variance reduction over paths. 5. Every pattern has a failure mode.

Four patterns covered. One more — role priming — in the synthesis. Then the pattern library ties it together.

Structured outputs and constrained decoding

WHEN THE OUTPUT HAS A GRAMMAR, DON'T RELY ON PROMPTING

JSON, a typed tool call, an enum value, a schema — the inference engine compiles the grammar to an **FSM / pushdown automaton** and masks the logits at every step so only grammar-valid tokens can be sampled. The model never has the *option* to emit an invalid character.

IT ENFORCES SHAPE, NOT TRUTH

A schema-valid JSON object can still contain hallucinated values. Use prompting to explain the task; use constrained decoding + a parser to enforce the shape. This is enforcement at the sampling layer — not prompting.

When prompting is the wrong tool

What you need	Why prompting alone fails	Right tool
Output adheres to a schema	describes, can't enforce structure	constrained decoding (§15.1)
Answer grounded in current facts	pre-training is frozen	retrieval (RAG), tool calls, doc injection
Code correct, not just plausible	fluent code can still fail on real inputs	tests, static analysis, type checks, review
Action respects trust boundaries	instructions are influence, not enforcement	permissions, sandboxing, secrets isolation, audit

Visible reasoning $\neq$ trustworthy reasoning

FAITHFULNESS

A chain-of-thought trace — elicited by "think step by step" or produced internally by a reasoning model — is a **signal, not a guarantee**. The model can generate a clean, plausible chain that *justifies* the answer post-hoc rather than *deriving* it (Lyu et al. 2023).

THE RULE

Ask for a plan, trace, or checklist when useful. For high-stakes verification, lean on outcome tests, schema validation, source checks, and human review — **not** the readable chain.

Model-class decision matrix

Model class	Best prompting style	Common mistake
Standard chat	clear task, examples, output format, optional CoT	trusting the visible chain as audit trail
Reasoning model	goal, constraints, criteria; let it reason internally	over-prompting visible CoT
Coding agent	repo context, task boundaries, acceptance tests, scope	treating CLAUDE.md as enforcement
Tool-using agent	clear permissions, typed schemas, scoped credentials	trusting unsanitised tool returns

THE FRAMING

The right prompting pattern is a function of the **model class**, not a universal recipe.

Security boundary: prompt injection

PROMPT INJECTION — OWASP LLM01

Untrusted input — in a document, a web page, a tool result — can override the application's instructions. To attention, CLAUDE.md tokens and attacker-injected tokens look the *same*. The model isn't "tricked"; it follows the most plausible continuation.

NOT ALL INSTRUCTION FAILURES ARE ACCIDENTAL

CLAUDE.md drift is a **reliability** problem. Prompt injection is a **security** problem. For adversarial input, assume the influence layer is *defeated* — safety lives in enforcement (sandboxing, validation, permissions) and human review.

Technique card · Role / persona priming

Role priming	set perspective & register
What it is	assign a role — "Act as a senior backend engineer"
Mechanism (L5)	RLHF taught persona tokens to bias attention toward a matching subspace
SE example	code review (Card 3): role + rubric + structured output
Evidence cue	"Act as...", "You are...", explicit role + rubric
When it fails	<i>theatre</i> if the role changes only tone, not substance

ROLE ALONE IS INSUFFICIENT

"You are an expert" without a rubric is folklore. Pair the role with a **rubric** (what to check) and **structured output** (how to format) — they amplify it.

Pattern summary: six rows

Pattern	What you do	Mechanism (L5)	Key citation
Zero-shot	ask directly	pre-training + RLHF priors	—
Temperature	set T at sampling	scales logits before softmax (§14.2)	—
Few-shot	k input → output pairs	induction heads	Olsson; Min 2022
Chain-of-thought	reason before answering	auto-regressive scratchpad	Wei; Kojima 2022
Self-consistency	sample N, majority vote	variance reduction over paths	Wang 2022
Role priming	assign a persona	re-weights heads toward a subspace	§16.1

PICK THE PATTERN

"Match our repo's error-handling style in a new function."

Zero-shot, few-shot, CoT, self-consistency, or role priming? Decide, then we walk the tree.

Decision tree: choosing a pattern

Ask, in order	If yes →
Well-known, standard task?	Zero-shot · $T = 0.3\text{--}0.7$
Project-specific style / format?	Few-shot · 2–4 examples, best last
Needs intermediate reasoning?	Zero-shot CoT first; escalate to few-shot CoT
Accuracy-critical, can afford $N\times$?	Self-consistency on CoT
Judgment / perspective task?	Role priming + rubric + structured output

THAT'S THE WHOLE LIBRARY

Tree-of-Thoughts, ReAct, Self-Refine, Reflexion are *orchestration* patterns built on top of these five — Lecture 7 and Act 3.

How each pattern fails

Pattern	Failure mode	Mechanism of failure
Zero-shot	hallucinates a plausible-but-wrong answer	no ground-truth check; flat OR confidently wrong (ch 4)
Temperature	T=0 first-token cascade	auto-regressive conditioning (ch 10)
Few-shot	imitates misleading examples	induction heads route regardless of correctness
Chain-of-thought	builds on a wrong intermediate step	scratchpad amplifies early commits
Self-consistency	consensus on a misconception	aggregates the mode, not the truth
Role priming	confident in the wrong register	persona shifts heads to a wrong subspace

No pattern is uniformly safe.

Better prompting: frequent → occasional. Better enforcement: occasional → caught. Better human review: caught → addressed.

How to diagnose a prompt: the 4-field lens

Field	The question to ask
Name	which strategy is being used?
Evidence	what phrase in the prompt reveals it?
Appropriateness	is this the right strategy for <i>this</i> task?
Anti-pattern	how does this strategy fail <i>here</i> ?

THE REUSABLE METHOD

Four questions, applied to any prompt. The exercise applies them to 12 real-world prompts across 11 SE task categories.

Card 3 walkthrough · role priming done right

Field	Card 3 — code review
Name	role / persona priming + rubric + structured output
Evidence	"Act as a senior backend engineer", explicit rubric, output format
Appropriate?	Yes — judgment task; perspective matters; rubric covers criteria
Anti-pattern risk	if the role is only tone ("be professional") with no rubric → theatre

THREE INGREDIENTS

role (who) · rubric (what to check) · structured output (how to format). Remove any one and the prompt degrades.

Card 12 walkthrough · CoT overuse

Field	Card 12 — over-applied CoT
Name	chain-of-thought (overuse)
Evidence	"think step by step" on a simple lookup ("which HTTP code = not found?")
Appropriate?	No — no multi-step reasoning; CoT adds overhead, risks a cascade
Anti-pattern	CoT on a task better served by zero-shot + validation

THE DEEPER LESSON

More technique ≠ better. The right answer is zero-shot; the CoT only buys tokens, latency, and cascade risk.

Supporting techniques

Rubric

what to check — makes role priming concrete.

PAIRS WITH ROLE

Structured output

how to format — makes any response parseable.

PAIRS WITH ANY PATTERN

Context grounding

paste relevant code / docs — makes few-shot specific.

PAIRS WITH FEW-SHOT

Validation

tests, schemas, CI — catches what prompting misses.

THE ENFORCEMENT LAYER

AMPLIFIERS, NOT PATTERNS

These four are not prompting patterns — they make the five primary patterns work. The exercise cards span all four.

Lab · Prompt Strategy Diagnosis

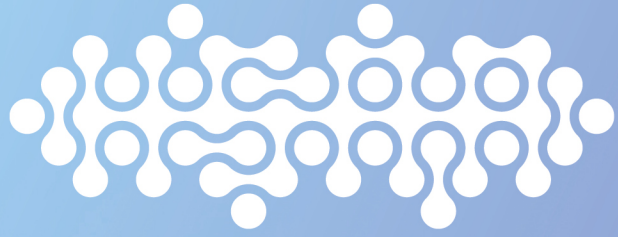
EXERCISE — 12 CARDS, 11 SE TASK CATEGORIES

For each prompt card, apply the four-field lens: name the strategy · cite the evidence · judge appropriateness · name the anti-pattern — then rewrite a stronger version.

25 min (pairs) · 45–60 min (extended) · output: 4–6 diagnosed cards + reflection · interactive self-study quiz + full workshop guide online

IN CLASS

Work 4–6 cards in pairs, then debrief Cards 3 and 12 — the two we just walked.



Hochschule Aalen
UNIVERSITY OF APPLIED SCIENCES

Closing the Board

C

Mystery board: almost fully resolved

#	Mystery	Status after L6
1	Why can't AI count letters?	Solved (L4) — tokenisation hides characters
2	Why does CLAUDE.md help in some sessions?	Solved (L5+L6 §6) — injection + drift + dilution
3	Why were English prompts often better?	Solved (L4+L5) — token efficiency + attention competition
4	Why was curated context better?	Solved (L5) — every token competes for the heads' routing
5	Why hallucinate API endpoints?	Solved (L6 §4–5) — no ground-truth check; factual
6	Why do vague prompts give nonsense?	Solved (L6 §4) — under-specified; model fills gaps from priors
7	Why is CLAUDE.md compliance inconsistent?	Solved (L6 §6–7) — attention competition + lost-in-middle
8	Why do tools differ on the same instructions?	Partly (L6 §3) — role hierarchy; finishes in L7
9	Why does few-shot work?	Solved (L6 §11) — induction heads; format > content
10	Why does "think step by step" help?	Solved (L6 §13) — auto-regressive scratchpad
11	Why does sampling + voting help?	Solved (L6 §14) — variance reduction over paths

What Lecture 7 adds

LECTURE 7 — THE ORCHESTRATION LAYER

The system *outside* the model that decides what tokens it sees, when, and in what order. It resolves **mystery 8** fully — same model, different orchestration, different behaviour.

Skills

codified few-shot patterns, loaded on demand — what we started today, formalised.

MCP

the open standard for connecting models to tools and data; the security boundary lives here.

SWE-Bench

how the field measures end-to-end agent quality — and why its protocol is few-shot.

Lab and deliverables

Exercises (separate slots)

Temperature experiments · hallucination taxonomy · context-vs-hallucination · the Prompt Strategy Diagnosis lab (12 cards).

Reflection log

The two halves in your words · your own pattern decision tree · a CLAUDE.md audit · the most counterintuitive finding · one question for L7.

OPTIONAL READING

The originals — Brown, Wei, Kojima, Min, Olsson, Wang, Liu, Ouyang, DeepSeek-R1 — are open-access; the figures alone are worth the time (§18.3).

Better prompting: **frequent** → **occasional**. Better enforcement: **occasional** → **caught**. Better human review: **caught** → **addressed**.

three layers, three guarantees — the operational takeaway of Lecture 6



Hochschule Aalen
UNIVERSITY OF APPLIED SCIENCES

Questions?

STEERING THE DISTRIBUTION