# Uncertainty, Belief, and Trust in Language Models

Timothy Baldwin

November 7, 2026



Mohamed bin Zayed
University of
Artificial Intelligence

# 1. The Fundamentals of Uncertainty and Belief

# Wouldn't It Be Nice ...



Translate the following sentence into French: My hovercraft is full of eels




Mon aéroglisseur est plein d'anguilles

`Confidence = 0.99`



Translate the following sentence into Wizzwiggian: My hovercraft is full of eels




Zizzi-wiggle oof eeler-wee!

`Confidence = 0.00`

# Wouldn't It Be Nice ...



Is the check-in queue long?

Yes, the check-in queue appears to be quite long.
You can see many people lined up in multiple lanes, stretching far back toward the check-in counters. This suggests a significant wait time.

`Confidence = 0.20`

# Wouldn't It Be Nice ...



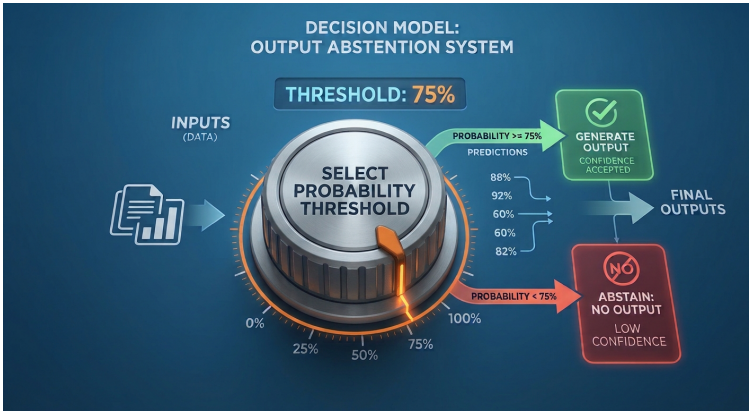
actually, I'm flying business class

If you're flying **business class**, your wait should be even shorter.

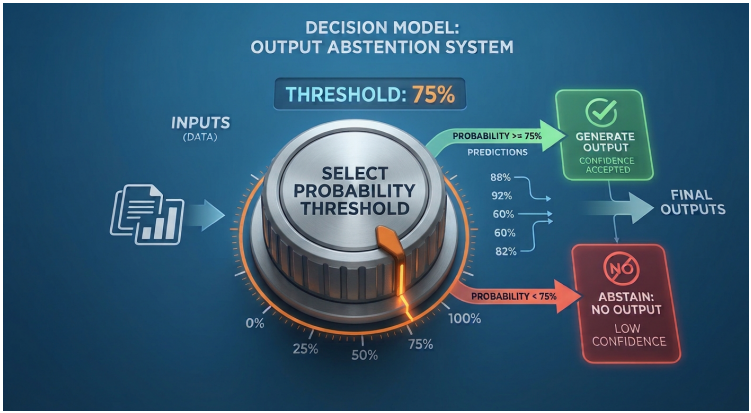
From the image, the longest queues seem to be for standard economy check-in farther down the hall. The area closer to the camera—where you mentioned your airline is—already looks relatively quiet, so with business-class priority you'll likely move through very quickly.

`Confidence = 0.03`

# Wouldn't It Be Nice ...



# Wouldn't It Be Nice ...




- JEV is a prominent recent example of a decision model with confidence outputs, but the idea is far from new ...

# Definitions

- **Uncertainty** = the probability of obtaining an equivalent or better result randomly
- **Belief** = a probability distribution that captures everything a model "knows" about a concept, including the uncertainty surrounding it ... expect Bayesian modelling!

# Definitions

- **Uncertainty** = the probability of obtaining an equivalent or better result randomly
- **Belief** = a probability distribution that captures everything a model "knows" about a concept, including the uncertainty surrounding it ... expect Bayesian modelling!
- **Calibration** = the model confidence is an accurate representation of the likelihood of the model being correct
  - tends to be evaluated in aggregate (e.g. ECE = expected calibration error)
  - tends to be achieved through post-training vs. uncertainty quantification which is generally output directly from the model or extracted from model logits/weights

# Definitions

- **Uncertainty** = the probability of obtaining an equivalent or better result randomly
- **Belief** = a probability distribution that captures everything a model "knows" about a concept, including the uncertainty surrounding it ... expect Bayesian modelling!
- **Calibration** = the model confidence is an accurate representation of the likelihood of the model being correct
  - tends to be evaluated in aggregate (e.g. ECE = expected calibration error)
  - tends to be achieved through post-training vs. uncertainty quantification which is generally output directly from the model or extracted from model logits/weights
- *Note that "certainty", "belief", and "trust" are used in the statistical senses of the terms throughout this talk, and are in no way an attempt to anthropomorphise the models*



# What Are the Theoretical Foundations of Uncertainty?

- Uncertainty is based on the principle that we always lack some information about model prediction and model parameters due to **(1) limited training data** ("epistemic" uncertainty) and **(2) task ambiguity** ("aleatoric" uncertainty):

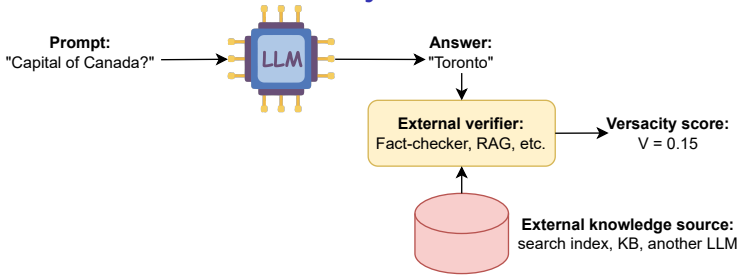
$$\mathbf{U}_{\text{pred}} :\triangleq \boldsymbol{H}(\boldsymbol{Y} \mid \boldsymbol{x}, \boldsymbol{D}) = E_y[-\log p(y \mid x, D)] \qquad \theta \sim \text{Prior}$$

Predicted variable $Y$ | Test instance $x$ | Training data | Realization of $Y$ | Model parameters

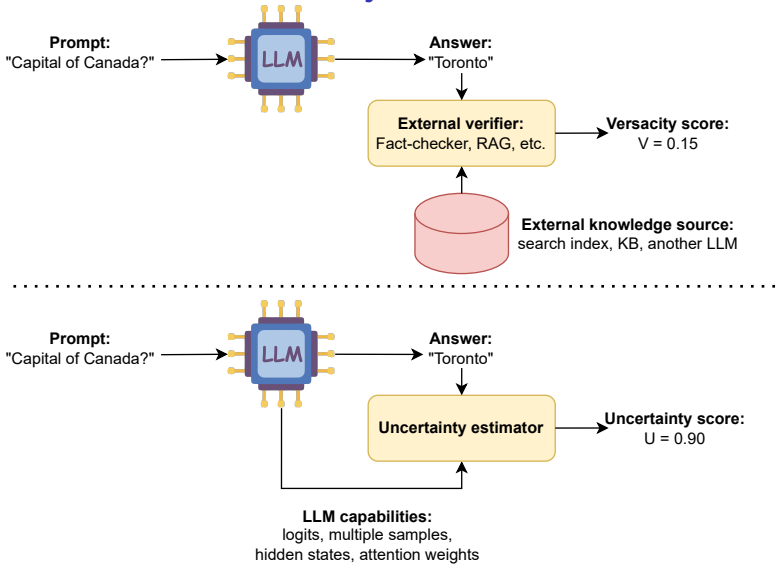
$$U_{\text{pred}} :\triangleq U_{\text{epistemic}} + U_{\text{aleatoric}}$$

$$H(Y \mid x, D) = I(Y, \theta \mid x, D) + E_{\theta \sim p(\theta \mid D)}[H(Y \mid x, \theta)]$$

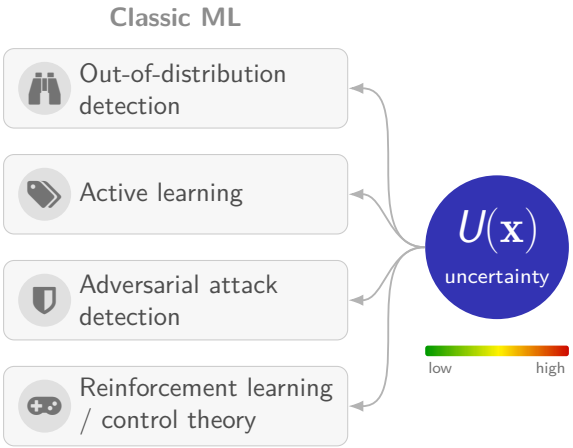
# External Verifier vs. Uncertainty Quantification



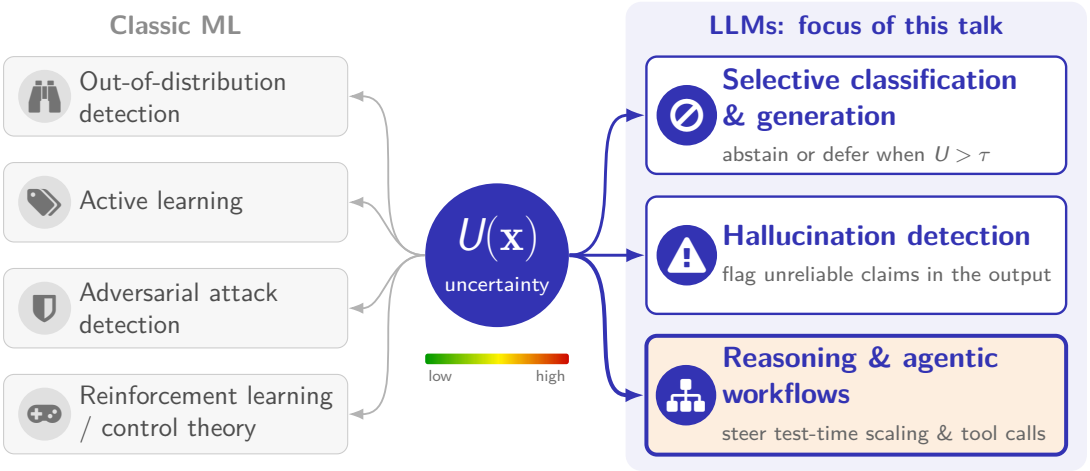
# External Verifier vs. Uncertainty Quantification



# Applications of Uncertainty



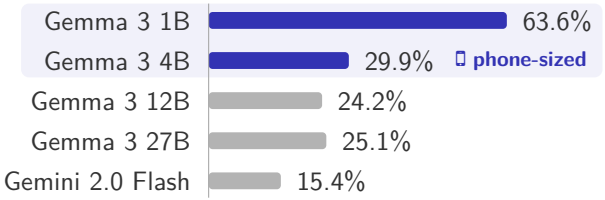
# Applications of Uncertainty



# 2. Uncertainty Quantification & Hallucination Detection

# LLMs Hallucinate, Small LLMs Particularly so[1]

**Answers not fully supported by the given document**







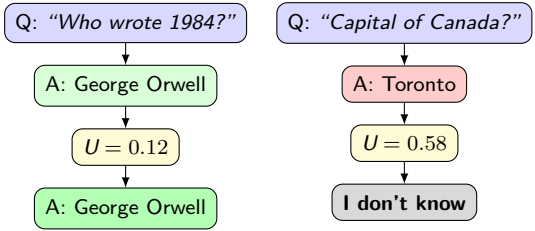
Dec 2024: Apple's ∼3B on-device model turned a BBC News alert into this false claim; Apple then paused news summaries.

[1] Gemma Team (2025). "Gemma 3 Technical Report". In: *arXiv preprint arXiv:2503.19786*; Liam Reilly (2024). "Apple urged to remove new AI feature after falsely summarizing news reports". In: *CNN*.

# Reliability Task: Selective Generation

- **Goal:** Deliver answers only when the LLM is confident
- **Selective rule:**

$$\text{output} = \begin{cases} \text{answer}, & \mathrm{U}(\mathbf{x}, \mathbf{y}) < \tau, \\ \text{“I don't know”}, & \mathrm{U}(\mathbf{x}, \mathbf{y}) \geq \tau. \end{cases}$$

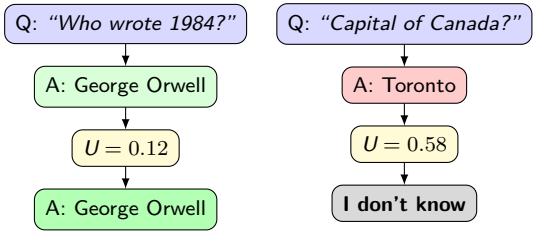


# Reliability Task: Selective Generation

- **Goal:** Deliver answers only when the LLM is confident
- **Selective rule:**

$$\text{output} = \begin{cases} \text{answer}, & \mathrm{U}(\mathbf{x}, \mathbf{y}) < \tau, \\ \text{“I don't know”}, & \mathrm{U}(\mathbf{x}, \mathbf{y}) \geq \tau. \end{cases}$$

- By **rejecting** uncertain answers, we increase performance for remaining answers



# Reliability Task: Claim-level Hallucination Detection



Give me a bio of Madonna.




Madonna is an American singer, songwriter, actress, and businesswoman. She was born on August 16, 1958, in Bay City, Michigan, and is best known for her provocative image and music. Madonna has sold over 300 million records worldwide, making her one of the best-selling music artists of all time. She has won numerous awards, including 20 Grammy Awards, a Golden Globe Award, and an Emmy Award. Madonna is also known for her philanthropic work and her activism on various social issues.

If $\mathrm{U}(\mathbf{x}, \mathbf{y}, c) \geq \tau$, highlight claim $c \in C(\mathbf{y})$ of LLM as potentially erroneous

# Sample Baseline Approaches to Uncertainty Quantification

- **Sequence probability** (log-probability) is the most straightforward measure of uncertainty for a response $\mathbf{y}^*$:

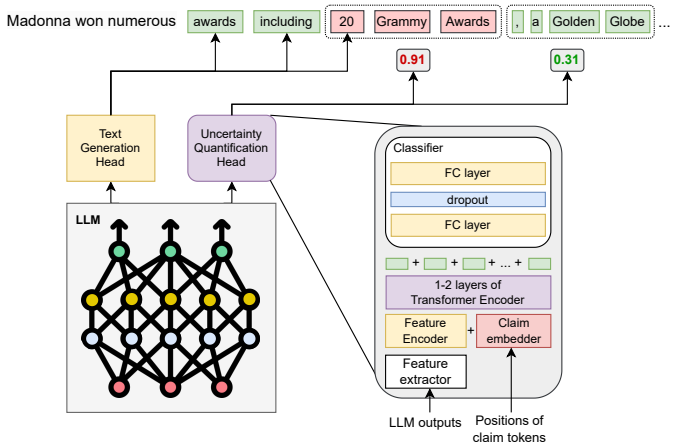
$$\mathrm{U}_{\mathrm{SP}}(\mathbf{y}^*, \mathbf{x}) = -\log P(\mathbf{y}^* \mid \mathbf{x}) = -\sum_{j=1}^{L} \log P(y_j^* \mid \mathbf{y}_{<j}^*, \mathbf{x}).$$

- Use **sampling** to estimate the bias in model outputs, e.g. via beam search over a set of sequences $\mathcal{B}$:

$$\mathbf{y}^i \in \mathcal{B}, \qquad \pi_i = \frac{P(\mathbf{y}^i \mid \mathbf{x})}{\sum_{j=1}^{N} P(\mathbf{y}^j \mid \mathbf{x})}.$$

- **Verbalised uncertainty**:
  - Have the model generate the numeric uncertainty directly
  - Examine the probability of the tokens corresponding to True/False when asked if the given response is correct (white-box)

# Idea 1: Train Attention Heads to Quantify Uncertainty[2]



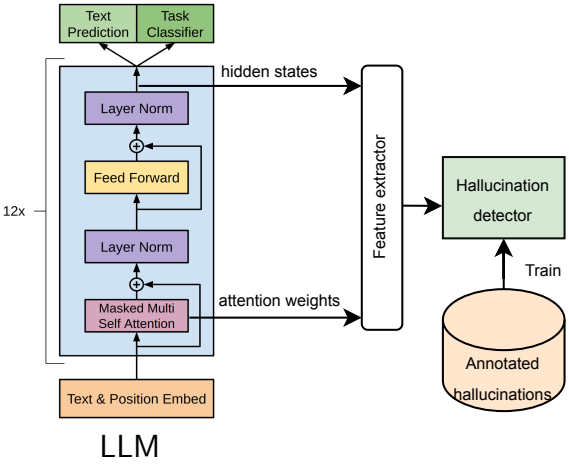




GitHub:

[2] Artem Shelmanov et al. (2025). "A Head to Predict and a Head to Question: Pre-trained Uncertainty Quantification Heads for Hallucination Detection in LLM Outputs". In: *EMNLP 2025*.

# From Manual Engineering to Data-driven Methods



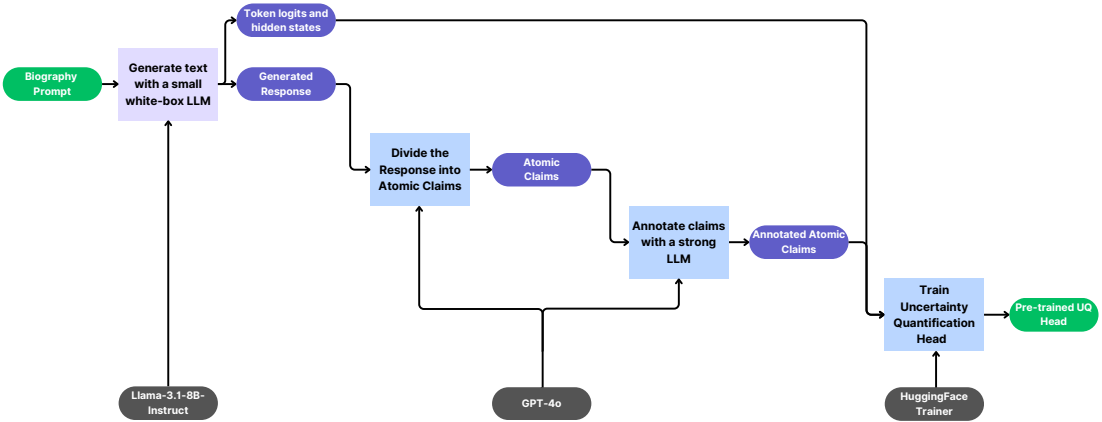

**Intuition:** Let's bypass all UQ challenges by learning from the data:

1. create features from internal states of the LLM
2. collect training data: hallucinations / non-hallucinations
3. train a supervised hallucination detector

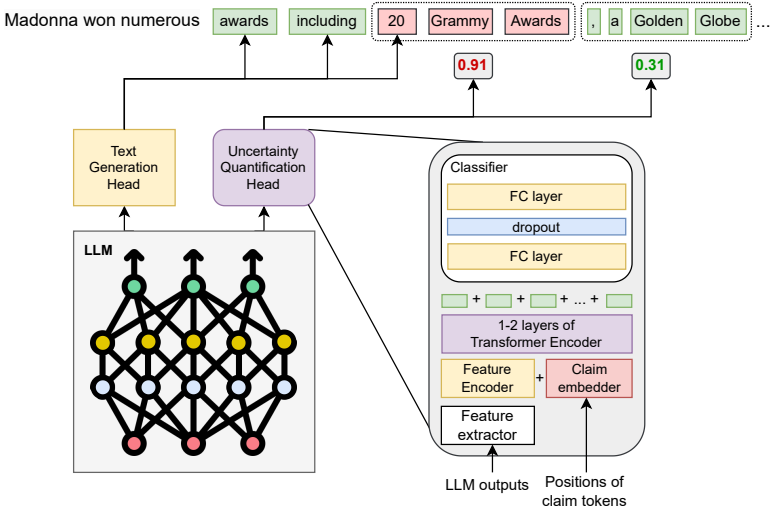
Attention weights are good features, presumably because:

- attention weights contain distinctive patterns of LLM hallucinations
- attention encodes conditional dependency between generation steps

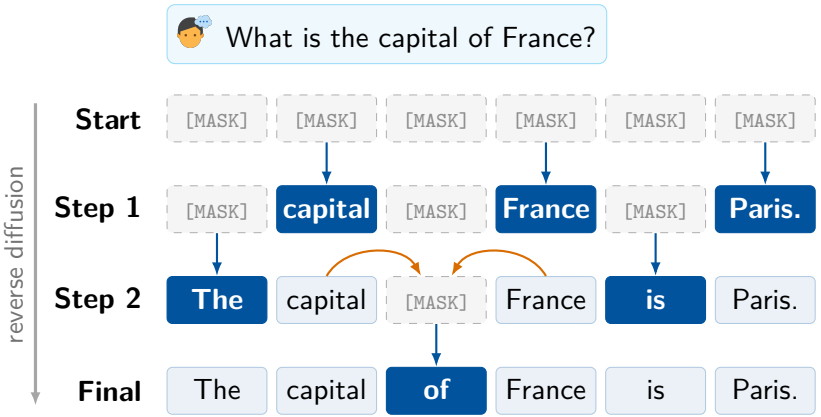
# Annotating Data for Pre-training UQ Heads



# Uncertainty Quantification Heads: Architecture



# Idea 2: UQ for Diffusion-based LLMs

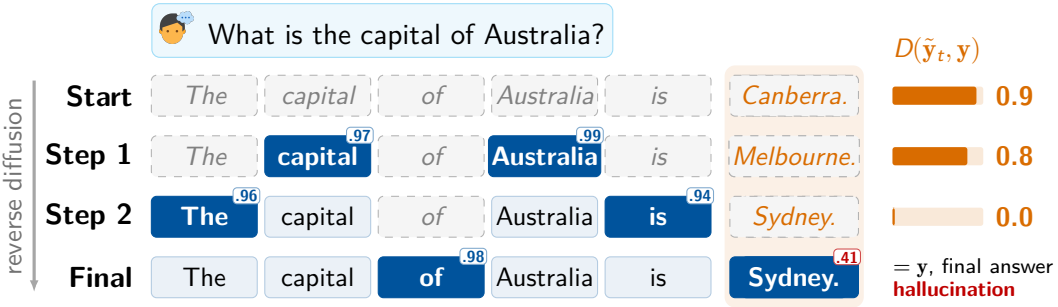



Unmask several positions **in parallel**

Predict from context on **both sides**

Repeat until no `[MASK]` is left

# D-CoCoA: Uncertainty from Reverse Diffusion[3]




$$u_{\text{D-CoCoA}} = \underbrace{u_{\text{D-inf}}}_{\text{diffusion likelihood}} \times \underbrace{\frac{1}{T}\sum_t D(\tilde{\mathbf{y}}_t, \mathbf{y})}_{\text{trajectory instability}}$$

**No resampling:** intermediate guesses replace sampled answers $\Rightarrow$ up to **100**$\times$ **cheaper** UQ

[3] Artem Vazhentsev et al. (2026). "Uncertainty Quantification for Large Language Diffusion Models". In: *NeurIPS 2026*.

# LM-Polygraph: Uncertainty Quantification Framework for LLMs

- Python library: **40+ SOTA UQ methods** with unified program API
- Built-in routines for **normalisation/calibration**
- Works with both **textual & visual** LLMs
- Integrated with
  - open-weight LLMs from **HuggingFace**: Llama, Gemma, Qwen, DeepSeek, etc.
  - models deployed via **vLLM**
  - **LLMs as a service**: ChatGPT, Gemini, OpenRouter, etc.
- **Extendable benchmark** covering MT, QA, summarisation, language understanding, fact-checking, reasoning, ...



GitHub:

# 3. Uncertainty for Reasoning

# What is Reasoning?

- LLMs solve complex tasks by generating chain-of-thought (CoT) reasoning traces
- Given an input question $\mathbf{x}$, the model generates a sequence of reasoning steps

$$\mathbf{r} = \{r_1, r_2, \ldots, r_T\}$$

followed by a final answer

$$y = g(\mathbf{r}, \mathbf{x}).$$

- Formally, an LLM parameterised by $\theta$ models the joint distribution:

$$P_\theta(y, \mathbf{r} \mid \mathbf{x}) = \prod_{t=1}^{T} P_\theta(r_t \mid \mathbf{x}, \mathbf{r}_{<t}) \cdot P_\theta(y \mid \mathbf{x}, \mathbf{r})$$

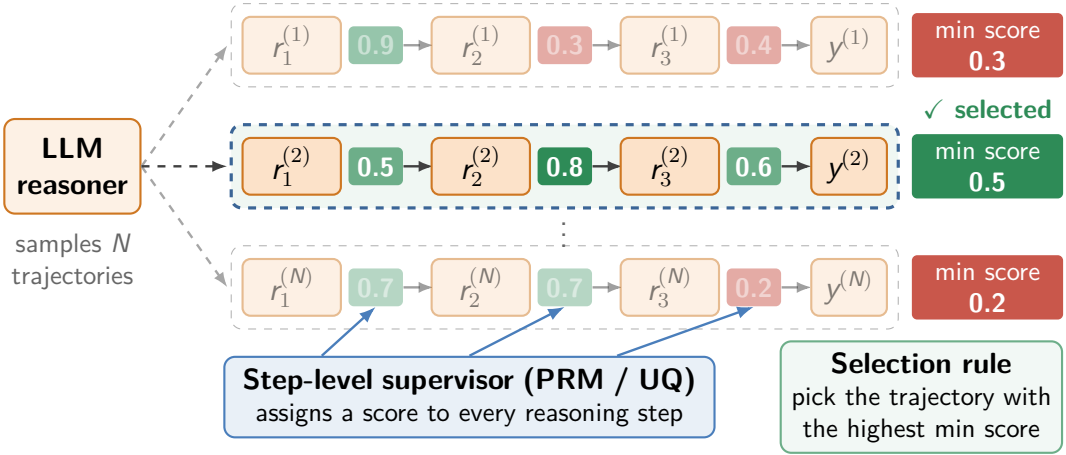
# Example: Chain of Thought of a Thinking Model

**Question:** A warehouse must ship 25,000 cans. A truck carries 24 boxes; each box holds 15 packs of 8 cans. How many trucks are needed?

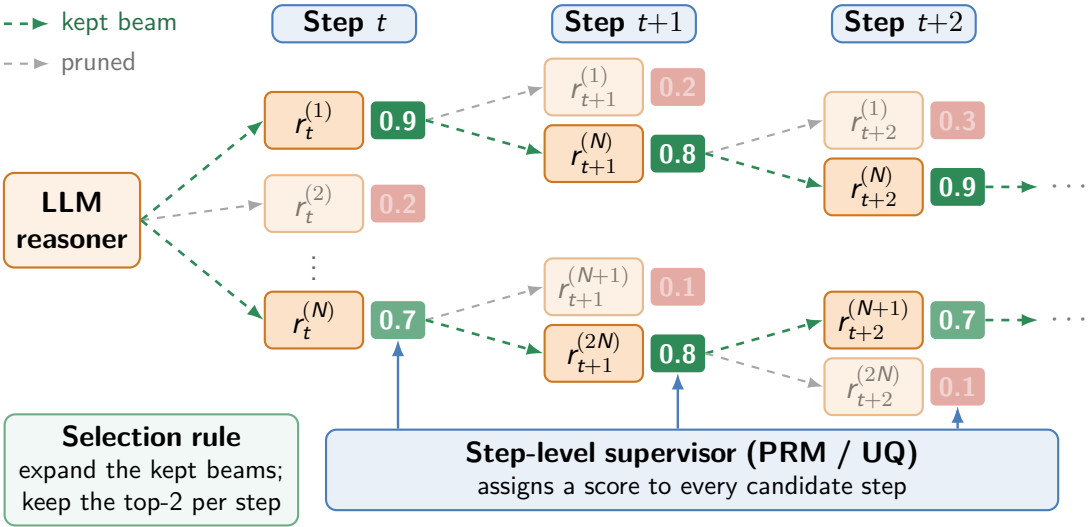
<think>
**[plan]** First find how many cans fit in one truck, then divide.
**[compute]** One box: $15 \times 8 = 120$ cans. One truck: $24 \times 120 = 2{,}880$ cans.
**[compute]** $25{,}000 \div 2{,}880 \approx 8.68$, so 8 trucks carry $8 \times 2{,}880 = 23{,}040$ cans.
**[self-check]** Wait – that leaves $1{,}960$ cans unshipped. The question asks how many trucks are *needed*, so I must round *up*, not down: 9 trucks.
**[verify]** Check: $9 \times 2{,}880 = 25{,}920 \geq 25{,}000$. ✓
</think>

**Answer:** 9 trucks.

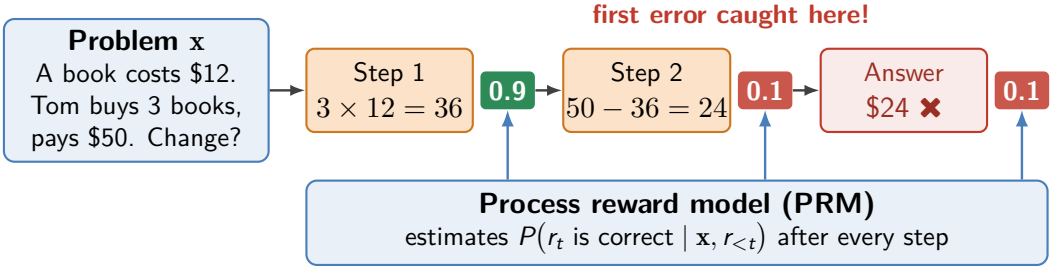
# Test Time Compute (TTC) Scaling: Best-of-$N$



# TTC Scaling: Beam Search over Reasoning Steps



# Process Reward Models (PRMs) for Scoring Reasoning Steps




**Large critique model**
A separate 1.5–8B LLM that judges every step

**Tuned for one task**
Fine-tuned on step labels for one domain, e.g., math

**Guides search**
Its step scores drive Best-of-$N$ and beam search

# Idea 3: Uncertainty as a Reasoning Step Scorer for Search

- Use uncertainty to score the quality of reasoning steps and entire trajectories
- Scorer for a reasoning step is confidence:

$$Q(r_t) = C(r_t) = 1 - U(r_t)$$

- Scorer for the entire reasoning trajectory is aggregation of step-level scores (e.g. average):

$$Q(r_{1:T}) = \frac{1}{T} \sum_{t=1}^{T} C(r_t)$$

# Prior Methods for Reasoning Step Supervision

**Process Reward Models (PRMs)**

- ✔ Strong step-level signal
- ✖ Not efficient: you deploy *another* LLM (1.5–8B) at every step
- ✖ Heavily optimised (overfit) on certain domains (math, code)

**Uncertainty quantification (UQ)**

- ✔ Cheap
- ✔ No overfitting
- ✖ But weak

# Prior Methods for Reasoning Step Supervision

**Process Reward Models (PRMs)**

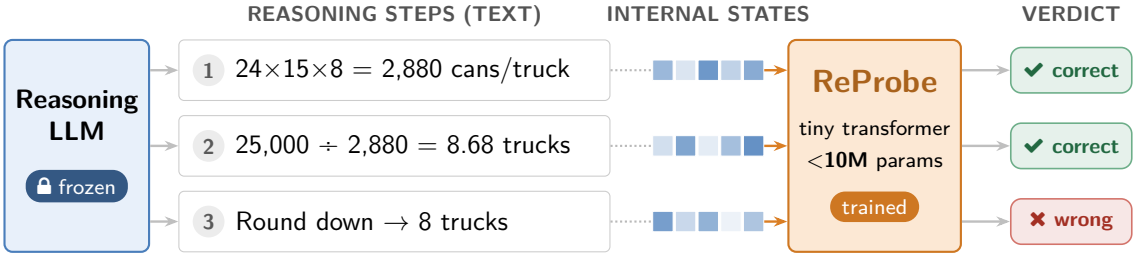
- ✔ Strong step-level signal
- ✖ Not efficient: you deploy *another* LLM (1.5–8B) at every step
- ✖ Heavily optimised (overfit) on certain domains (math, code)

**Uncertainty quantification (UQ)**

- ✔ Cheap
- ✔ No overfitting
- ✖ But weak

**Can we incorporate ideas from PRMs into UQ to get a cheap but strong model?**

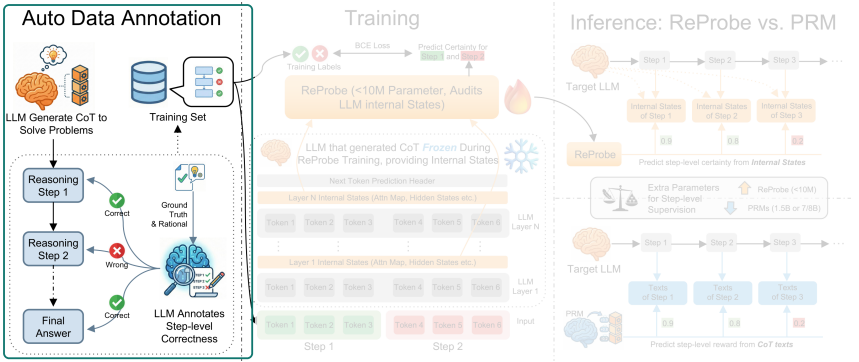
# ReProbe[4]




By contrast, a **PRM** is a separate **large critique model** (1.5–8B params), **fine-tuned for a particular task** (e.g., math or code), that re-reads every step

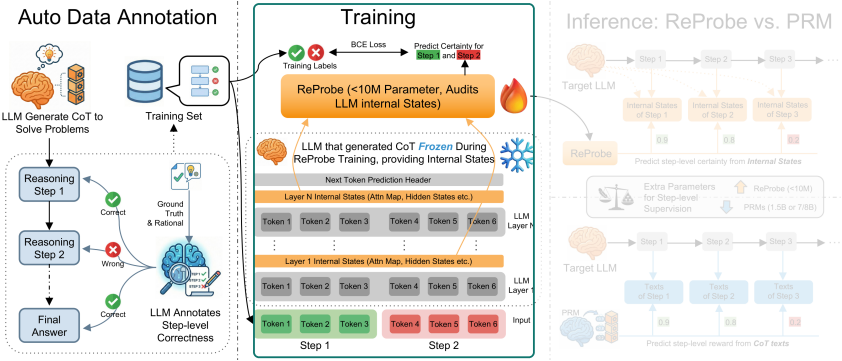
[4] Jingwei Ni et al. (2026a). "ReProbe: Efficient Test-Time Scaling of Multi-Step Reasoning by Probing Internal States of Large Language Models". In: *ACL 2026*.

# ReProbe Pipeline: 1/3 – Auto-annotate Step Correctness



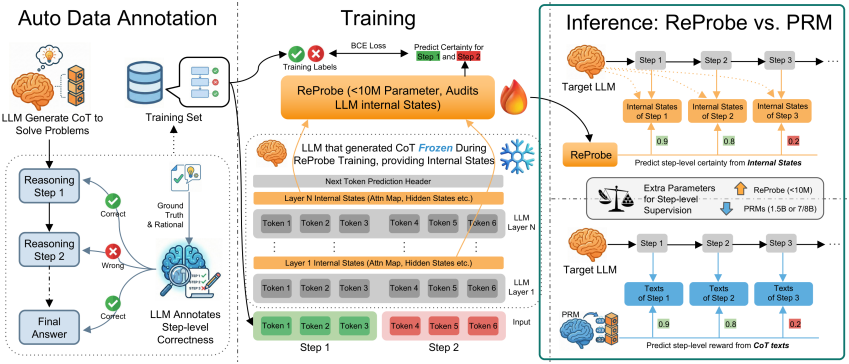

The LLM writes chain-of-thought solutions; each step is auto-labeled correct / incorrect – by the model itself or by a strong LLM – against the gold answer. **No human labels.**

# ReProbe Pipeline: 2/3 – Train a Tiny Probe on Internal States




The base LLM stays **frozen**. For each step, ReProbe reads its internal states – hidden states, attention, logits – and a <10M-parameter transformer predicts step certainty, trained with BCE

# ReProbe Pipeline: 3/3 – Score Steps at Inference (vs. PRM)




At test time, ReProbe reads the target LLM's internal states to score each step **in one pass** – no separate 1.5–7B PRM re-reading the generated text.

# PRMs vs. ReProbe

**PRM** PROCESS REWARD MODEL

**1.5–8B** params, on every step

A second large model re-reads the text

✔ **Strong** – accurate on math, in-domain

✖ **Costly** – billions of params re-run at every reasoning step

✖ **Poor generalisation** – trained on math → brittle on planning & QA

**vs.**

**ReProbe** INTERNAL-STATE PROBE

$<$**10M** probe params, frozen base

$\sim$800$\times$ smaller – reads states it already computes.

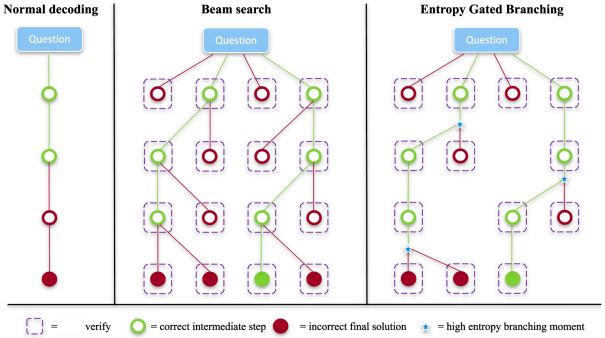
✔ **Accurate** – competitive in-domain, pulls ahead out of domain

✔ **Cheap** – 25$\times$ faster; $\sim$4 GPU-hours to train

✔ **General** – reads internal states → holds up out of domain

Comparable results – accurate, general, cheap – at a thousandth of the size.

# Idea 4: Use Uncertainty to Selectively Expand Search[5]

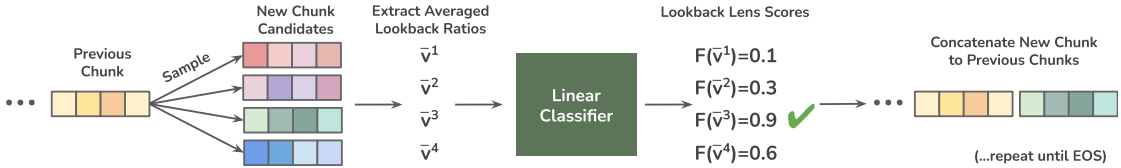



- **Low-uncertainty step** ⇒ continue the default reasoning trajectory *(no test-time scaling)*.
- **High-uncertainty step** ⇒ trigger branching to explore alternatives *(enable test-time scaling)*.
- **(Optional) Stop expanding** highly uncertainty trajectories

[5] Xianzhi Li et al. (2026). "Entropy-Gated Branching for Efficient Test-Time Reasoning". In: *EACL 2026*.

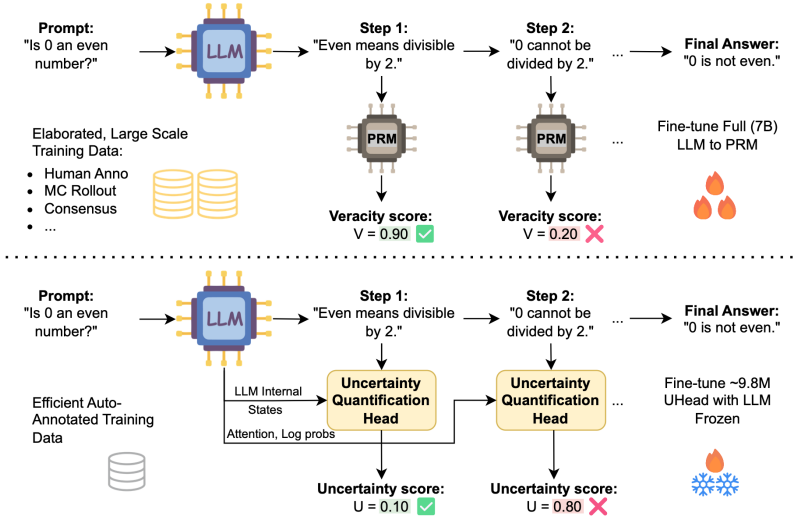
# Intervening in the Decoding Process

**Idea:** modify the decoding process using confidence, e.g. sample multiple continuations and choose the one with the highest confidence[6]




[6] Yung-Sung Chuang et al. (2024). "Lookback Lens: Detecting and Mitigating Contextual Hallucinations in Large Language Models Using Only Attention Maps". In: *EMNLP 2024*.

# Verification of Reasoning Steps via Uncertainty Heads



# ThinkBooster[7] – A Unified Framework for TTC Scaling

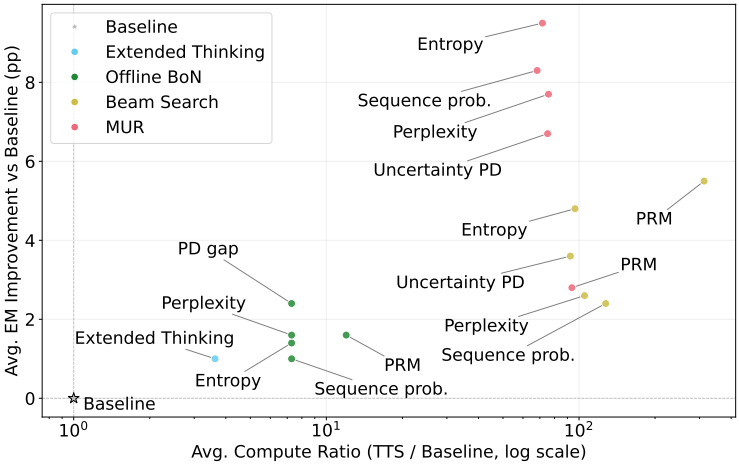
- Python library that implements:
  - SOTA TTC scaling strategies: tree of thought, online best-of-$n$, majority voting, MUR, etc.
  - SOTA reasoning scorers: PRMs, confidence scores, ReProbes, LLM as a critic, etc.
  - Access to local LLMs, vLLM, OpenRouter, ChatGPT, Together.ai, etc.
- A benchmark for evaluating the performance–efficiency trade off for TTC scaling methods on reasoning-intensive tasks
- A transparent TTC scaling endpoint with OpenAI compatible API




GitHub:

[7] Vladislav Smirnov et al. (2026). "ThinkBooster: A Unified Framework for Seamless Test-Time Scaling of LLM Reasoning". In: *ACL 2026, System Demonstrations*.

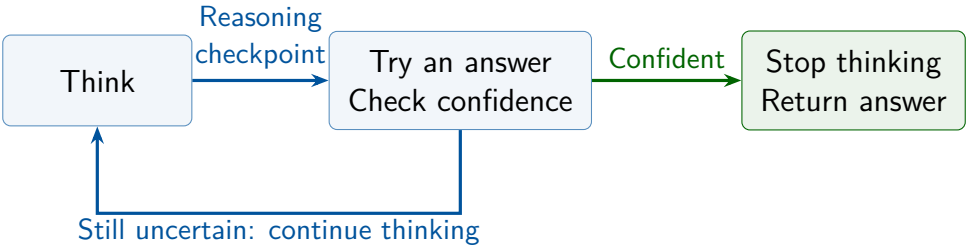
# ThinkBooster: Sample Experimental Results




Uncertainty scores can provide better performance–efficiency trade-offs than PRMs

# Idea 5: Early Exit during Thinking in Reasoning LLMs[8]

**Idea:** Check whether the model can already answer before thinking further




[8] Chenxu Yang et al. (2026). "Dynamic Early Exit in Reasoning Models". In: *ICLR 2026*.

# Impact on Compute Reduction and Performance in Native Thinking

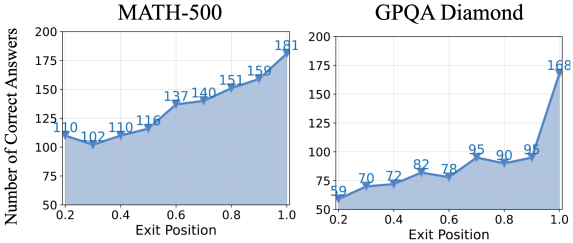



Figure: Number of answers that stay correct with early exit (illustration in DEER)

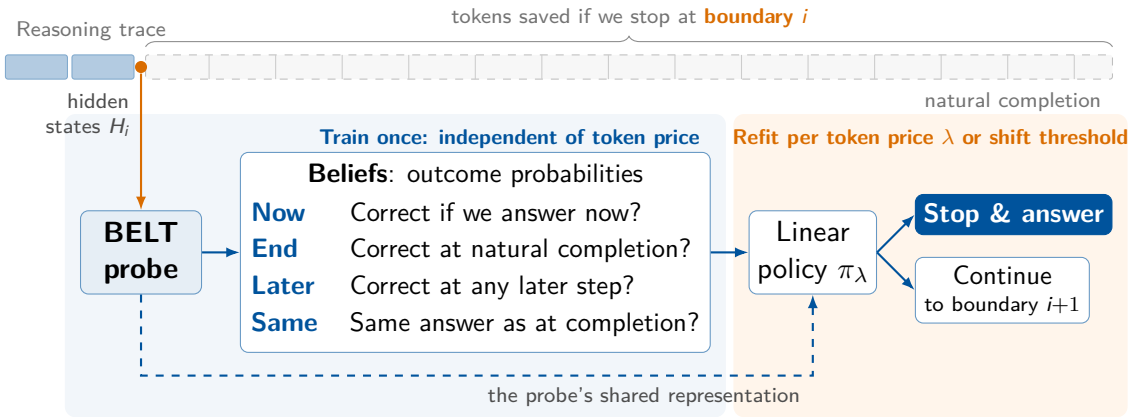
- DEER[9] ↓ thinking length by 19.1–80.1% while ↑ accuracy by 0.3–5.0%
- Certaindex[10] ↓ compute by up to 50%

[9] Chenxu Yang et al. (2026). "Dynamic Early Exit in Reasoning Models". In: *ICLR 2026*.

[10] Yichao Fu et al. (2025). "Efficiently Scaling LLM Reasoning Programs with Certaindex". In: *NeurIPS 2025*.

# BELT: Learning When to Stop[11]

**Idea:** Predict reasoning outcomes from the frozen LLM's hidden states




[11] Jingwei Ni et al. (2026b). "Same Beliefs, Different Policies: Learning Reusable Representations for Early Stopping in LLM Reasoning". In: *arXiv preprint arXiv:2610.30450*.

# BELT: Reducing Reasoning Cost

**Fewer generated tokens at matched natural-completion accuracy**

**41%**
Qwen3-1.7B

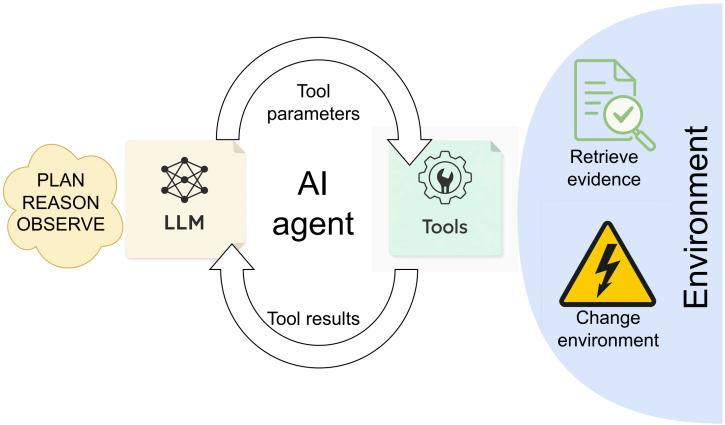
**51%**
OLMo-3-7B-Think

BELT-H-T; average over five out-of-distribution math benchmarks.

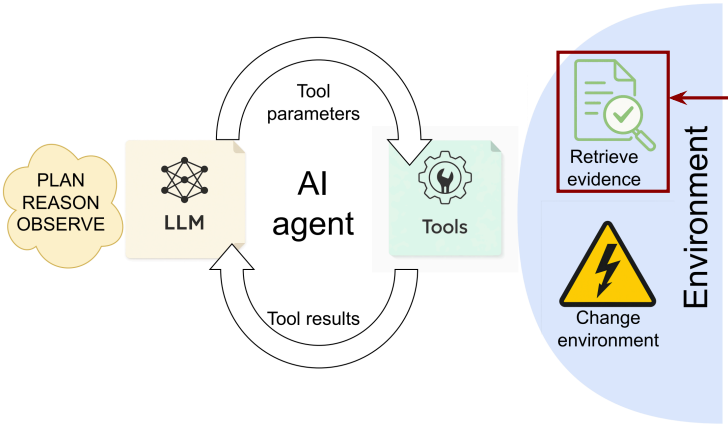
- **Cheaper adaptation:** about 55× fewer training GPU-hours than five price-specific OS-Pruner controllers (OLMo3)
- **Transfer beyond math:** math-trained probes also control thinking in coding (LiveCodeBench) and agent tasks (ALFWorld, ScienceWorld)

# 4. Uncertainty for AI Agents

# Uncertainty for Tool Use

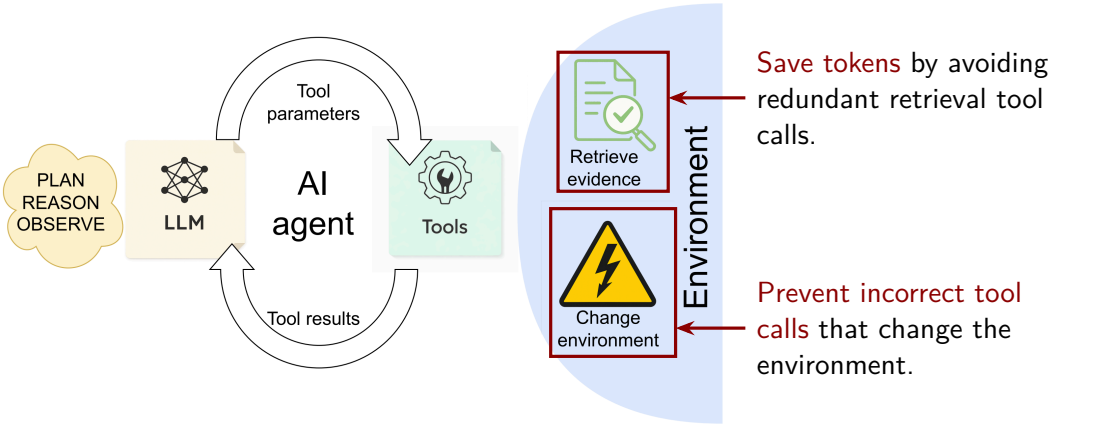


# Uncertainty for Tool Use




Save tokens by avoiding redundant retrieval tool calls.

# Uncertainty for Tool Use



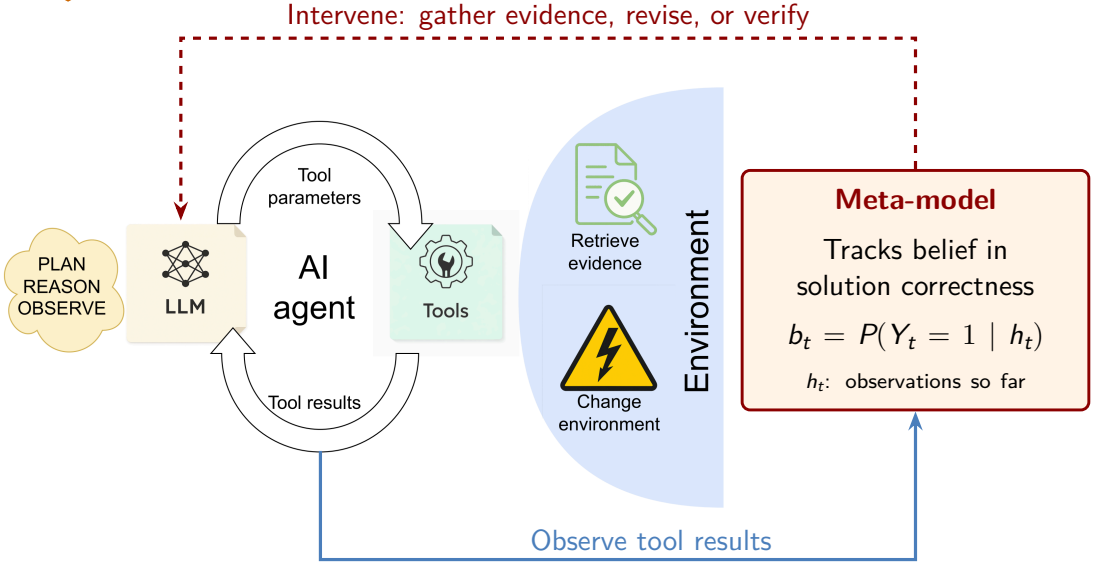
# Idea 6: Use Larger Models Only When Necessary[12]

**Core idea** – for agentic applications, use a small, cheap reasoning model by default, and a large, more capable (but expensive) model only when the small model is uncertain (reason–defer–act = "ReDAct")

- Generate a next step with small reasoning model $\mathcal{M}_{\text{small}}$
- Measure answer uncertainty $u$; **keep** if $u \leq \tau$, else **defer** to larger model $\mathcal{M}_{\text{large}}$
- Based on experiments over ALFWorld and MiniGrid:
  - perplexity works best as the UQ method
  - the method defers to $\mathcal{M}_{\text{large}}$ only $\sim$15% of the time but results in an improved success rate of $\sim$12pp (compared to $\sim$13pp using $\mathcal{M}_{\text{large}}$ only) with $\sim$66% greater computational cost (compared to $>$400% using $\mathcal{M}_{\text{large}}$ only)

[12] Dzianis Piatrashyn et al. (2026). "ReDAct: Uncertainty-Aware Deferral for LLM Agents". In: *arXiv:2604.07036*.

# Idea 7: Meta-model for Correctness Tracking and Control

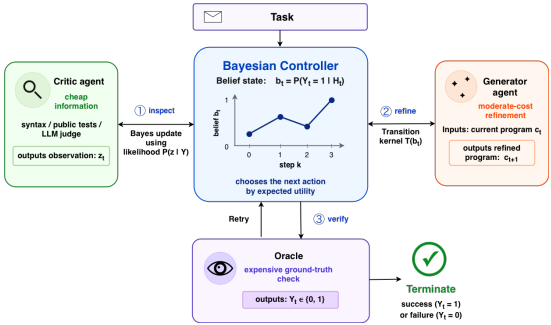


# Bayesian Control as Sequential Hypothesis Testing[13]

**Idea:** orchestration as **cost-sensitive sequential hypothesis testing** over latent candidate correctness $Y \in \{0, 1\}$:

- maintain a belief $b_t = P(Y_t{=}1 \mid \text{evidence})$
- critic calls $\rightarrow$ noisy **observations**
- generator calls $\rightarrow$ stochastic **transitions**
- verifier call $\rightarrow$ costly **terminal action**

The controller acts by **value of information** on *frozen* LLMs (generators are never trained).




Reward $R$ for a correct solution minus accumulated tool costs.

[13] Theodore Papamarkou et al. (2026b). "Position: Agentic AI Orchestration Should Be Bayes-Consistent". In: *ICML 2026*; Theodore Papamarkou et al. (2026a). "Bayesian Control for Coding Agents". In: *EMNLP 2026*.

# Two Belief-state Controllers from the Bellman Equation

**Belief updates.** (1) A critic observation $z$ is used by **Bayes' rule**:

$$b_{t+1} = \frac{P_i(z \mid Y{=}1)\, b_t}{P_i(z \mid Y{=}1)\, b_t + P_i(z \mid Y{=}0)(1 - b_t)}$$

(2) a generator call is propagated through a measured **transition kernel** (fix prob. $p_{01}$, break prob. $p_{10}$):

$$b_{t+1} = b_t(1 - p_{10}) + (1 - b_t)\, p_{01}$$

**Bellman optimality** picks the action with the largest $Q$-value:

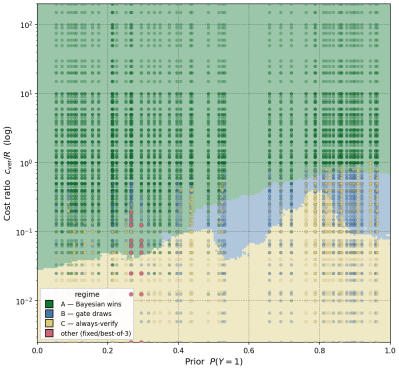
$$V(b_t) = \max\Big( \underbrace{Q_{\text{gen}}(b_t)}_{\text{refine}},\ \underbrace{\max_i Q^i_{\text{crit}}(b_t)}_{\text{buy evidence}},\ \underbrace{Q_{\text{ver}}(b_t) = b_t R - C_{\text{ver}}}_{\text{stop \& verify}} \Big)$$

- `bayesian_greedy` – one-step lookahead, assumes the next move is *verify*
- `bayesian_DP` – finite-horizon backward induction over $(b, h)$; the transition kernel lets it value multi-step *critic* $\rightarrow$ *refine* $\rightarrow$ *verify* chains the greedy policy cannot see

# When does Bayesian Control Help? Three Regimes

Empirical decision map over the $\big(P(Y{=}1),\ C_{\text{ver}}/R\big)$ plane: 7,020 sweep points pooled across 6 generators $\times$ 9 coding benchmarks.

| Regime | Winner | Why |
|---|---|---|
| **A** | **Bayesian** | verification costly ($C_{\text{ver}}/R \gtrsim 1$); aggregates many imperfect critics, downweights weak ones |
| **B** | `gate(test)` | one near-oracle public-test critic; Bayesian aggregation adds little |
| **C** | `always_verify` | verification cheap, or prior already high; info not worth its cost |




**Sweet spot:** noisy LLM generations (low-prior tasks), costly verifier, informative but imperfect critics.

# 5. Trust

# Trust of and in LLMs

- One important aspect I haven't yet mentioned is **trust**, which has two related components:
  1. trust **of** the LLM in noisy inputs/agents to be more resilient against noise

     *training regimens to encourage more robust behaviours*

  2. trust **in** the LLM on the part of users/organisation

     *how to make LLMs more controllable, interpretable, scrutable, value-aligned, ethical, ... in how they operate*

# Trust of and in LLMs

- One important aspect I haven't yet mentioned is **trust**, which has two related components:
  1. trust **of** the LLM in noisy inputs/agents to be more resilient against noise

     *training regimens to encourage more robust behaviours*

  2. trust **in** the LLM on the part of users/organisation

     *how to make LLMs more controllable, interpretable, scrutable, value-aligned, ethical, ... in how they operate*

     Reliable estimates of uncertainty are a critical piece of this much larger puzzle

# Idea 8: Teaching LLMs to (Mis)trust

- **Training environment**: dynamic simulated multi-agent environment where some agents are (noisily) trustworthy and others are adversaries, and where partial information about an outcome is incrementally elicited
  - for training purposes, we have access to knowledge of both the reliability of individual agents and the probability distribution over the solution space at any given point $\longrightarrow$ possible to extract exact posteriors at any point
- When models are trained with exact posterior distillation, they generalise well to OOD test scenarios where there are conflicts in knowledge sources
- Also effective as a pre-training objective for decision models

# 6. Wrapping Up

# Take-home Messages

- LLMs are always limited by limited training data and task ambiguity
- Uncertainty quantification can be a crucial component of modern-day LLMs, especially in complex operating environments with many moving pieces, such as reasoning and agentic applications
- Uncertainty not only about safety, and can also improve performance and efficiency of reasoning and AI agents
- LM-Polygraph and uncertainty heads allow you to easily integrate uncertainty into your favourite model/task
- Much more work to be done on UQ for different language tasks and model types, and to adapt methods developed for UQ to other problems

# Thank you!

Bayesian agent:



ThinkBooster:



LM-Polygraph:



# References I


- Chuang, Yung-Sung et al. (2024). "Lookback Lens: Detecting and Mitigating Contextual Hallucinations in Large Language Models Using Only Attention Maps". In: *EMNLP 2024*.
- Fu, Yichao et al. (2025). "Efficiently Scaling LLM Reasoning Programs with Certaindex". In: *NeurIPS 2025*.
- Gemma Team (2025). "Gemma 3 Technical Report". In: *arXiv preprint arXiv:2503.19786*.
- Li, Xianzhi et al. (2026). "Entropy-Gated Branching for Efficient Test-Time Reasoning". In: *EACL 2026*.
- Ni, Jingwei et al. (2026a). "ReProbe: Efficient Test-Time Scaling of Multi-Step Reasoning by Probing Internal States of Large Language Models". In: *ACL 2026*.
- Ni, Jingwei et al. (2026b). "Same Beliefs, Different Policies: Learning Reusable Representations for Early Stopping in LLM Reasoning". In: *arXiv preprint arXiv:2610.30450*.

# References II


- Papamarkou, Theodore et al. (2026a). "Bayesian Control for Coding Agents". In: *EMNLP 2026*.
- Papamarkou, Theodore et al. (2026b). "Position: Agentic AI Orchestration Should Be Bayes-Consistent". In: *ICML 2026*.
- Piatrashyn, Dzianis et al. (2026). "ReDAct: Uncertainty-Aware Deferral for LLM Agents". In: *arXiv:2604.07036*.
- Reilly, Liam (2024). "Apple urged to remove new AI feature after falsely summarizing news reports". In: *CNN*.
- Shelmanov, Artem et al. (2025). "A Head to Predict and a Head to Question: Pre-trained Uncertainty Quantification Heads for Hallucination Detection in LLM Outputs". In: *EMNLP 2025*.
- Smirnov, Vladislav et al. (2026). "ThinkBooster: A Unified Framework for Seamless Test-Time Scaling of LLM Reasoning". In: *ACL 2026, System Demonstrations*.

# References III


- Vazhentsev, Artem et al. (2026). “Uncertainty Quantification for Large Language Diffusion Models”. In: *NeurIPS 2026*.
- Yang, Chenxu et al. (2026). “Dynamic Early Exit in Reasoning Models”. In: *ICLR 2026*.