

CS329Z: Engineering AI Agents

Optimization

Diyi Yang, Michael Ryan, John Yang

Announcements

- Guest Lecture next Monday!
- Homework 2 releases next Monday.
- Homework 1 due next Friday, Midway report is the following week.

Quick Poll: Which Prompt is best for PaLM 2-L-IT

A	[Math Problem] Let's think step by step.	71.8%
B	[Math Problem] Take a deep breath and work on this problem step-by-step.	**80.2%**
C	[Math Problem] Let's combine our numerical command and clear thinking to quickly and accurately decipher the answer.	74.5%

Optimization enables us to specify intent through **data and evals** and improve our system.

Today we will break down **common optimization algorithms** and **when to use them**.

Learning Objectives

- **Identify** when to use optimization for AI systems.
- **Describe** various common context/weight optimization algorithms.
- **Differentiate** times to use each optimization strategy.

Lecture Overview

- Conditions for Optimization (5 minutes)
- Context Optimization (25 minutes)
- Weight Optimization (25 minutes)
- Optimization with Agents (10 minutes)
- Which Strategy to Use (10 minutes)

Conditions for Optimization

Three Pillars of AI Systems

Data

- Training Data
- Rollouts
- Demonstrations
- Human Feedback
- …

System or Model

- Language Model
- Compound AI System / Workflow
- Agentic System
- …

Evaluation + Metric

- Pass Rate
- Exact Match
- LLM Judge
- Cost/Latency
- …

To optimize an AI System you need all three!

Is (Context) Optimization Necessary?

- Do we really need to fiddle with prompts anymore?
- Aren't models going to just get better with scale?



- Optimization is useful for **LAST MILE** adaptation.
  - Once we know your exact setting we can adapt to it.
  - Models offer a great general solution, we can optimize for the specifics.

Image Credit: https://msukhareva.substack.com/p/the-last-mile-problem

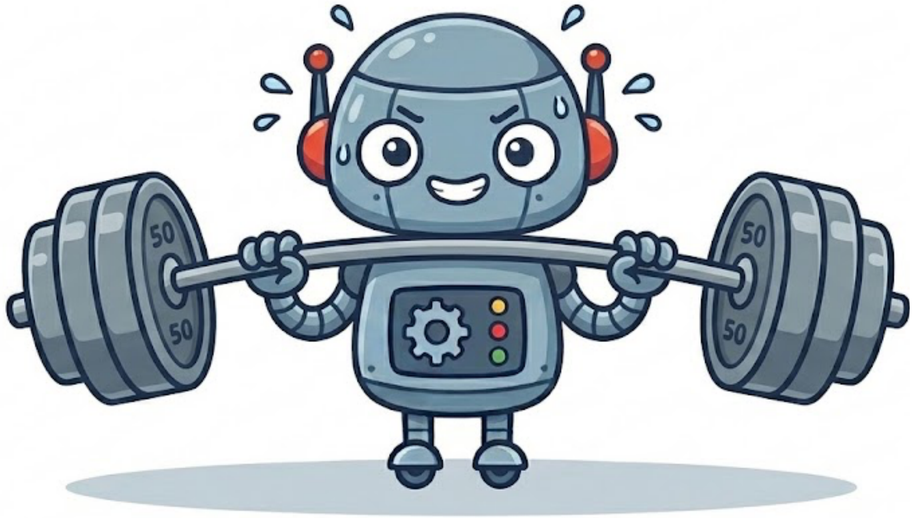
When to Optimize

- You have data and evals that suit your particular domain.
- Examples and feedback can communicate requirements that are difficult to express completely in a prompt/system
  - Sometimes proper evals convey intent better and you can tune the system to your evals.
- The system has many complex components that are difficult to tune manually.



Benefits of Optimization

- **Greater reliability**: Reduce recurring errors and improve consistency across tasks.
- **Domain adaptation**: Improve performance on specialized tasks, workflows, and requirements.
- **Lower cost and latency**: Maintain quality with smaller models, shorter prompts, or fewer tool calls.
- **Learning from experience**: Turn successful trajectories and failure feedback into lasting improvements.
- **Adaptability**: Adjust behavior as models, tools, and task requirements change.



Context Optimization

Fewshot Optimization

Fewshot Prompting

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

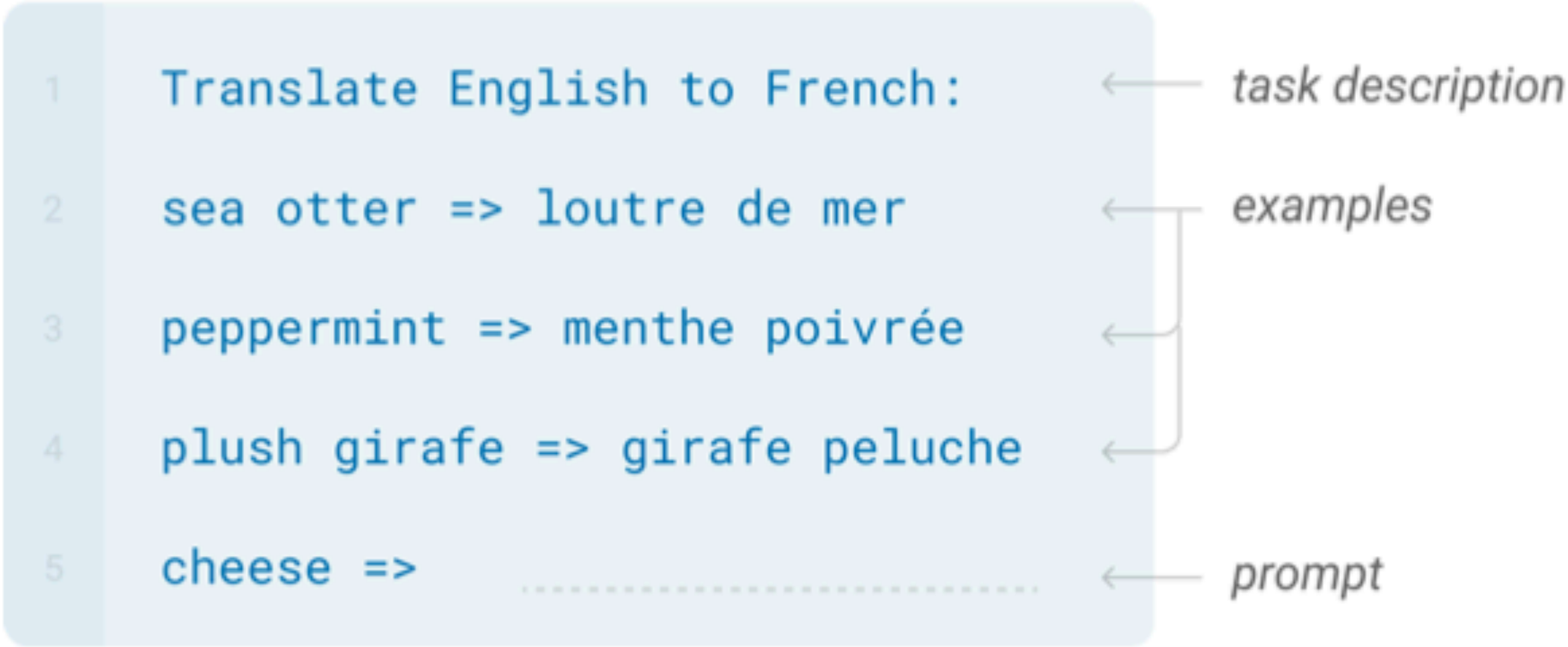
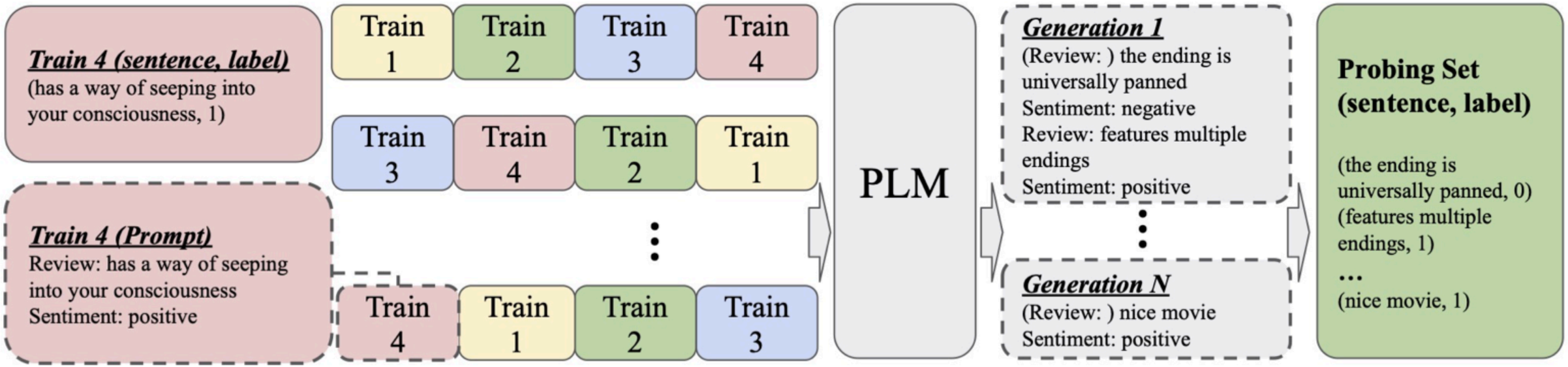


Image Credit: https://arxiv.org/pdf/2005.14165

Fantastically Ordered Prompts and Where to Find Them [Lu et al., 2022]



Try a bunch of prompt orderings on the trainset and measure how this changes labels on the probing set. Interestingly we throw ignore the ground truth labels on the probing set.

$$\text{GlobalE}_m = \sum_{v \in V} -p_m^v \log p_m^v \qquad \text{LocalE}_m = \frac{\sum_i \sum_{v \in V} -p_{i,m}^v \log p_{i,m}^v}{|D|}$$

Image Credit: https://arxiv.org/pdf/2005.14165

Fantastically Ordered Prompts and Where to Find Them [Lu et al., 2022]

High Global Entropy

the ending is universally panned	→ 0
features multiple endings	→ 1
quite hard to sit through	→ 0
nice movie	→ 1

Low Global Entropy

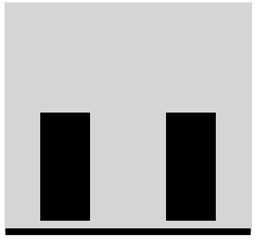
the ending is universally panned	→ 0
features multiple endings	→ 0
quite hard to sit through	→ 0
nice movie	→ 0

$$\text{GlobalE}_m = \sum_{v \in V} -p_m^v \log p_m^v$$

Fantastically Ordered Prompts and Where to Find Them [Lu et al., 2022]

High Local Entropy

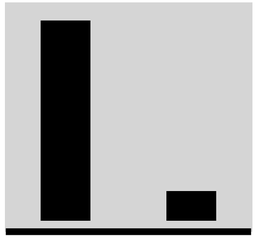
the ending is universally panned



logprobs

Low Local Entropy

the ending is universally panned



logprobs

Average over probe set

$$\text{LocalE}_m = \frac{\sum_i \sum_{v \in V} -p_{i,m}^v \log p_{i,m}^v}{|D|}$$

Fantastically Ordered Prompts and Where to Find Them [Lu et al., 2022]

	SST-2	SST-5	DBPedia	MR	CR	MPQA	Subj	TREC	AGNews	RTE	CB
Majority	50.9	23.1	9.4	50.0	50.0	50.0	50.0	18.8	25.0	52.7	51.8
Finetuning (Full)	95.0	58.7	99.3	90.8	89.4	87.8	97.0	97.4	94.7	80.9	90.5
GPT-3 2.7B	$78.0_{10.7}$	$35.3_{6.9}$	$81.1_{1.8}$	$68.0_{12.9}$	$76.8_{11.7}$	$66.5_{10.3}$	$49.1_{2.9}$	$55.3_{4.4}$	$72.9_{4.8}$	$48.6_{1.9}$	$50.4_{0.7}$
LocalE	$\mathbf{81.0}_{6.0}$	$42.3_{4.7}$	$80.3_{1.7}$	$75.6_{4.1}$	$79.0_{5.5}$	$72.5_{5.8}$	$54.2_{4.2}$	$54.0_{2.6}$	$72.3_{4.6}$	$50.4_{1.9}$	$50.5_{0.8}$
GlobalE	$80.2_{4.2}$	$\mathbf{43.2}_{4.3}$	$\mathbf{81.2}_{0.9}$	$\mathbf{76.1}_{3.8}$	$\mathbf{80.3}_{3.4}$	$\mathbf{73.0}_{4.3}$	$\mathbf{54.3}_{4.0}$	$\mathbf{56.7}_{2.0}$	$\mathbf{78.1}_{1.9}$	$\mathbf{51.3}_{1.8}$	$\mathbf{51.2}_{0.8}$
Oracle	$89.8_{0.7}$	$48.0_{1.1}$	$85.4_{1.6}$	$87.4_{0.9}$	$90.1_{0.7}$	$80.9_{1.4}$	$60.3_{10.3}$	$62.8_{4.2}$	$81.3_{2.9}$	$53.4_{3.1}$	$52.5_{1.4}$
GPT-3 175B	$\mathbf{93.9}_{0.6}$	$54.4_{2.5}$	$95.4_{0.9}$	$\mathbf{94.6}_{0.7}$	$91.0_{1.0}$	$83.2_{1.5}$	$71.2_{7.3}$	$72.1_{2.7}$	$85.1_{1.7}$	$70.8_{2.8}$	$75.1_{5.1}$
LocalE	$93.8_{0.5}$	$\mathbf{56.0}_{1.7}$	$95.5_{0.9}$	$94.5_{0.7}$	$91.3_{0.5}$	$\mathbf{83.3}_{1.7}$	$75.0_{4.6}$	$71.8_{3.2}$	$\mathbf{85.9}_{0.7}$	$\mathbf{71.9}_{1.4}$	$74.6_{4.2}$
GlobalE	$\mathbf{93.9}_{0.6}$	$53.2_{2.1}$	$\mathbf{95.7}_{0.7}$	$\mathbf{94.6}_{0.2}$	$\mathbf{91.7}_{0.4}$	$82.0_{0.8}$	$\mathbf{76.3}_{3.5}$	$\mathbf{73.6}_{2.5}$	$85.7_{1.0}$	$71.8_{1.9}$	$\mathbf{79.9}_{3.3}$
Oracle	$94.7_{0.2}$	58.2	$96.7_{0.2}$	$95.5_{0.2}$	$92.6_{0.4}$	$85.5_{0.8}$	$81.1_{4.9}$	$77.0_{1.2}$	$87.7_{0.6}$	$74.7_{0.4}$	$83.0_{0.9}$

Using the probing set and measuring entropy approaches the oracle ordering accuracy (trying every ordering to find the best on the devset)

Image Credit: https://arxiv.org/pdf/2005.14165

Bootstrap Reasoning

We know Chain of Thought Reasoning is important… but labeled examples for few shot often do not come with reasoning.

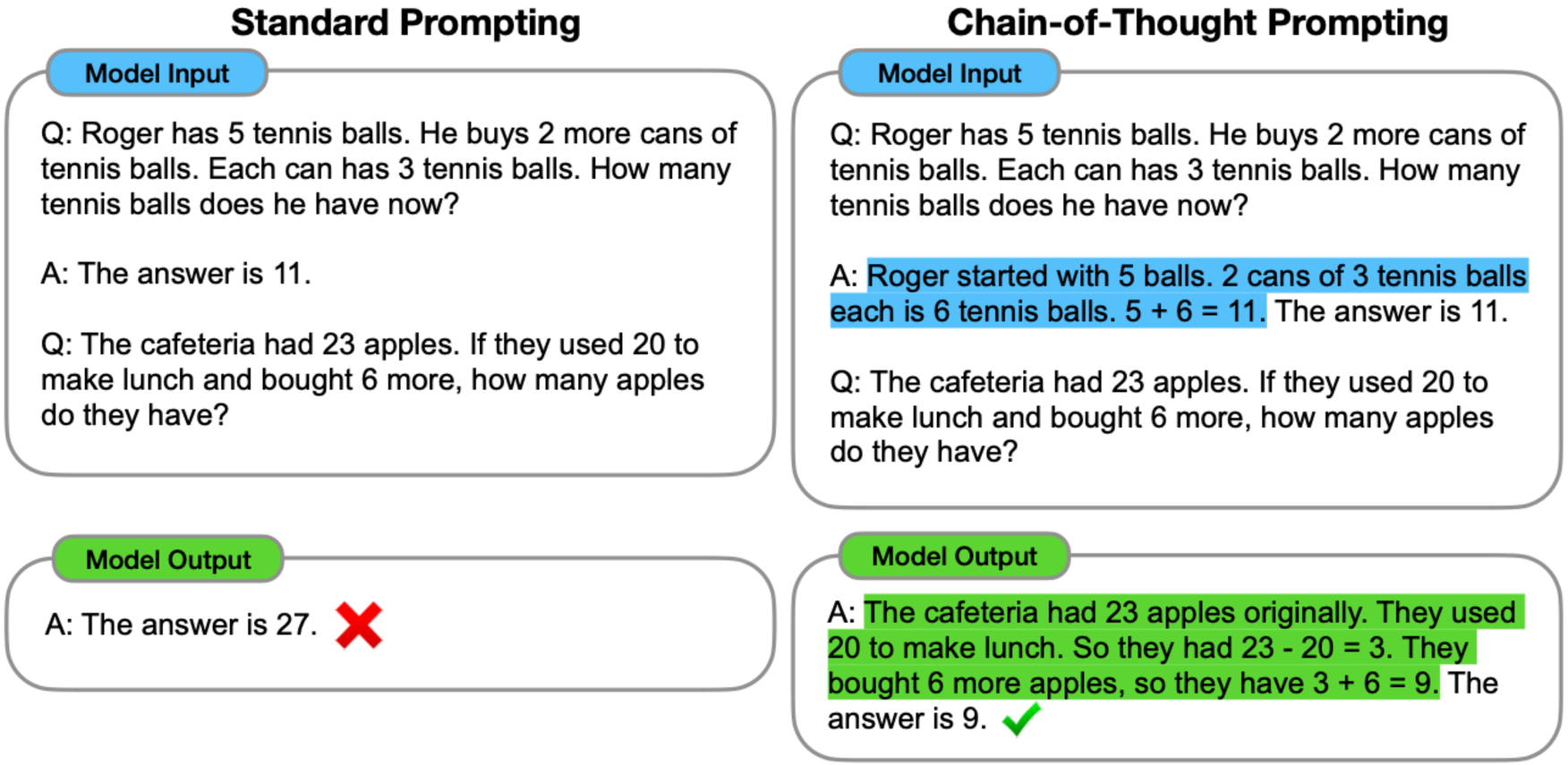
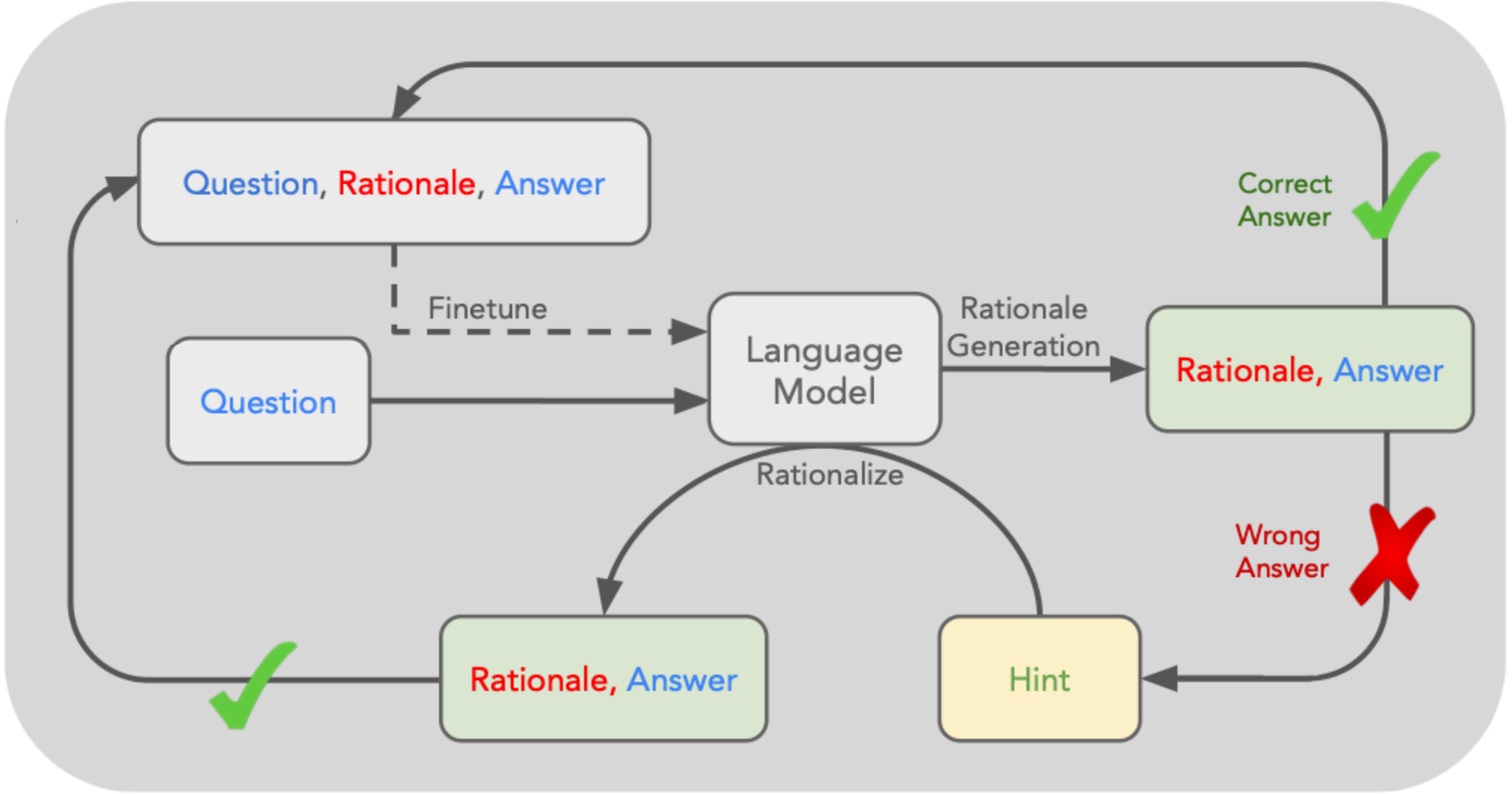


Image Credit: https://arxiv.org/pdf/2201.11903

Bootstrap Reasoning: STaR [Zelikman et al., 2022]

We can rejection sample our way to mining reasoning traces

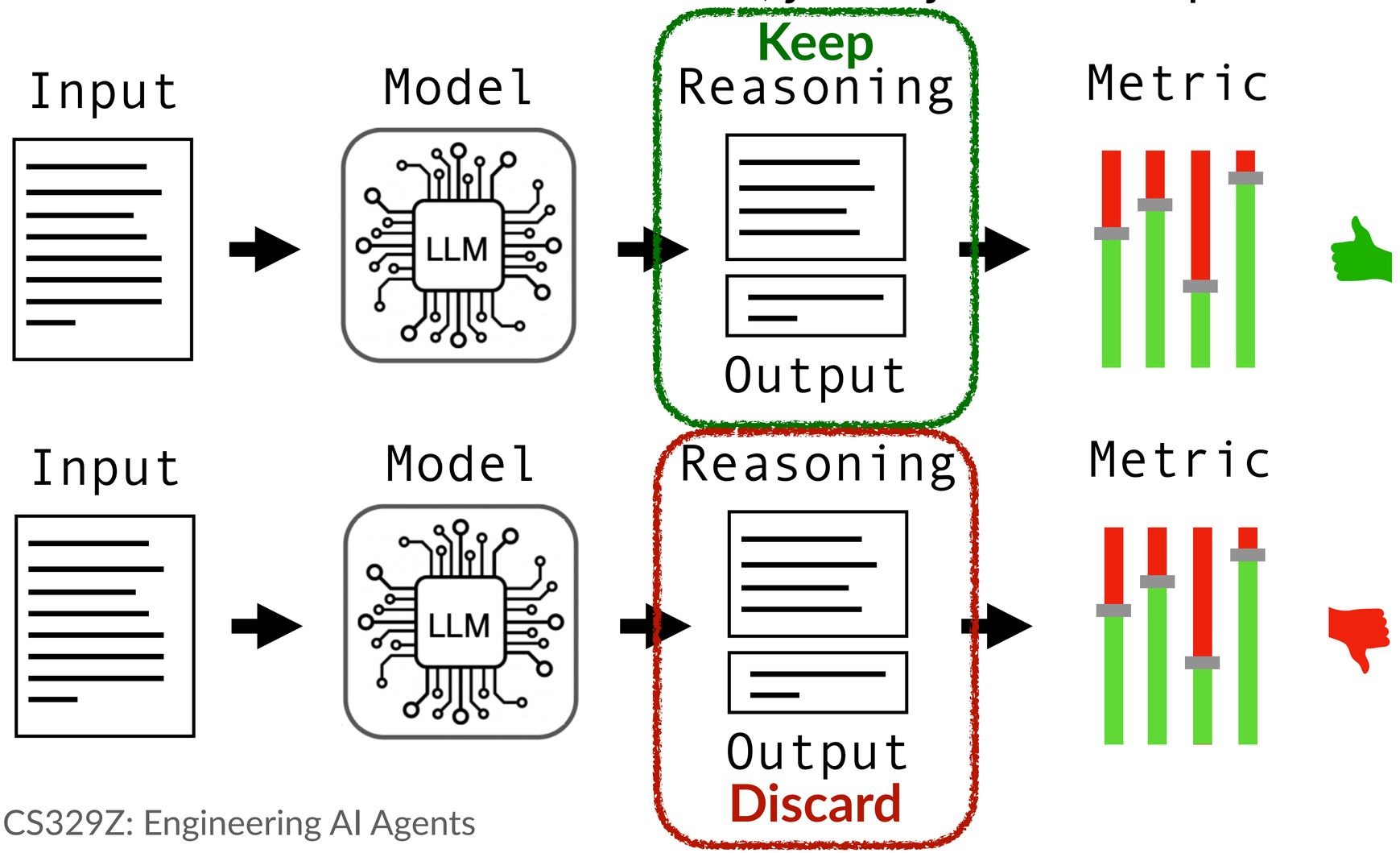



```
Q: What can be used
to carry a small dog?
Answer Choices:
(a) swimming pool
(b) basket
(c) dog show
(d) backyard
(e) own home
A: The answer must be
something that can be
used to carry a small
dog. Baskets are
designed to hold things.
Therefore, the answer
is basket (b).
```


Image Credit: https://arxiv.org/pdf/2203.14465

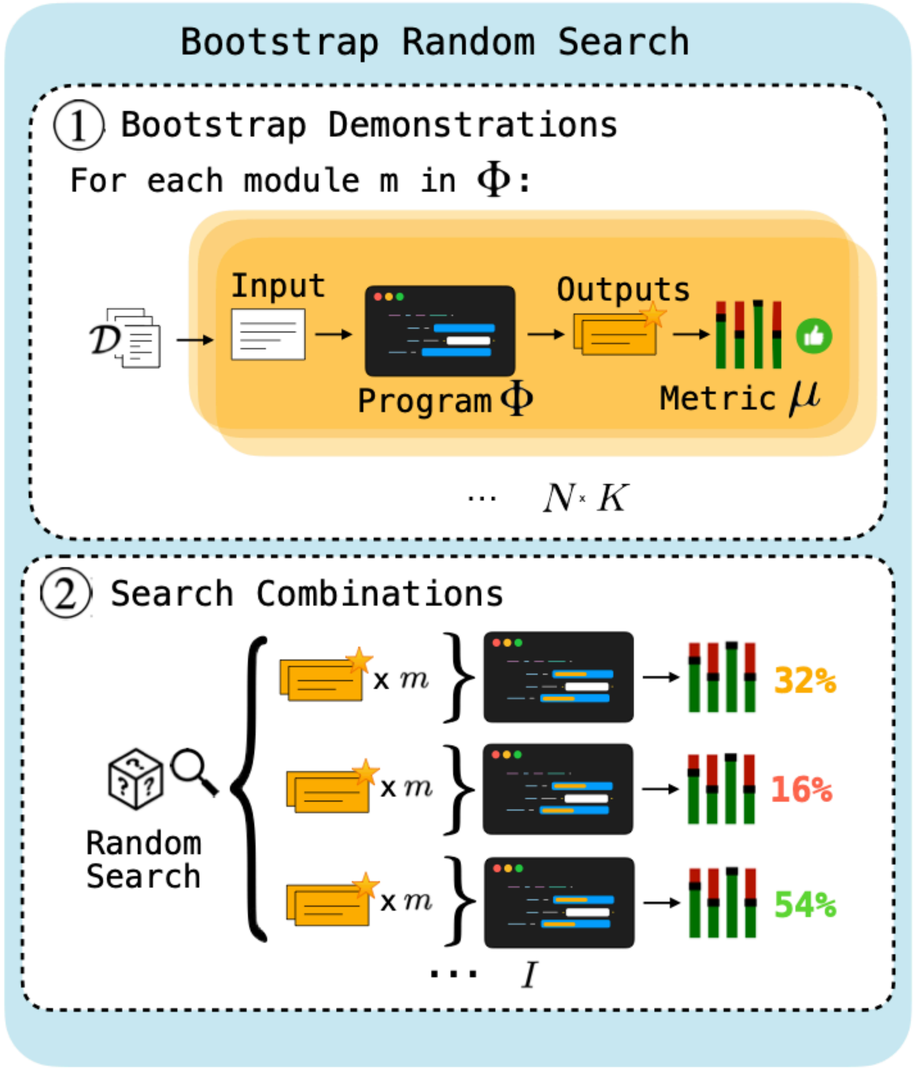
Bootstrap Reasoning: DSPy [Khattab et al., 2023]

We don't even need to fine-tune; just rejection sample successful reasoning.



Then try out combinations on the devset to find best performing fewshot examples.

Bootstrap Reasoning: DSPy [Khattab et al., 2023]



Program	Compilation	Training	GPT-3.5 Dev	GPT-3.5 Test	Llama2-13b-chat Dev	Llama2-13b-chat Test
vanilla	none	n/a	24.0	25.2	7.0	9.4
	fewshot	trainset	33.1	–	4.3	–
	bootstrap	trainset	44.0	–	28.0	–
	bootstrap×2	trainset	64.7	61.7	37.3	36.5
CoT	none	n/a	50.0	–	26.7	–
	fewshot	trainset	63.0	–	27.3	–
	fewshot	+human_CoT	78.6	72.4	34.3	33.7
	bootstrap	trainset	80.3	72.9	43.3	–

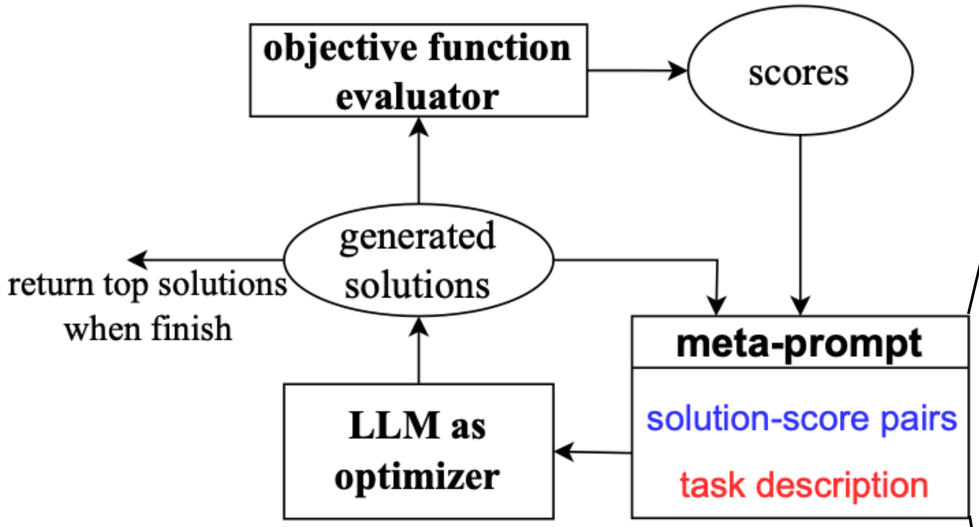
Table 1: Results with in-context learning on GSM8K math word problems.

Bootstrapping Reasoning/Outputs and then randomly searching on the trainset can be better than providing human reasoning or gold labels.

Image Credit: https://aclanthology.org/2024.emnlp-main.525.pdf
Table Credit: https://arxiv.org/pdf/2310.03714

Reflective Optimization

Large Language Models as Optimizers [Yang et al., 2024]



I have some texts along with their corresponding scores. The texts are arranged in ascending order based on their scores, where higher scores indicate better quality.

text:
Let's figure it out!
score:
61

text:
Let's solve the problem.
score:
63

(. . . more instructions and scores . . .)

The following exemplars show how to apply your text: you replace <INS> in each input with your text, then read the input and give an output. We say your output is wrong if your output is different from the given output, and we say your output is correct if they are the same.

input:
Q: Alannah, Beatrix, and Queen are preparing for the new school year and have been given books by their parents. Alannah has 20 more books than Beatrix. Queen has 1/5 times more books than Alannah. If Beatrix has 30 books, how many books do the three have together?
A: <INS>
output:
140

(. . . more exemplars . . .)

Write your new text that is different from the old ones and has a score as high as possible. Write the text in square brackets.

Allow the LLM to see past instructions and how performant they were, then ask the LLM to write a new instruction.

Image Credit: https://arxiv.org/pdf/2309.03409

Large Language Models as Optimizers [Yang et al., 2024]

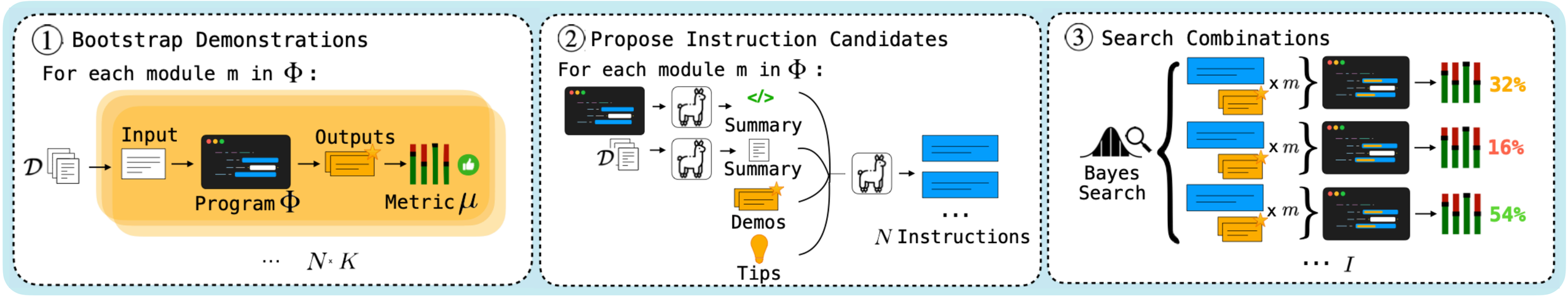
Table 1: Top instructions with the highest GSM8K zero-shot test accuracies from prompt optimization with different optimizer LLMs. All results use the pre-trained `PaLM 2-L` as the scorer.

Source	Instruction	Acc
Baselines		
(Kojima et al., 2022)	Let's think step by step.	71.8
(Zhou et al., 2022b)	Let's work this out in a step by step way to be sure we have the right answer.	58.8
	(empty string)	34.0
Ours		
`PaLM 2-L-IT`	Take a deep breath and work on this problem step-by-step.	**80.2**
`PaLM 2-L`	Break this down.	79.9
`gpt-3.5-turbo`	A little bit of arithmetic and a logical approach will help us quickly arrive at the solution to this problem.	78.5
`gpt-4`	Let's combine our numerical command and clear thinking to quickly and accurately decipher the answer.	74.5

Identifies interesting strategies like "Take a deep breath…"

Image Credit: https://arxiv.org/pdf/2309.03409

MIPROv2 [Opsahl-Ong et al., 2024]



1. Bootstrap Demonstrations (Reasoning/Output if the program is CoT)
2. Propose Instructions based on Reflection
3. Search the Combinations of Instructions and Demonstrations

Let's walk through an example!

Image Credit: https://aclanthology.org/2024.emnlp-main.525.pdf

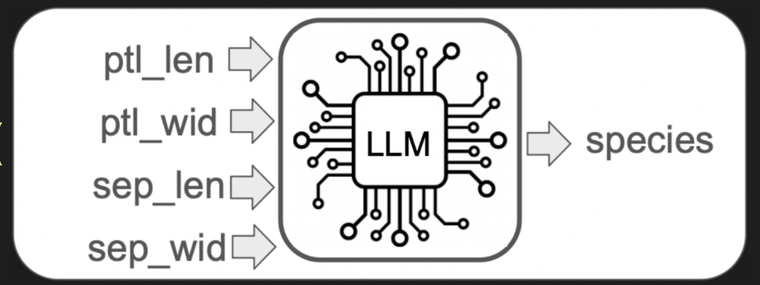
> Bootstrap Fewshot Generation

We run training examples several times to collect successful traces of your program to use as few shot demonstrations.

> Bootstrap Fewshot Generation

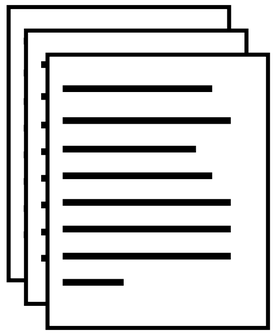
LM Program

```
def classify_iris(ptl_len, ptl_wid, sep_len, sep_wid):
    return (
    )
```



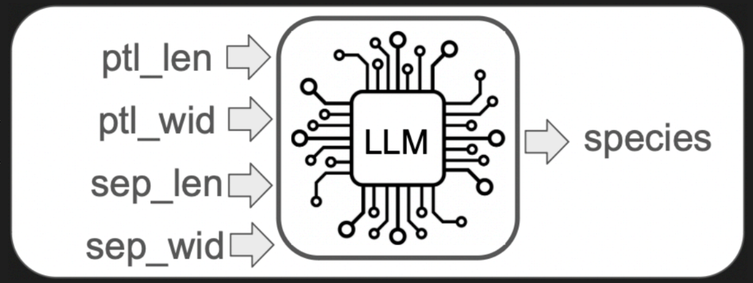
> Bootstrap Fewshot Generation

Training Input



LM Program

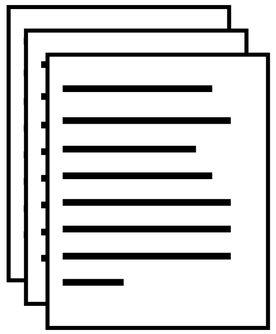
```
def classify_iris(ptl_len, ptl_wid, sep_len, sep_wid):
    return (
```




```
    )
```

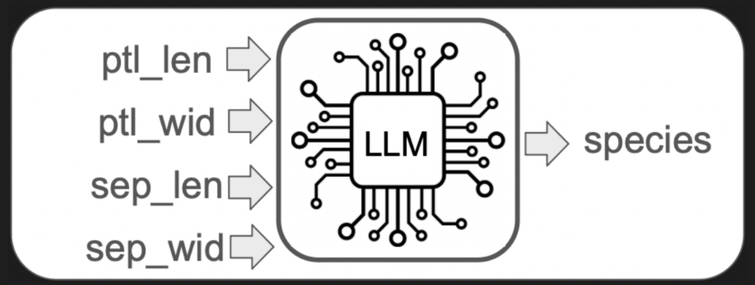
> Bootstrap Fewshot Generation

Training Input



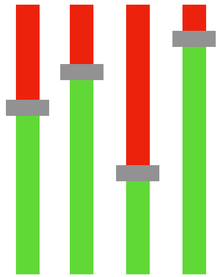
LM Program

```
def classify_iris(ptl_len, ptl_wid, sep_len, sep_wid):
    return (
```

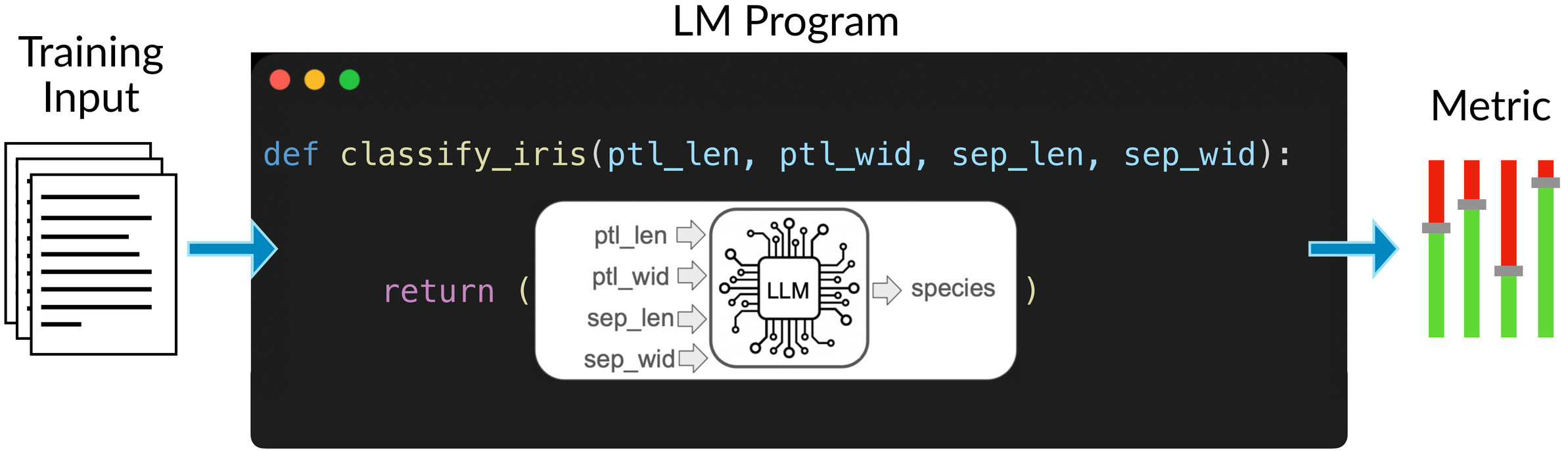



```
    )
```

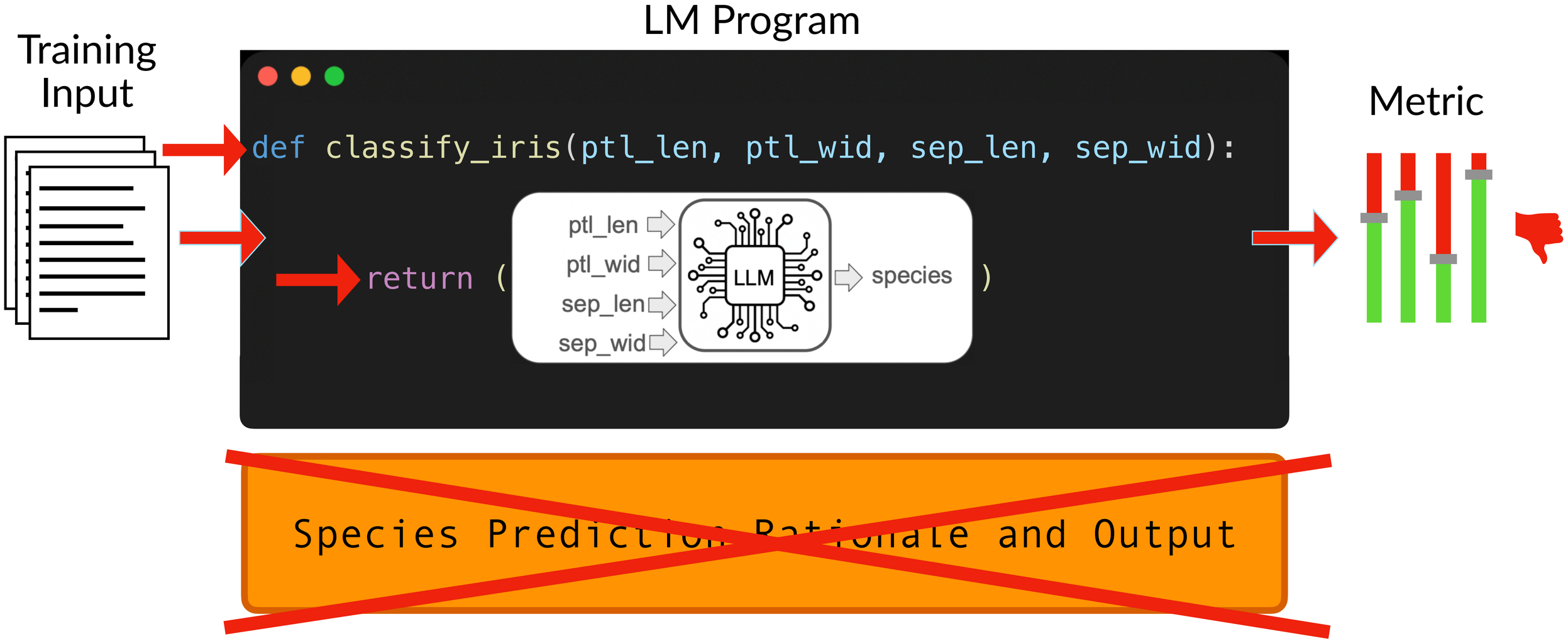
Metric



> Bootstrap Fewshot Generation

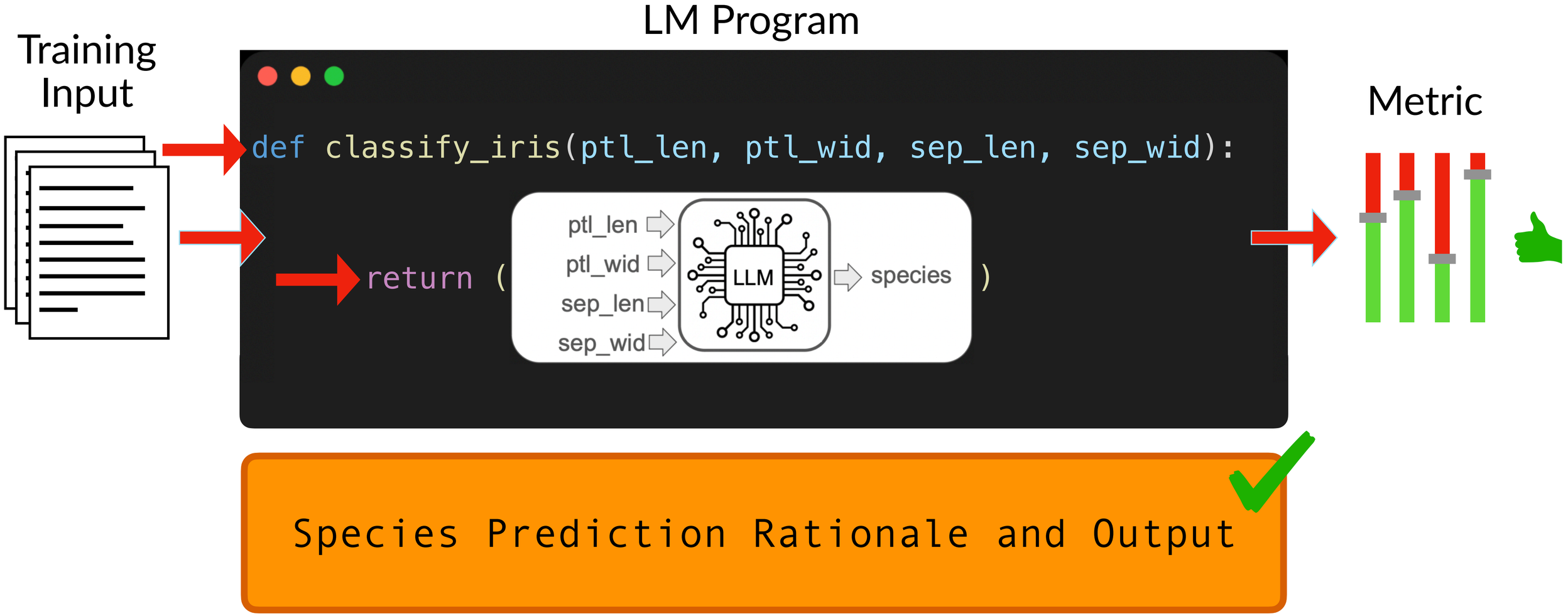


> Bootstrap Fewshot Generation



If the output scores poorly on the metric we discard the trace

> Bootstrap Fewshot Generation



If the output scores highly on the metric we keep the trace

> Bootstrap Fewshot Generation

```
petal_length: "1.4",
petal_width: "0.2",
sepal_length: "4.6",
sepal_width: "3.2",
rationale: “Let’s think step by step in order to predict
the iris species. We know that the petal length is 1.4 cm,
which is relatively short compared to the other species.
The petal width is 0.2 cm, which is also relatively narrow.
The sepal length is 4.6 cm, which is relatively long, and
the sepal width is 3.2 cm, which is relatively wide. Based
on these characteristics, we can conclude that the iris
species is likely to be setosa.",
answer: "setosa"
```

...

> Grounded Instruction Proposal with MIPROv2

We collect several details about your task to inform the prompt proposal LM.

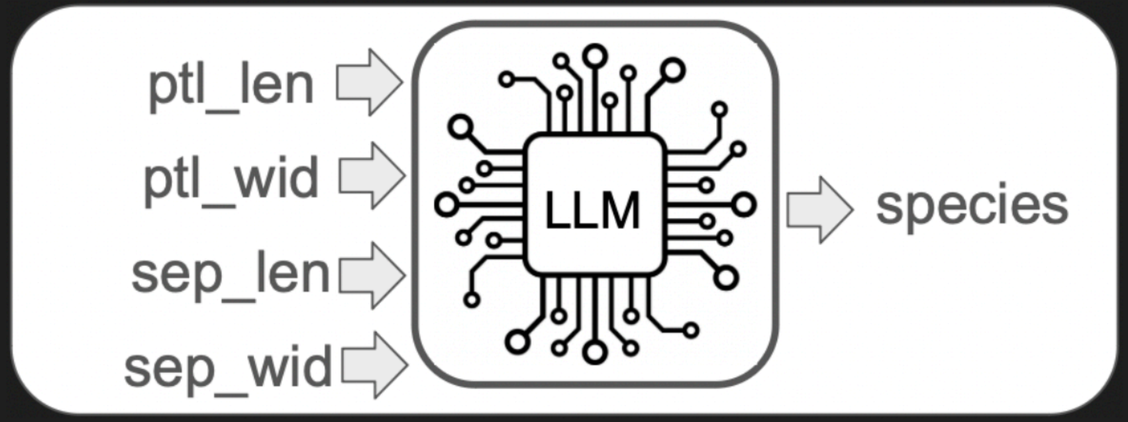
> Grounded Instruction Proposal with MIPROv2

1. Program-Aware Grounding

> Grounded Instruction Proposal with MIPROv2

1. Program-Aware Grounding

```
def classify_iris(ptl_len, ptl_wid, sep_len, sep_wid):
    return (
```

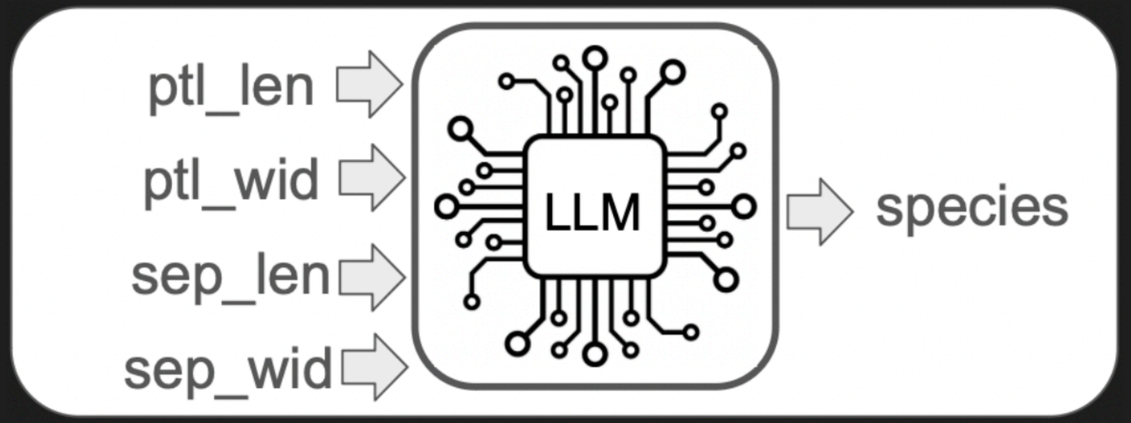



```
    )
```

> Grounded Instruction Proposal with MIPROv2

1. Program-Aware Grounding

```
def classify_iris(ptl_len, ptl_wid, sep_len, sep_wid):
    return (
```




```
    )
```

The task the program is designed to solve is classifying iris flowers into one of three species (setosa, versicolour, or virginica) based on the given dimensions of their petals and sepals. The program uses a language model to predict the species by reasoning through the provided dimensions.

> Grounded Instruction Proposal with MIPROv2

2. Data-Aware Grounding

Sepal Length	Sepal Width (cm)	Pedal Length (cm)	Pedal Width (cm)	Species
5.1	3.5	1.4	0.2	Iris-setosa
6.3	2.3	4.4	1.3	Iris-versicolor
5.4	3.7	1.5	0.2	Iris-setosa
6.9	3.1	5.1	2.3	Iris-virginica

...

> Grounded Instruction Proposal with MIPROv2

2. Data-Aware Grounding

Sepal Length	Sepal Width (cm)	Pedal Length (cm)	Pedal Width (cm)	Species
5.1	3.5	1.4	0.2	Iris-setosa
6.3	2.3	4.4	1.3	Iris-versicolor

...

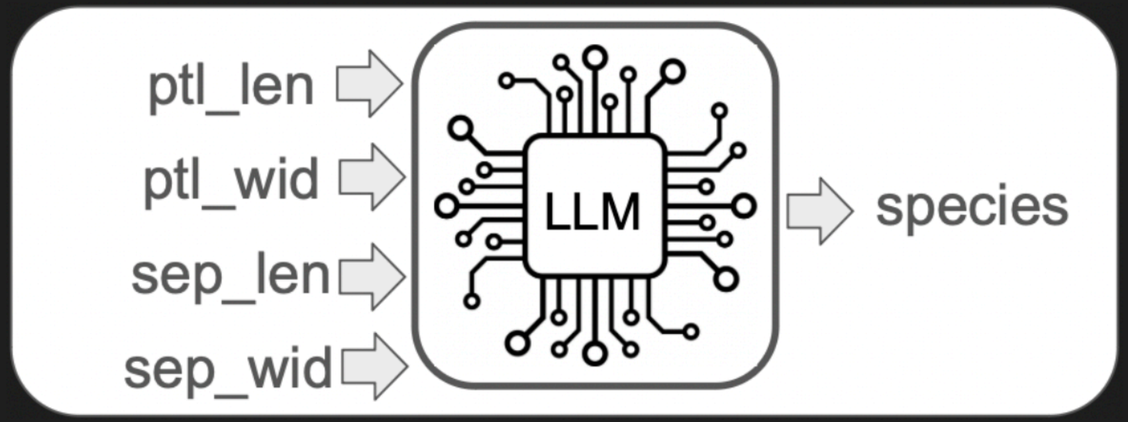
```
The dataset appears to have three classes: setosa, versicolor, and
virginica. Each of the classes has a distinct distribution on the
petal_length and petal_width variables. However, the sepal_length and
sepal_width variables appear to have less clear cut trends. Additionally,
the species variable does not provide much additional information since it
is a duplicate of the answer variable. ---

Is this correct? Can I improve this? Please let me know if so.
```

> Grounded Instruction Proposal with MIPROv2

3. Successful Bootstrapped Program Traces

```
def classify_iris(ptl_len, ptl_wid, sep_len, sep_wid):
    return (
```




```
    )
```

```
classify_iris(5.1, 3.5, 1.4, 0.2)    “versicolor”
```

```
classify_iris(5.1, 1.8, 5.9, 3.0)    “verginica”
```

> Grounded Instruction Proposal with MIPROv2

3. Successful Bootstrapped Program Traces

```
classify_iris(5.1, 1.8, 5.9, 3.0)    “verginica”
```



> Grounded Instruction Proposal with MIPROv2

3. Successful Bootstrapped Program Traces

```
classify_iris(5.1, 1.8, 5.9, 3.0)   “verginica”
```



```
petal_length: 5.1
petal_width: 1.8
sepal_length: 5.9
sepal_width: 3.0
rationale: Let’s think step by step to produce the answer. We know that the
petal length of 5.1 is relatively long compared to the setosa species,
which typically has a petal length of around 4.8. The petal width of 1.8 is
also slightly wider than the setosa species, which typically has a petal
width of around 1.5. The sepal length and width are both relatively long
compared to the setosa species. Therefore, based on these characteristics,
we can conclude that the iris is likely to be either a versicolor or
virginica. However, the sepal length of 5.9 and the sepal width of 3.0 are
slightly more characteristic of the virginica species. Therefore, we can
answer: virginia
```

> Grounded Instruction Proposal with MIPROv2

4. Prompting Tips
We only show one of these at a time.

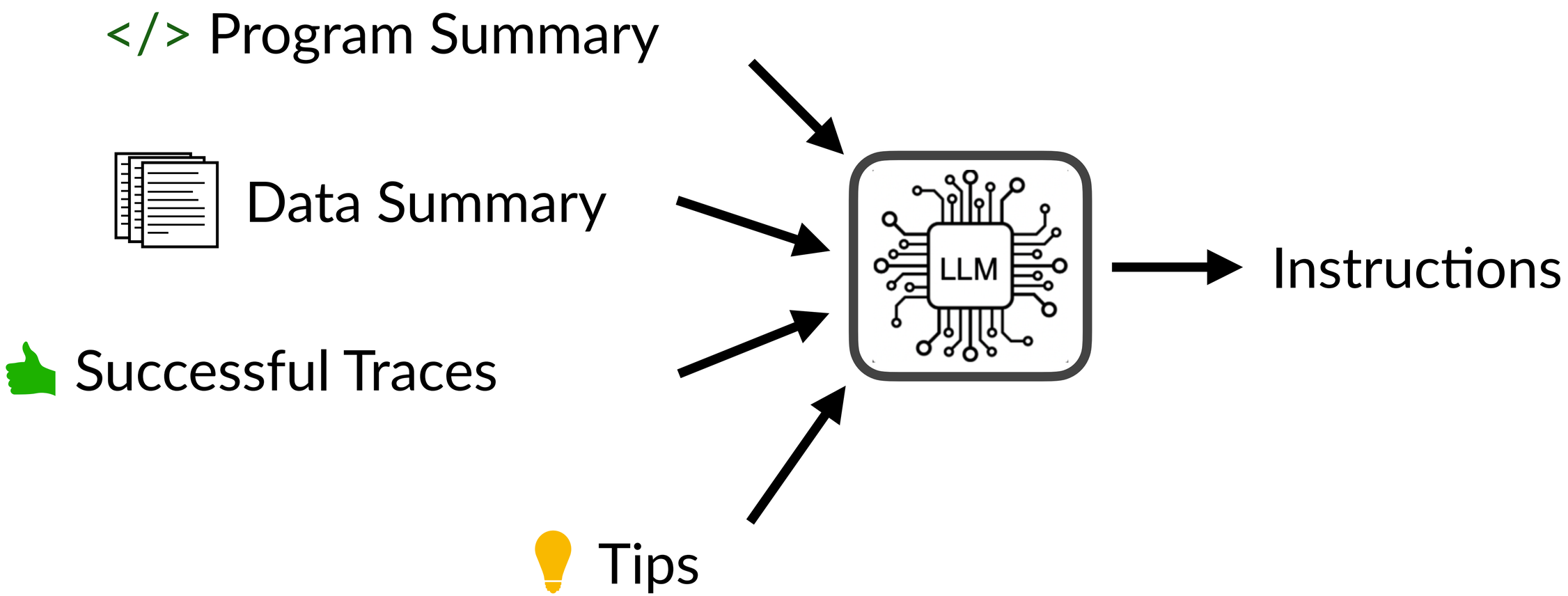
“Don’t be afraid to be creative”

“The instruction should include a high stakes scenario in which the LM must solve the task!”

“Provide the LM with a persona that is relevant to the task (ie. \"You are a …\")”

“Keep the instruction clear and concise.”

> Grounded Instruction Proposal with MIPROv2



> Grounded Instruction Proposal with MIPROv2

Instructions

> Grounded Instruction Proposal with MIPROv2

Instructions

Using the provided petal length, petal width, sepal length, and sepal width measurements in cm, predict the iris species accurately to save a critically endangered species from extinction.

Given the petal length, petal width, sepal length, and sepal width in centimeters, predict the species of the iris flower (setosa, versicolour, or virginica). Provide a step-by-step reasoning for your prediction based on the given measurements.

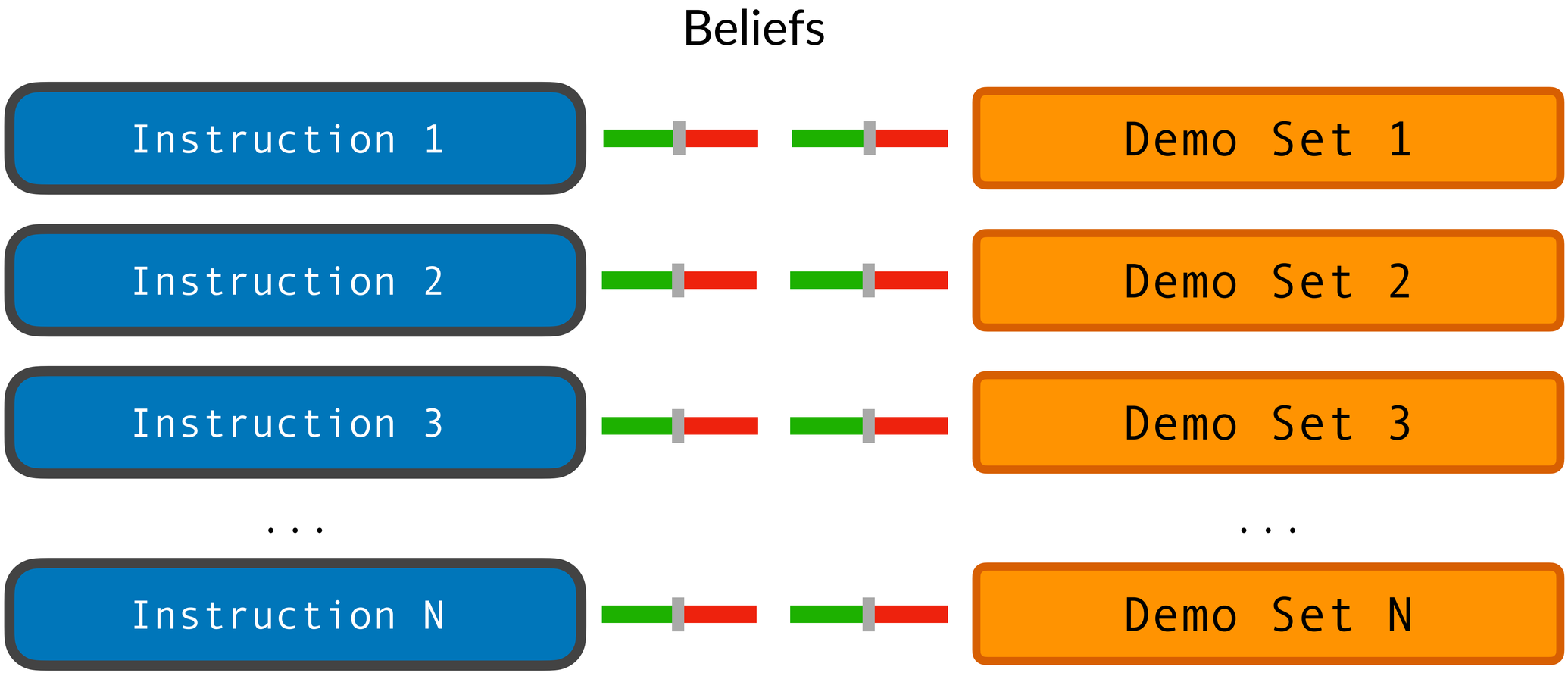
Given the critical situation in which a rare species of iris is on the brink of extinction, predict the iris species based on the dimensions of the petals and sepals in order to save it from extinction.

...

> Optimizing the Combination of Instructions and Demonstrations in MIPROv2

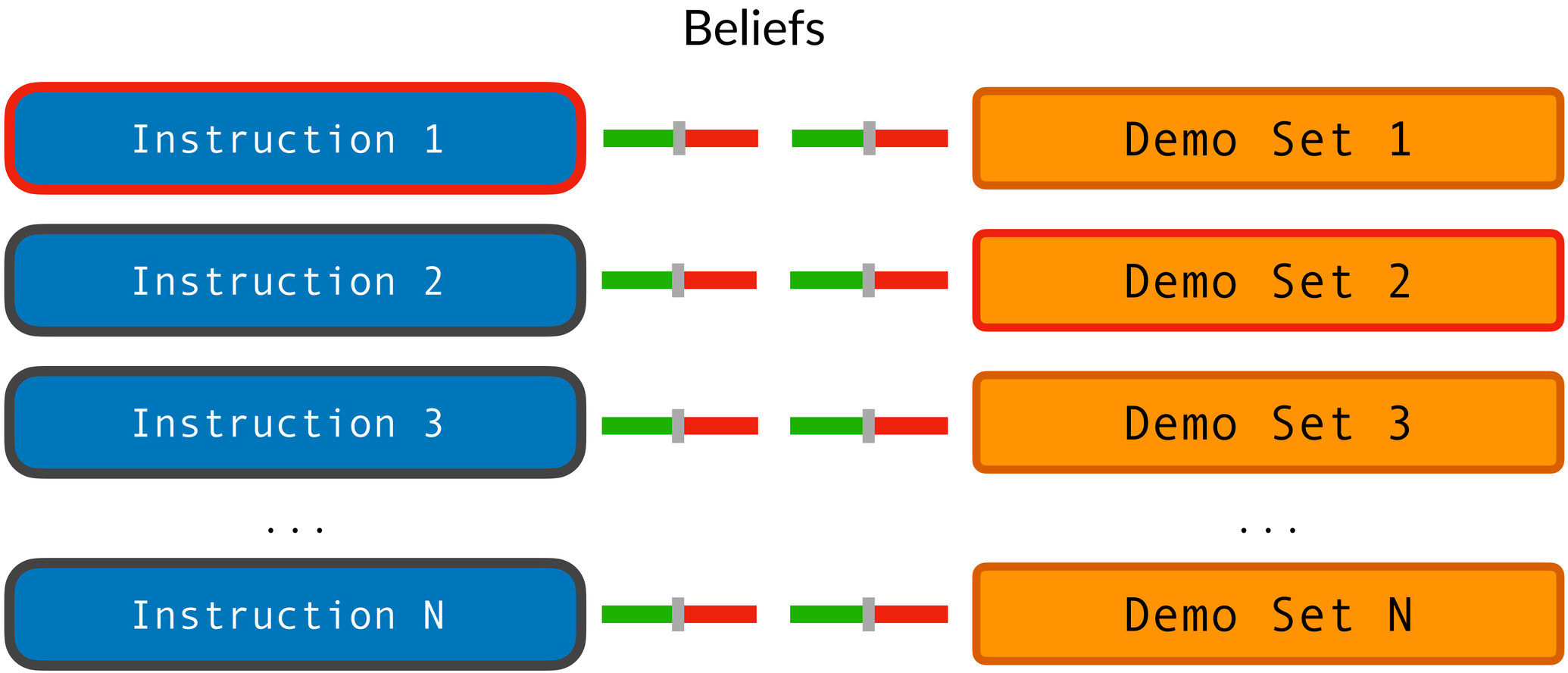
Given our set of instructions and demonstrations from the first two steps we iteratively sample combinations to update our belief that a given instruction or demonstration is good using a bayesian surrogate model.

> Optimizing the Combination of Instructions and Demonstrations in MIPROv2



Sample parameters for our program based on beliefs

> Optimizing the Combination of Instructions and Demonstrations in MIPROv2



Sample parameters for our program based on beliefs

> Optimizing the Combination of Instructions and Demonstrations in MIPROv2

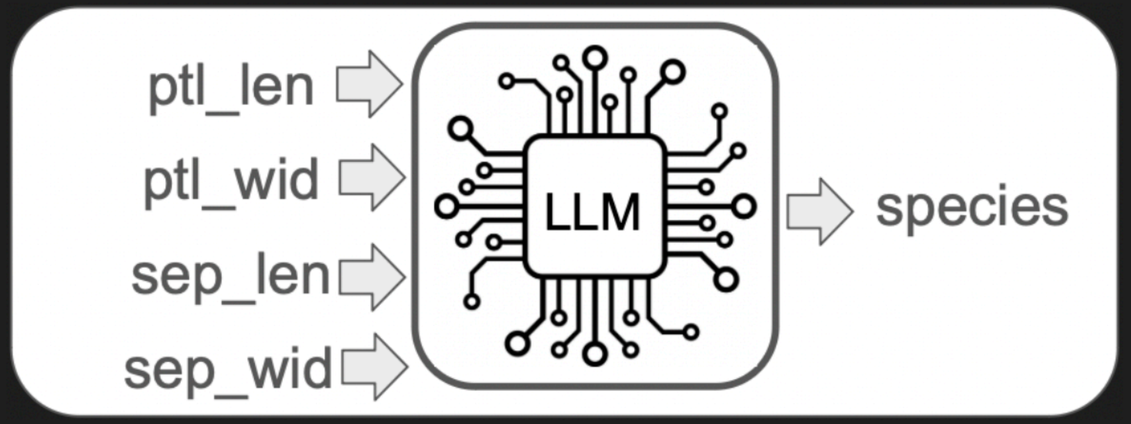


> Optimizing the Combination of Instructions and Demonstrations in MIPROv2

Instruction 1

Demo Set 2

```
def classify_iris(ptl_len, ptl_wid, sep_len, sep_wid):
    return (
                                                    )
```



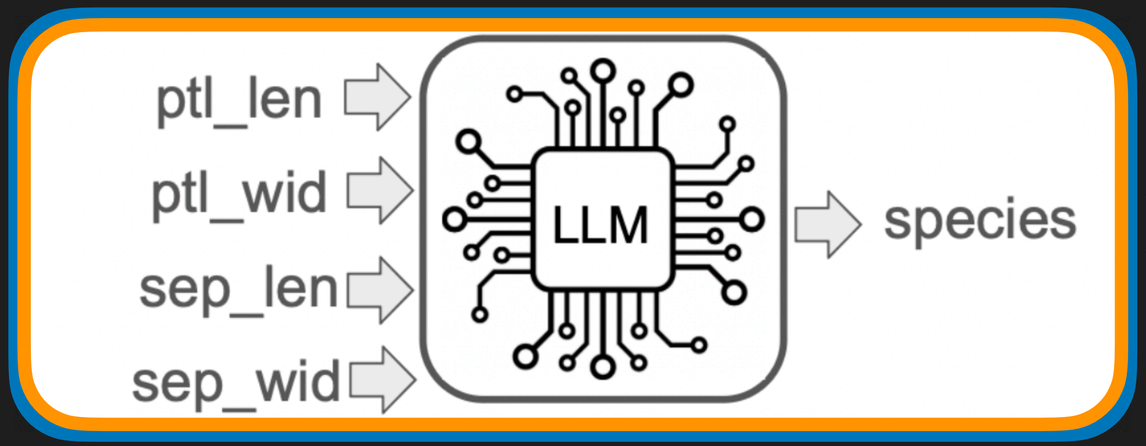
Slot parameters into the program

> Optimizing the Combination of Instructions and Demonstrations in MIPROv2

Instruction 1

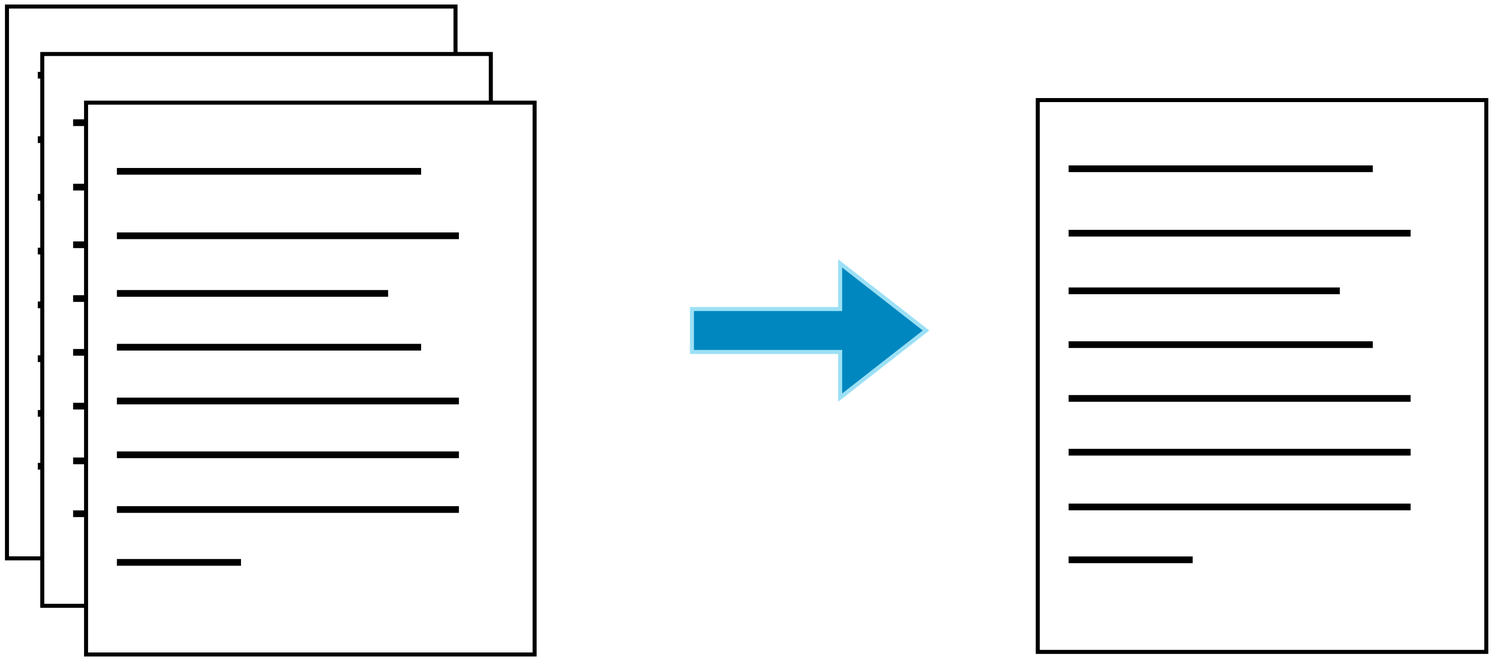
Demo Set 2

```
def classify_iris(ptl_len, ptl_wid, sep_len, sep_wid):
    return (
    )
```



Slot parameters into the program

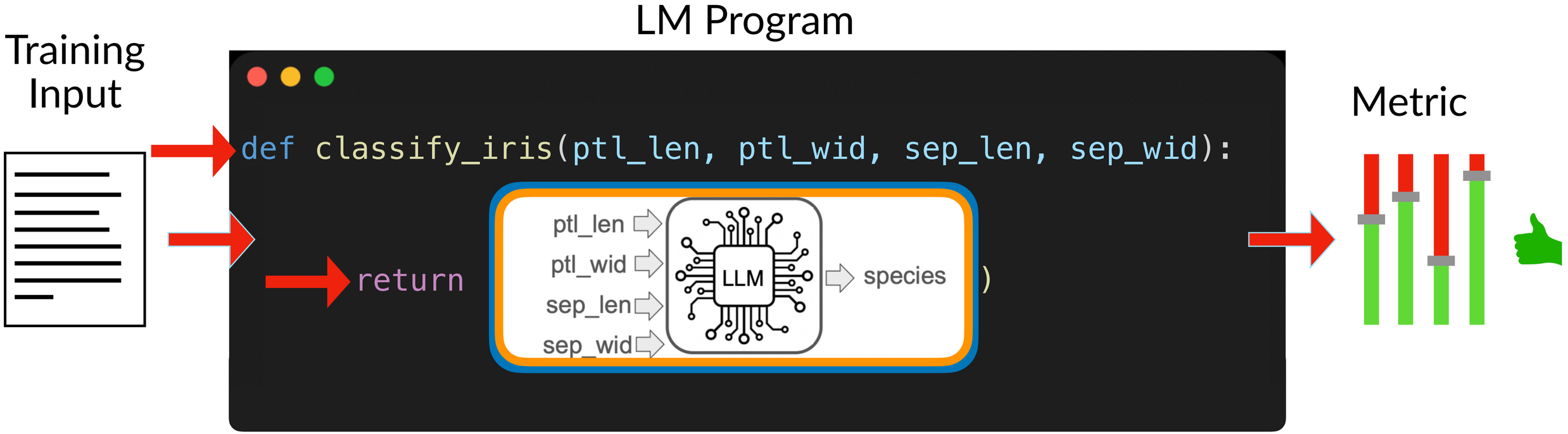
> Optimizing the Combination of Instructions and Demonstrations in MIPROv2



Sample minibatch from training data

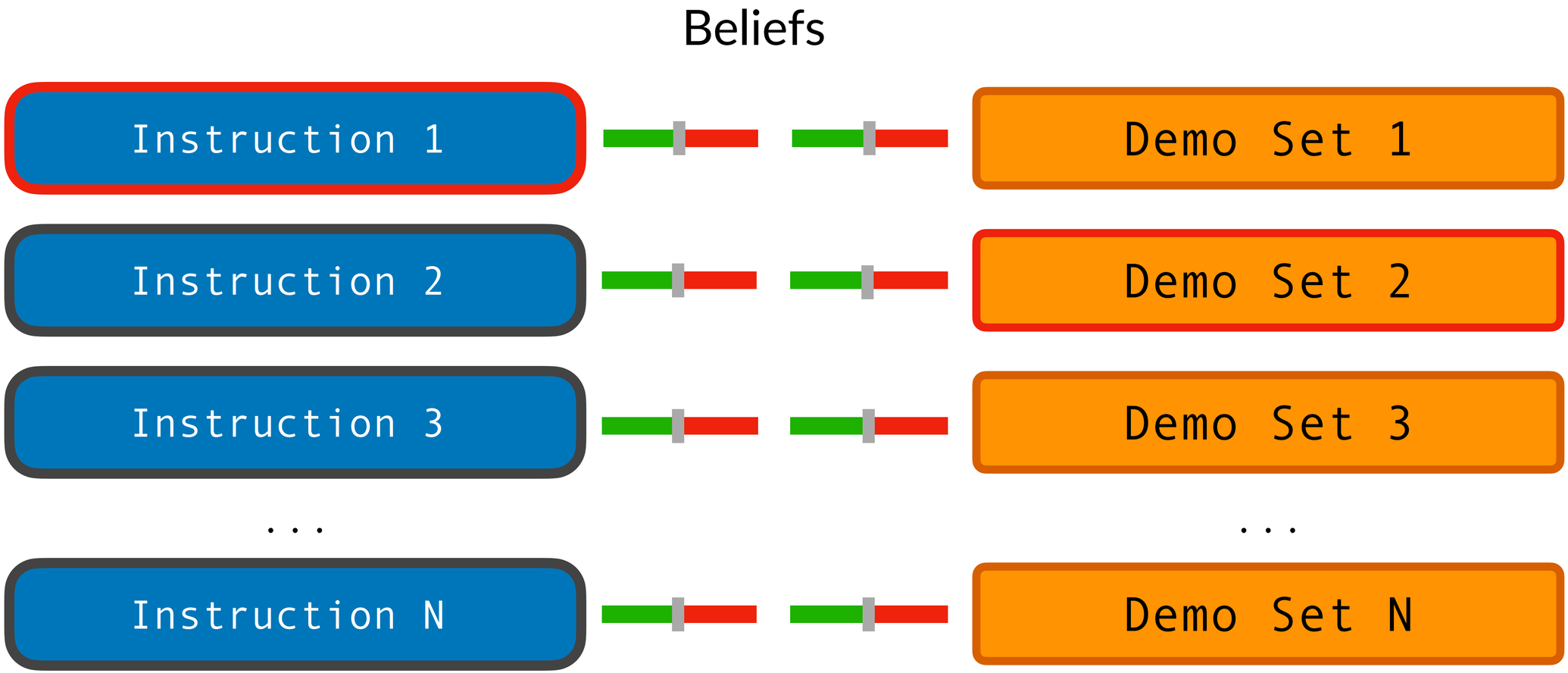
*Every few iterations we also do a full evaluation

> Optimizing the Combination of Instructions and Demonstrations in MIPROv2



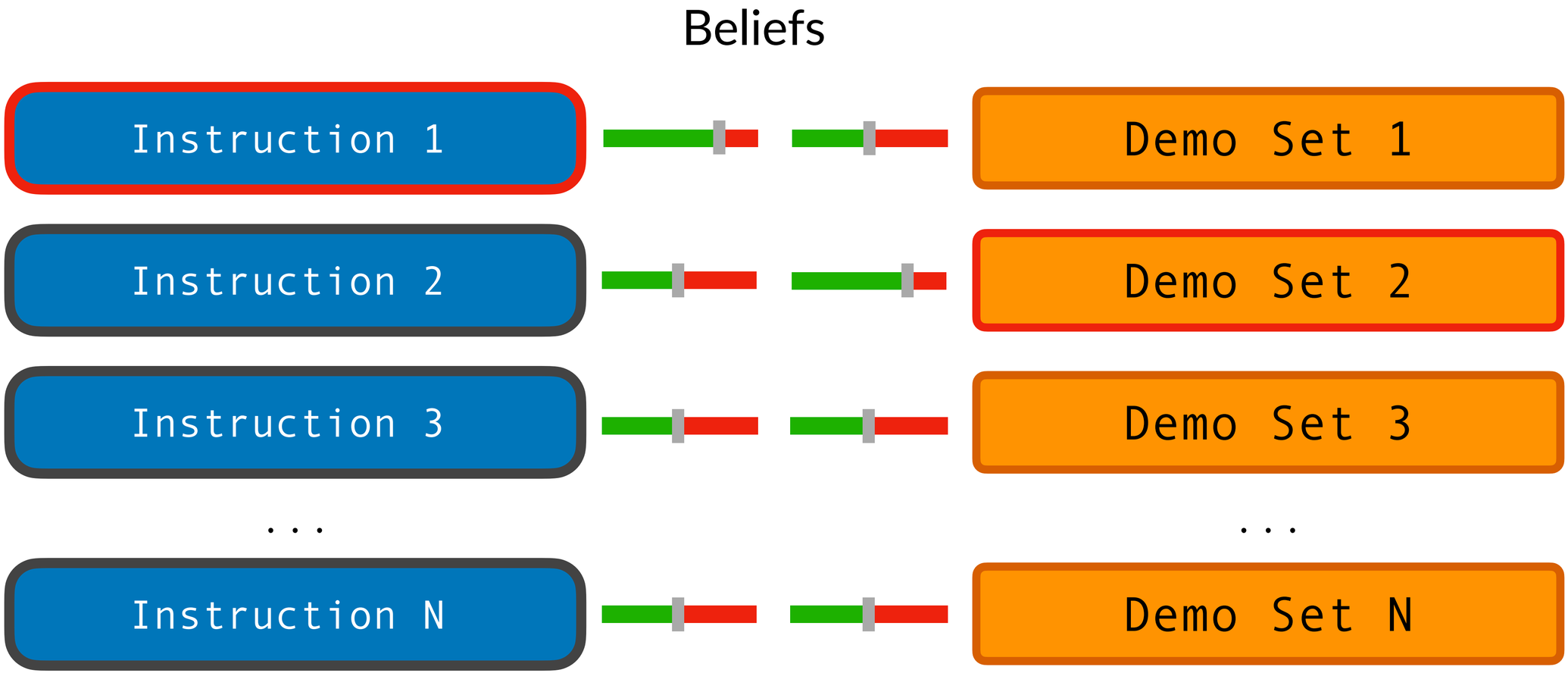
Run the minibatch on the updated program

> Optimizing the Combination of Instructions and Demonstrations in MIPROv2



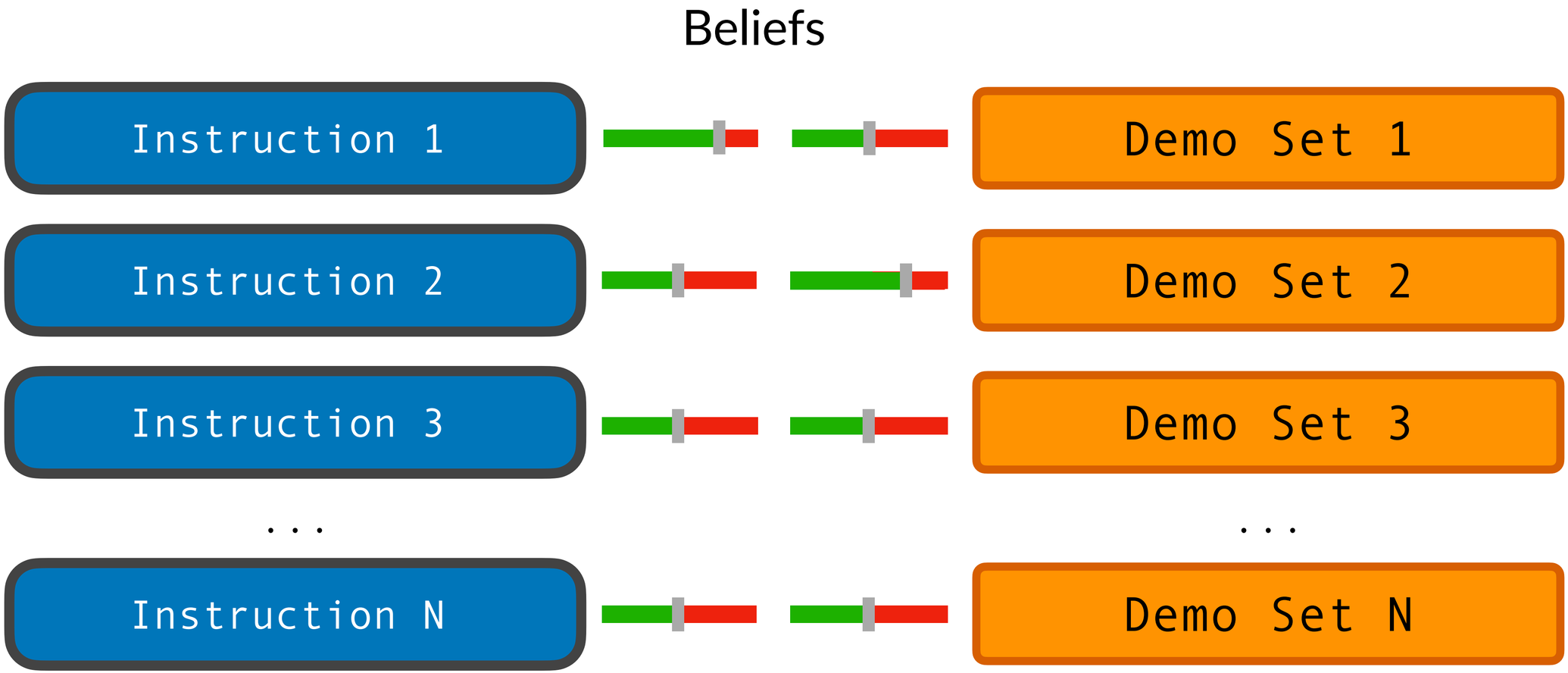
Update beliefs based on mini batch results

> Optimizing the Combination of Instructions and Demonstrations in MIPROv2



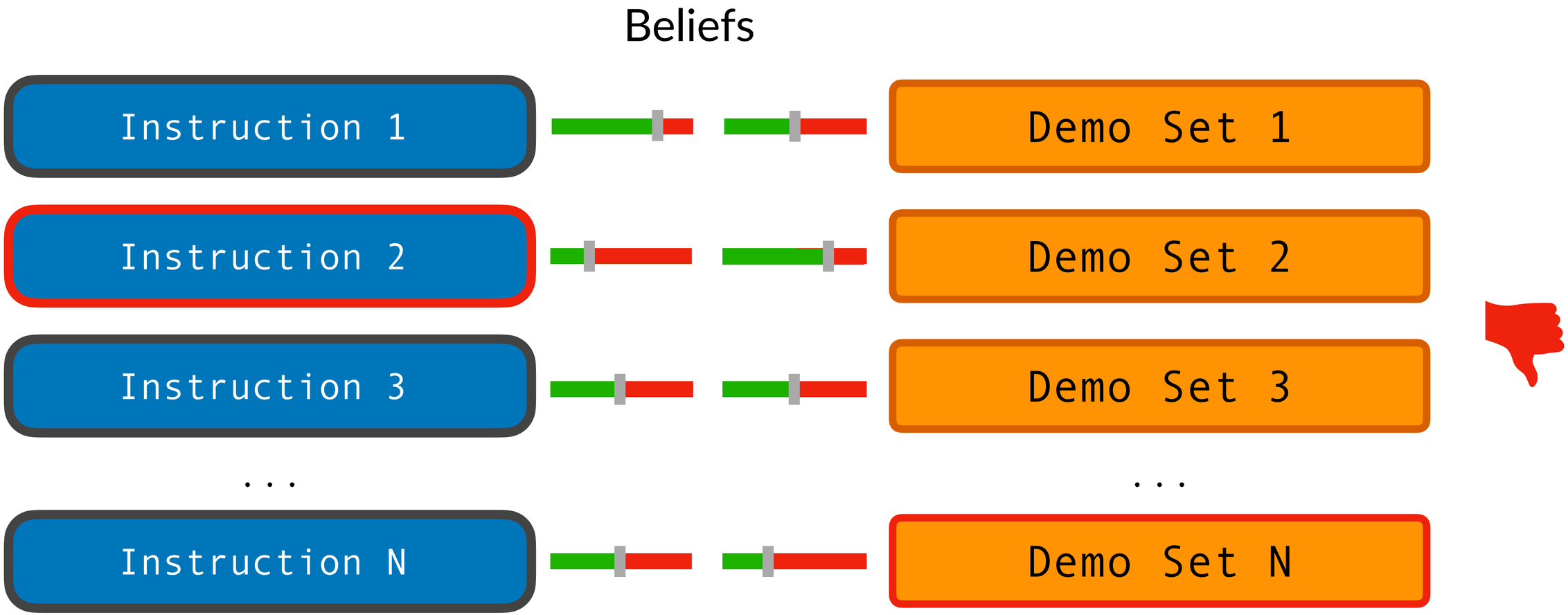
Update beliefs based on mini batch results

> Optimizing the Combination of Instructions and Demonstrations in MIPROv2



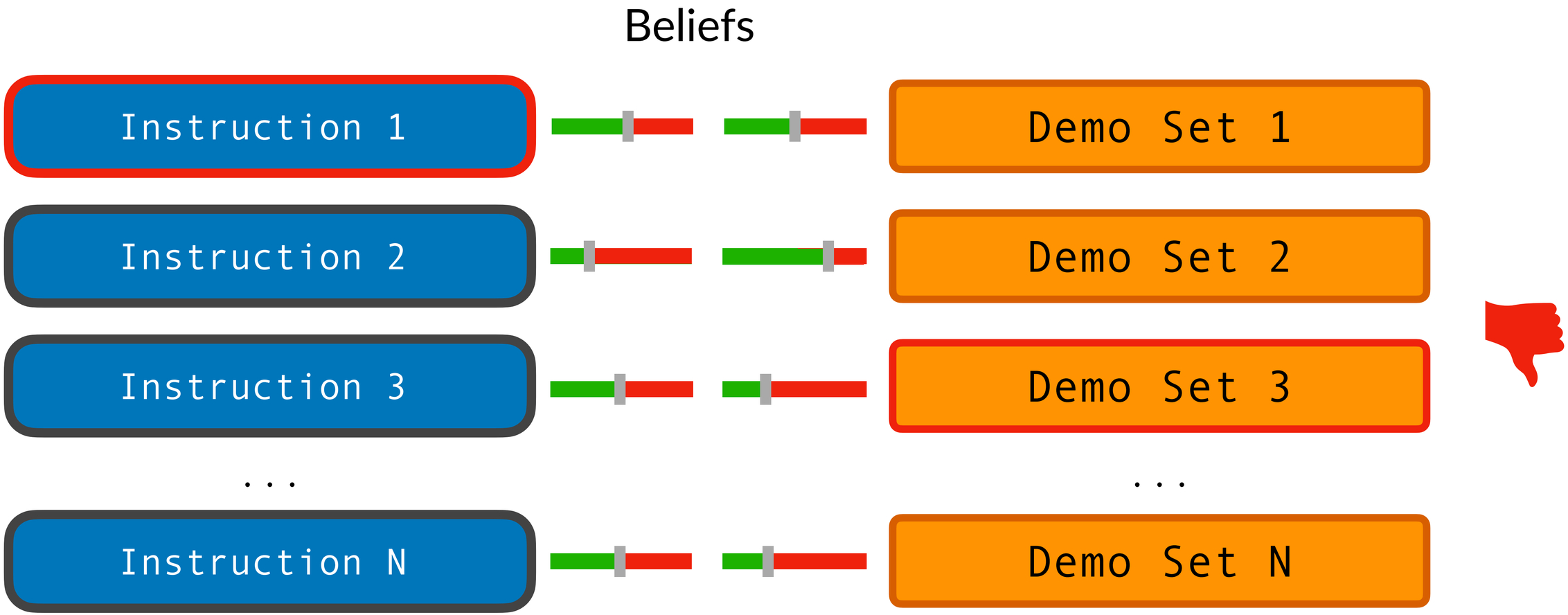
Repeat num_batches times, doing a full evaluation every few iterations

> Optimizing the Combination of Instructions and Demonstrations in MIPROv2



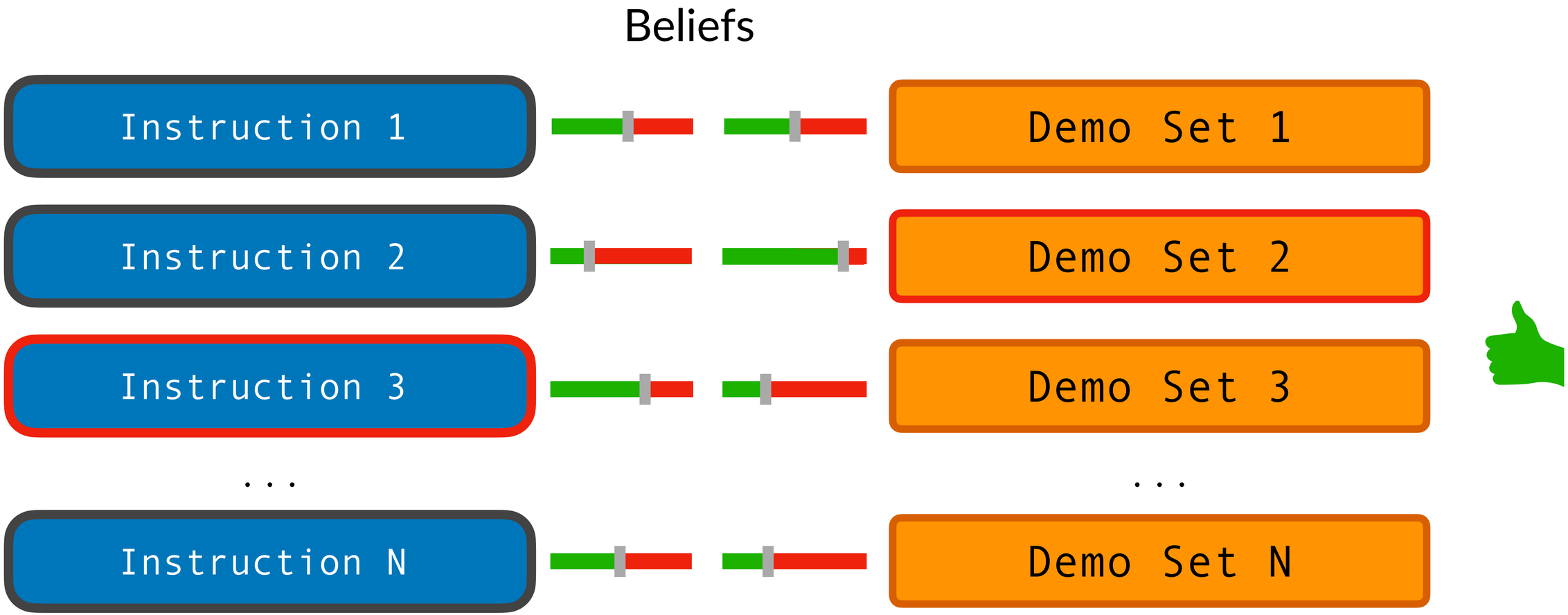
Repeat num_batches times, doing a full evaluation every few iterations

> Optimizing the Combination of Instructions and Demonstrations in MIPROv2



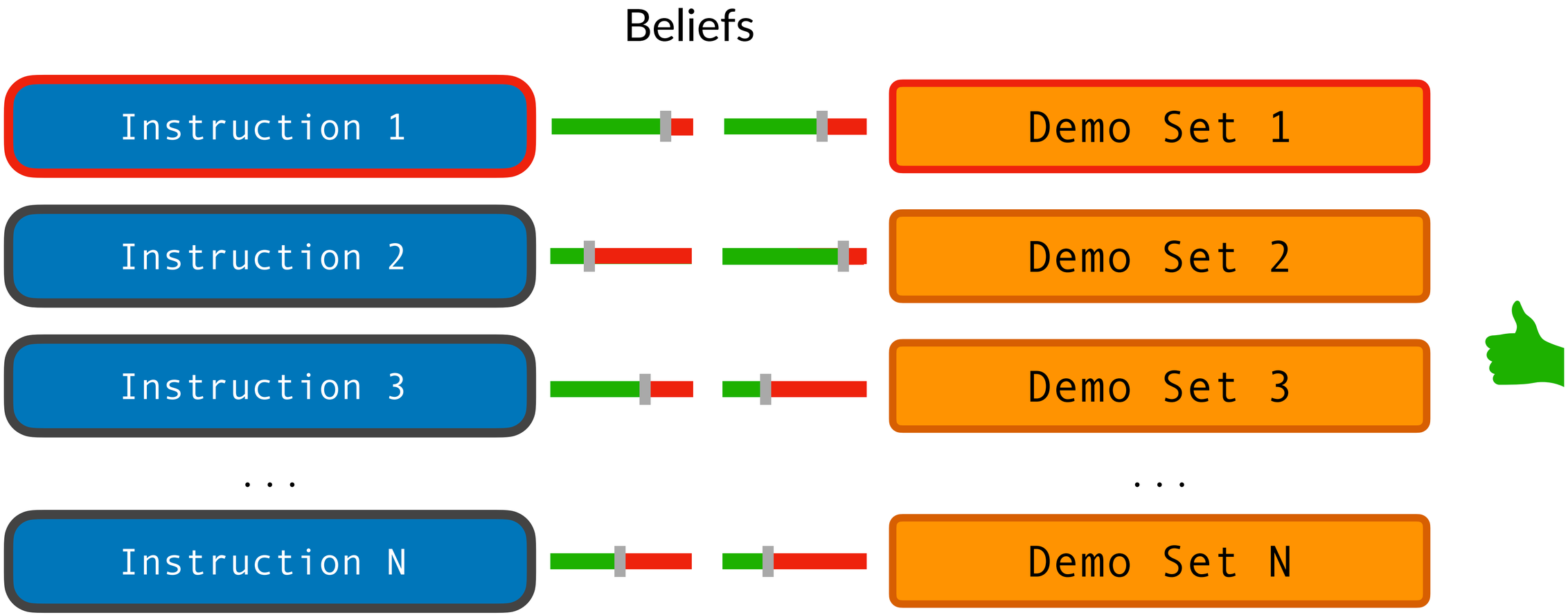
Repeat num_batches times, doing a full evaluation every few iterations

> Optimizing the Combination of Instructions and Demonstrations in MIPROv2



Repeat num_batches times, doing a full evaluation every few iterations

> Optimizing the Combination of Instructions and Demonstrations in MIPROv2



Repeat num_batches times, doing a full evaluation every few iterations

> Optimizing the Combination of Instructions and Demonstrations in MIPROv2

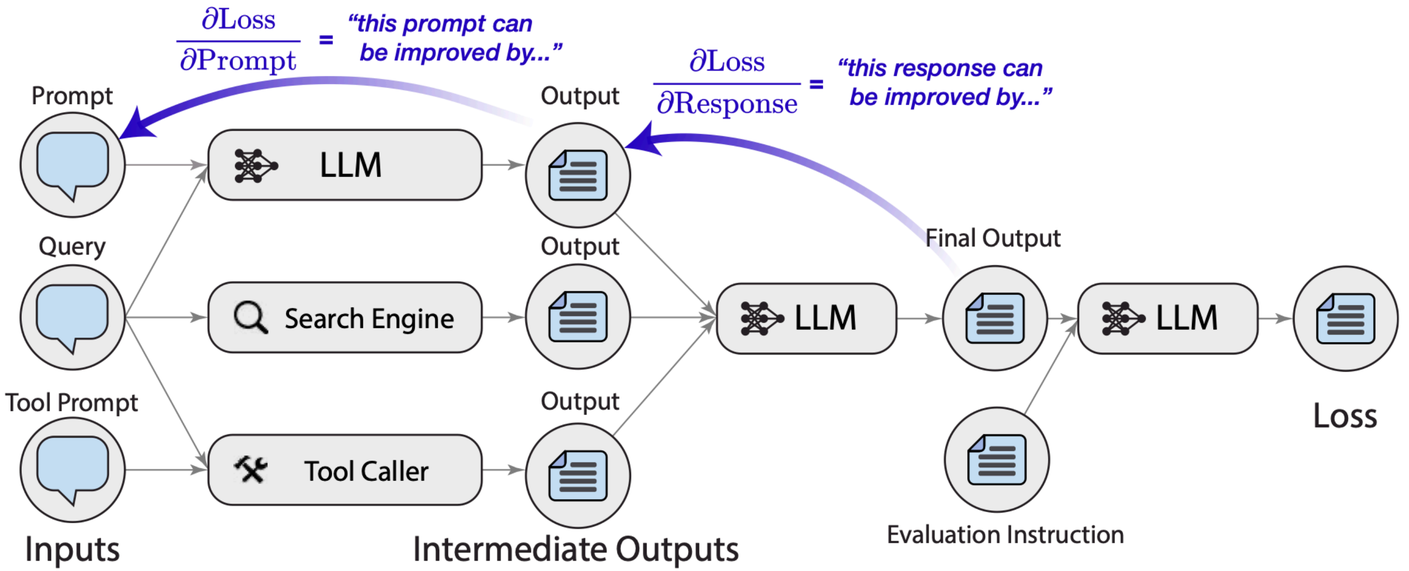
Example Optimized Module

```
“Given the petal and sepal dimensions in cm, predict the iris species.”
```



```
Given the critical situation in which a rare species of iris is on the brink
of extinction, predict the iris species based on the dimensions of the
petals and sepals in order to save it from extinction.
—
petal_length: "1.5",petal_width: "0.1",sepal_length: "5.2",sepal_width:
"4.1",
rationale: "predict the iris species. We know that the petal length is 1.5
cm, which is relatively short compared to the other species. The petal width
is 0.1 cm, which is also relatively narrow. The sepal length is 5.2 cm,
which is relatively long, and the sepal width is 4.1 cm, which is also
relatively wide. Based on these characteristics, we can conclude that the
iris species is likely to be setosa.",
answer: “setosa"
—
```

Feedback Gradients (TextGrad) [Yuksekgonul et al., 2024]



You will answer a reasoning question. Think step by step. The last line of your response should be of the following format: 'Answer: $VALUE' where VALUE is a numerical value.

Prompt at initialization (Accuracy = 77.8%)

You will answer a reasoning question. List each item and its quantity in a clear and consistent format, such as '- Item: Quantity'. Sum the values directly from the list and provide a concise summation. Ensure the final answer is clearly indicated in the format: 'Answer: $VALUE' where VALUE is a numerical value. Verify the relevance of each item to the context of the query and handle potential errors or ambiguities in the input. Double-check the final count to ensure accuracy."

Prompt after optimization (Accuracy = 91.9%)

Have an LLM reflect on what can be improved about an output and propagate that feedback back through the prompts in the system.

Image Credit: https://arxiv.org/pdf/2406.07496

Genetic Pareto Prompt Evolution (GEPA) [Agrawal et al., 2026]

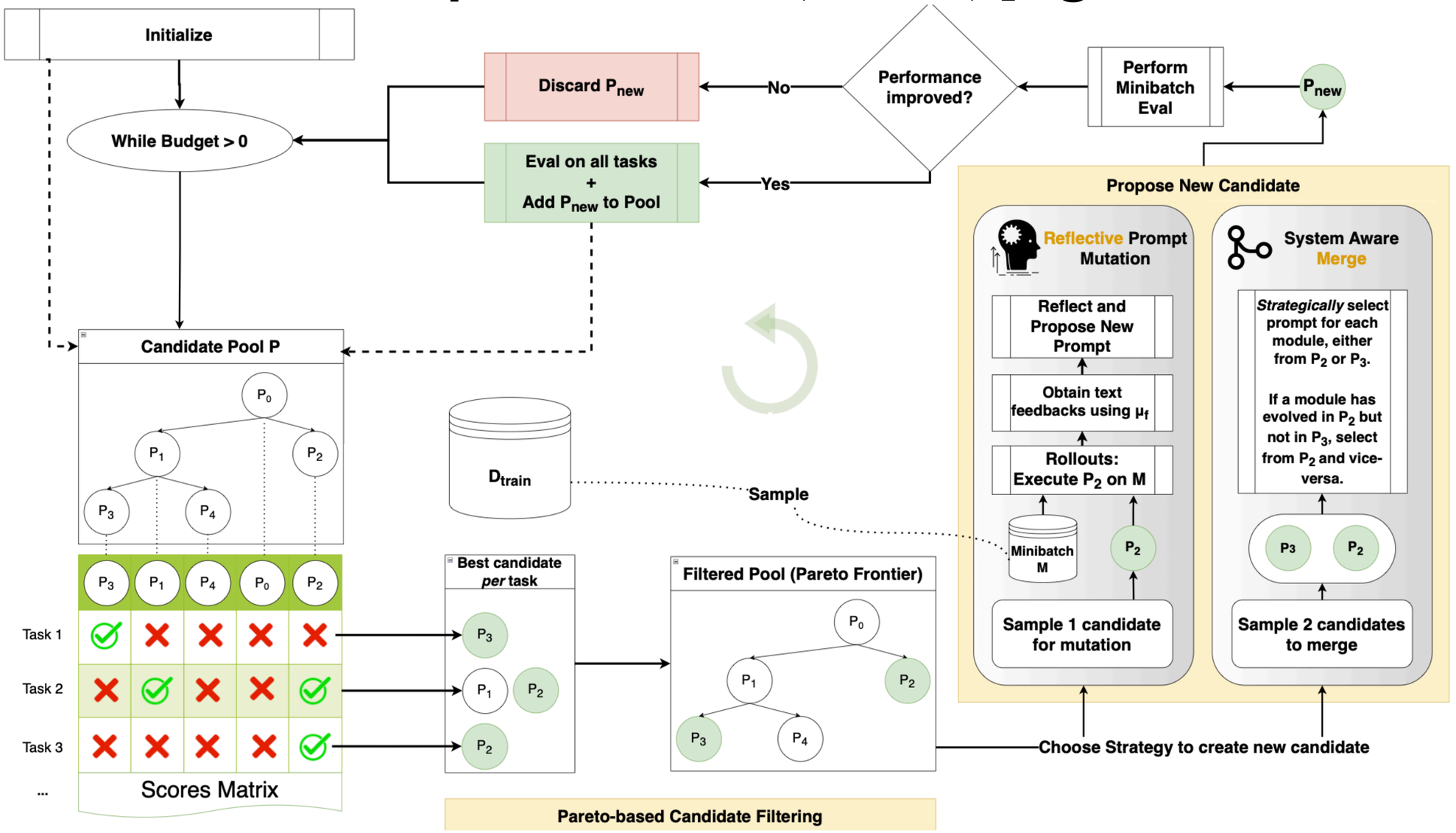
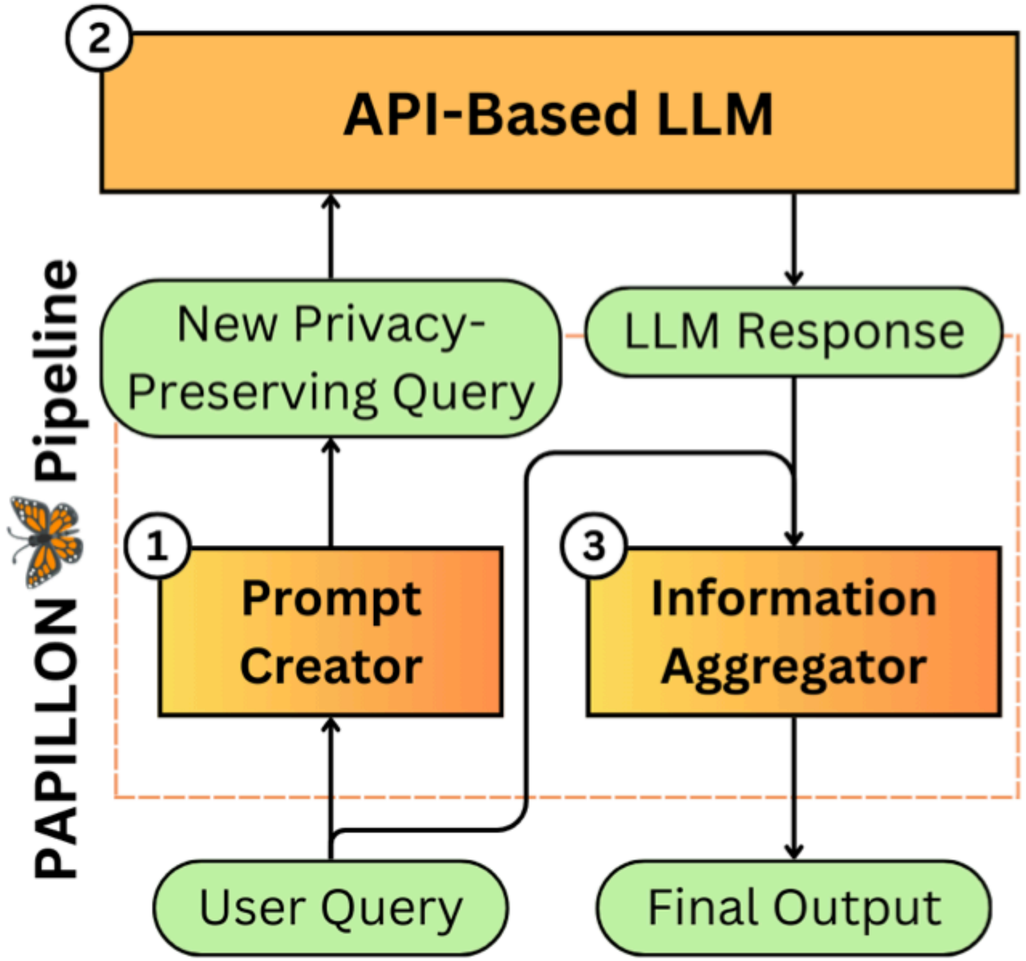


Image Credit: https://arxiv.org/pdf/2507.19457

GEPA Example: PAPILLON [Siyan et al., 2025]



Baseline Prompts from GEPA Paper (not the baseline prompts from the PAPILLON paper)

1

Given a private user query, create a privacy-preserving request for a powerful external LLM.

The LLM may assist without learning private information about the user.

3

Respond to a user query.

For inspiration, we found a potentially related request to a powerful external LLM and its response.

Image Credit: https://arxiv.org/pdf/2410.17127

GEPA Example

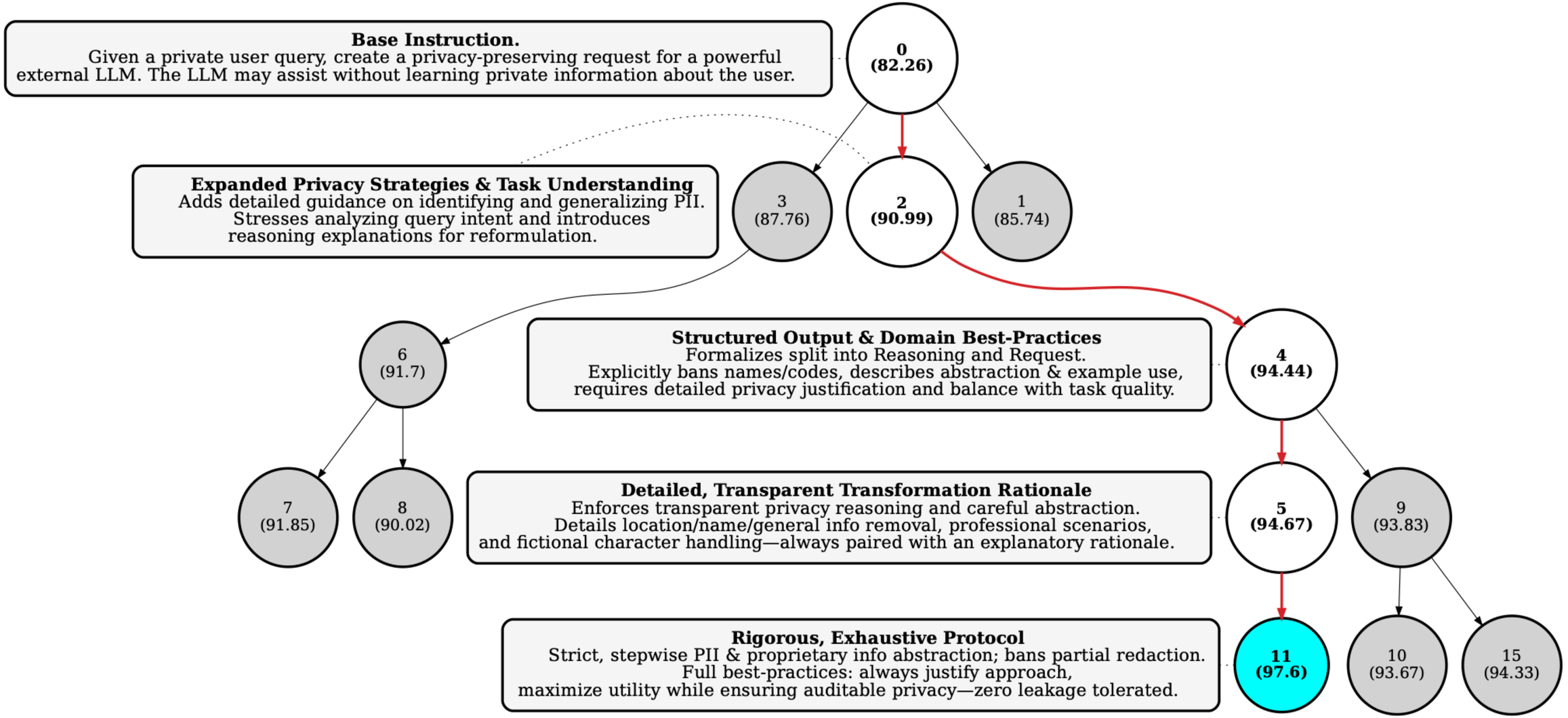


Image Credit: https://arxiv.org/pdf/2507.19457

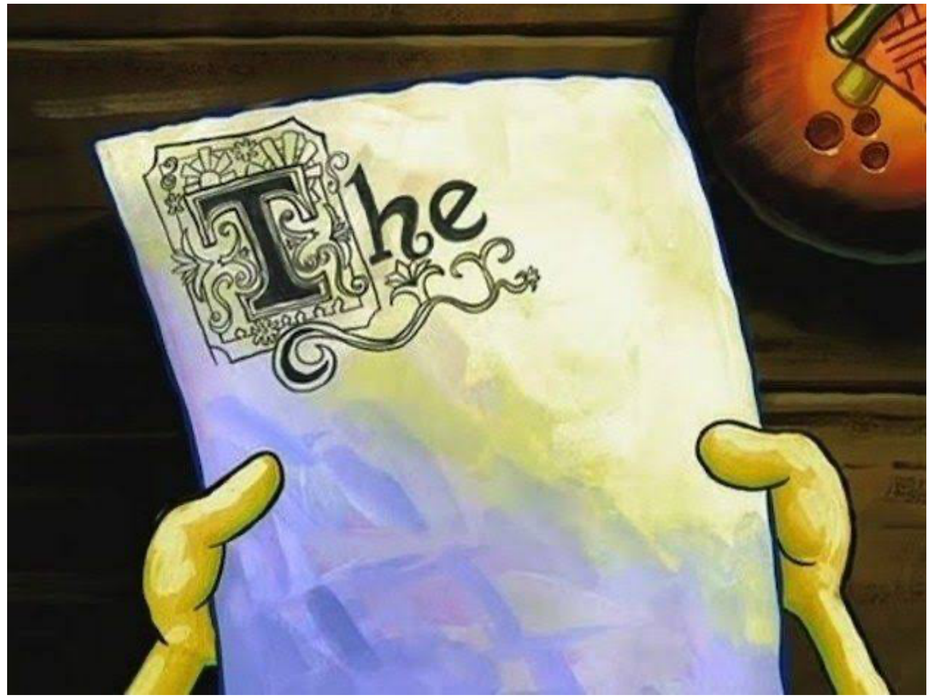
GEPA Results

GPT-4.1 Mini	HotpotQA	IFBench	Hover	PUPA	AIME-2025	LiveBench-Math	Aggregate	Improvement
Baseline	38.00	47.79	46.33	78.57	49.33	58.20	53.03	—
Trace (OptoPrime)	60.33	51.19	46.00	74.18	45.33	60.74	56.30	+3.27
MIPROv2-No-Demos	38.00	52.04	51.33	91.85	48.67	60.97	57.14	+4.11
MIPROv2	58.00	49.15	48.33	83.37	51.33	61.84	58.67	+5.64
TextGrad	62.33	48.64	47.67	85.68	46.67	63.84	59.14	+6.11
GEPA	**69.00**	52.72	51.67	94.47	**59.33**	**64.13**	65.22	+12.19
GEPA+Merge	65.67	**55.95**	**56.67**	**96.46**	**59.33**	**64.13**	**66.36**	**+13.33**
Optimized with Qwen3-8B, evaluated on GPT-4.1-Mini								
GEPA-Qwen-Opt	65.67	49.83	54.67	90.05	52.67	59.31	62.03	+9.00

Reflective prompt evolution with GEPA outperforms baselines across tasks.

Image Credit: https://arxiv.org/pdf/2507.19457

Context Optimization Summary



- **Fewshot Optimization:**
  - **Takeaway:** Bootstrapping reasoning/output and searching for useful few shot examples is a strong pattern for prompt optimization.
- **Reflective Optimization:**
  - **Takeaway:** Running the task and reflecting on what to change, then including that into the prompt is a strong pattern. Evolving the prompt based on improvements to the Pareto frontier also helps.

Weight Optimization

Supervised Finetuning

Recall:
Cross Entropy Loss

1 when this is the token in the data

0 when this is not the token in the data

$$L = -\sum_{i=1}^{V} y_i \log(p_i)$$

log probability of that token

For every token in the vocabulary

Recall:
Instruction Finetuning

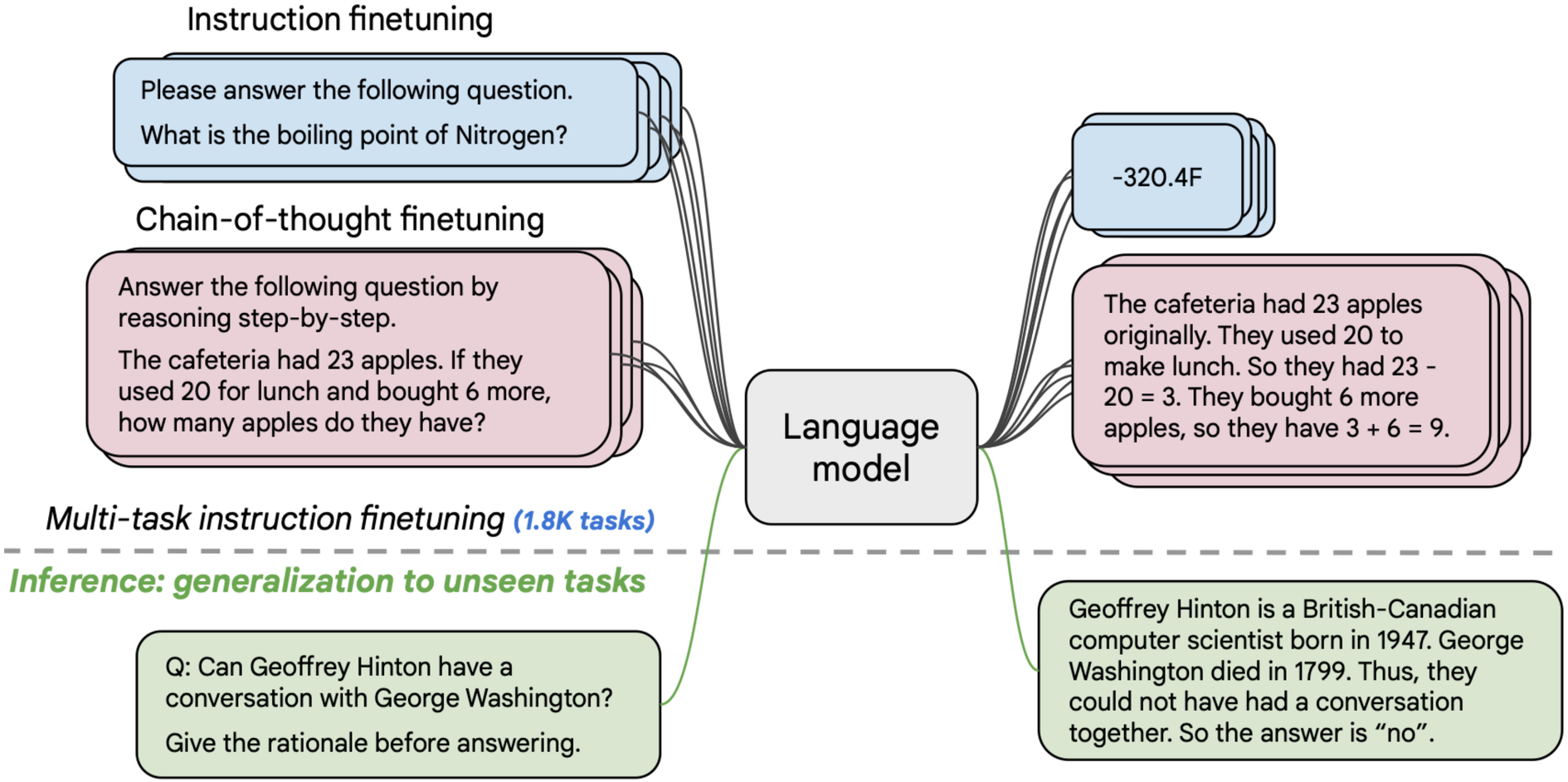
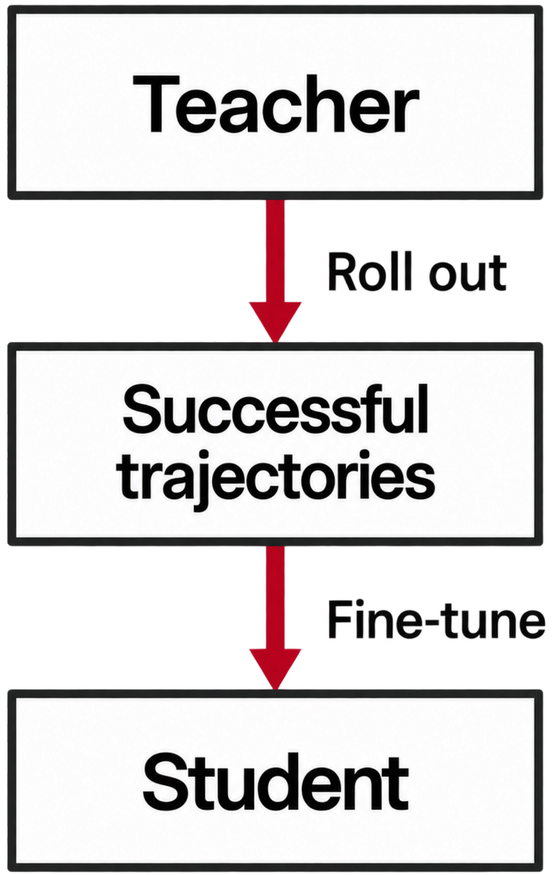


Image Credit: https://arxiv.org/pdf/2210.11416

SFT For Agents

- Rollout teacher on agentic tasks.
- Post-train student models on teacher completed tasks.
- Focus on diversity of tasks and environments.
- Student learns task execution.



OpenThoughts-Agent

task_1087

USER

Locate all files (case-insensitive) whose names end with .example.com under the current directory but ignore anything inside the beta subdirectory; capture both stdout and stderr of the command into a provided capture file for inclusion in the generated task.

AGENT

I need to capture both stdout and stderr to /output/command_capture.txt.

```
find . -type f -iname '*.example.com' ! -path './beta/*' > /output/command_capture.txt 2>&1
wc -c /output/command_capture.txt
```

TERMINAL

```
48 /output/command_capture.txt
```

AGENT

I should verify the contents of the capture file to confirm it contains the expected file listings.

```
cat /output/command_capture.txt
```

TERMINAL

```
./alpha/one.example.com
./alpha/two.example.com
```

AGENT

Both stdout and stderr were captured to /output/command_capture.txt, which contains 48 bytes of content. The capture file exists and contains the expected output.

```
"task_complete": true
```

Have a teacher model perform the task and then train on the outputs.

Also possible to verify the output of the model somehow and train using rejection sampling.

OpenThoughts-Agent [Raoof et al., 2026]



Figure 2: **Six-stage SFT data pipeline for OpenThoughts-Agent.** Each stage is ablated independently in Sections 3.1–3.6.

Stay tuned for the Agent Data lectures!

Image Credit: https://arxiv.org/pdf/2606.24855

OpenThoughts-Agent [Raoof et al., 2026]

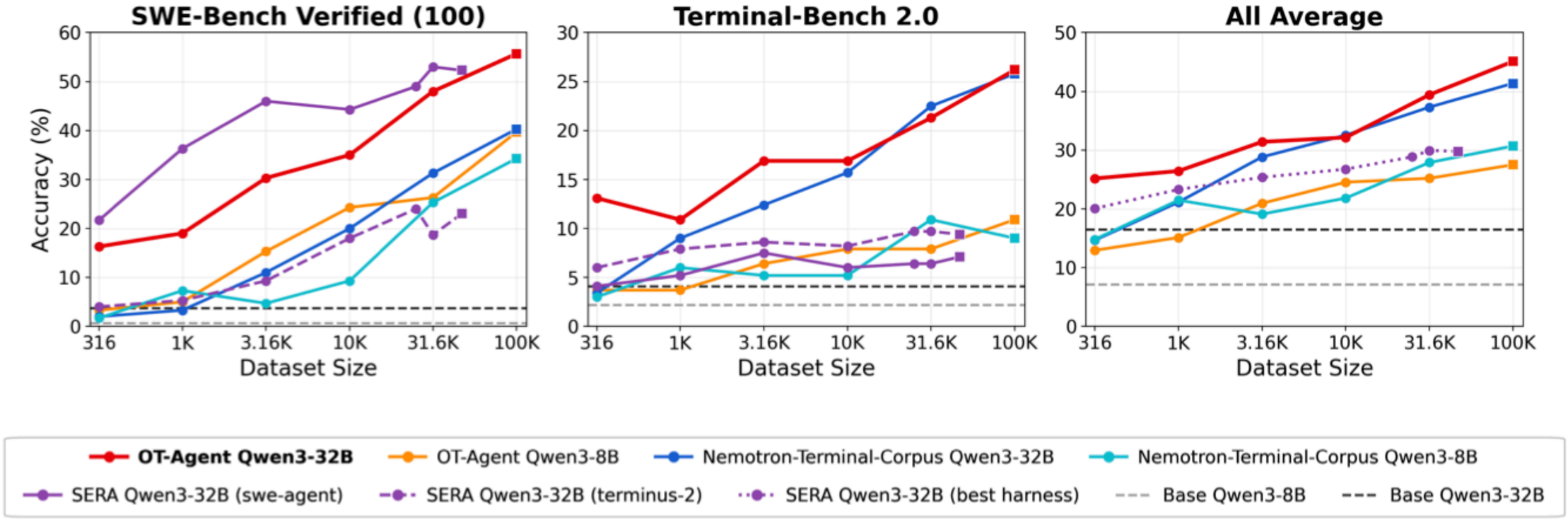
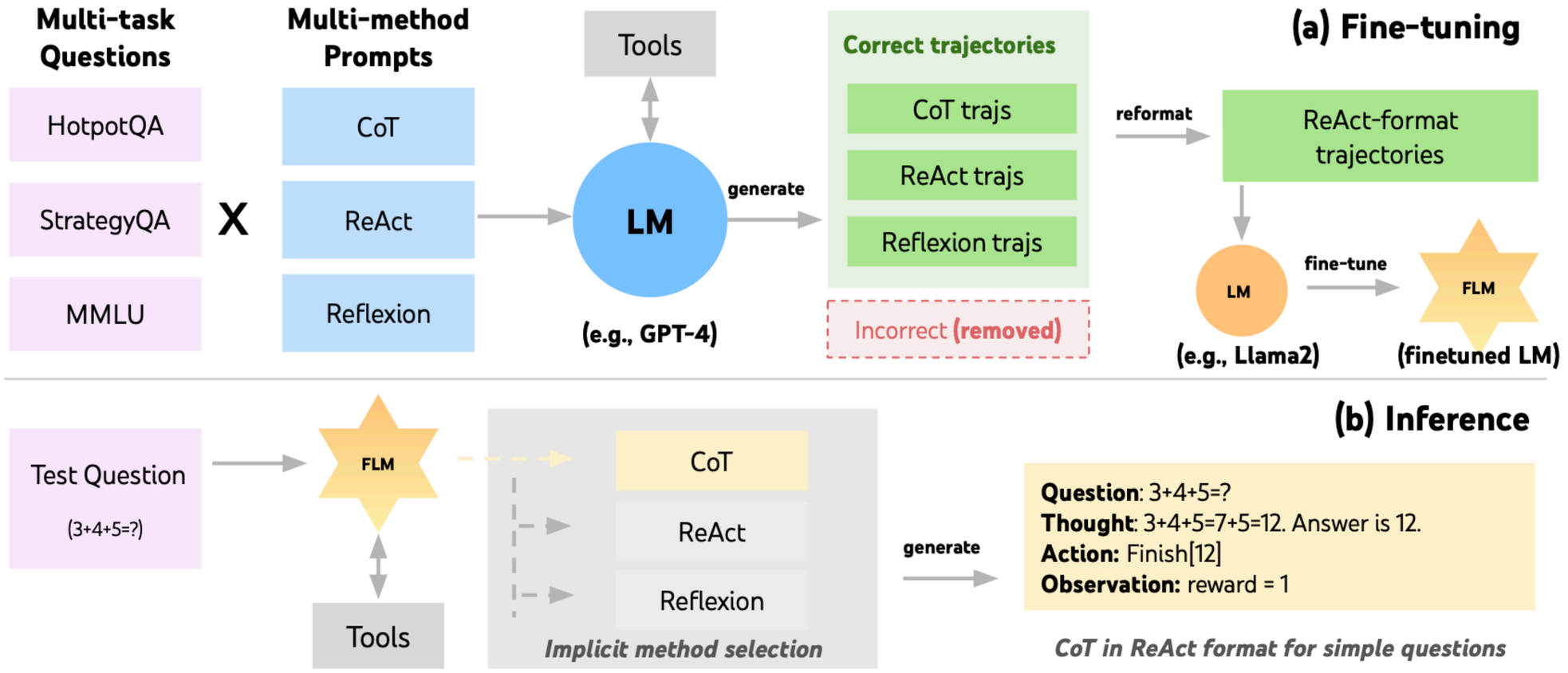


Figure 1: **The OpenThoughts-Agent-SFT dataset leads to SotA performance on Terminal-Bench 2.0 and an 100-subset of SWE-Bench Verified at large dataset scales.** The all-benchmark average is over the seven benchmarks reported in Table 1. Additional notes on SERA below.[1]

Finetuning on rollouts from a strong teacher helps!

Image Credit: https://arxiv.org/pdf/2606.24855

FireAct [Chen et al., 2023]



Finetune LLama2 on GPT-4 rollouts with various agent reasoning design patterns

At inference time allow the model to select how to reason.

Image Credit: https://arxiv.org/pdf/2310.05915

FireAct [Chen et al., 2023]

Table 1: Prompting results.

	Prompt	EM
GPT-4	IO	37.2
	CoT	**45.0**
	ReAct	42.0
GPT-3.5	IO	22.4
	CoT	28.0
	ReAct	**31.4**

Table 2: Prompting vs. fine-tuning, with absolute/relative increases.

	ReAct	FireAct	abs./rel. diff
Llama-2-7B	14.8	26.2	+11.4 / **77%**
Llama-2-13B	21.2	34.4	**+13.1** / 62%
CodeLlama-7B	17.4	27.8	+10.4 / 60%
CodeLlama-13B	20.8	29.0	+8.2 / 39%
CodeLlama-34B	22.2	27.8	+5.6 / 25%
GPT-3.5	**31.4**	**39.2**	+7.8 / 25%

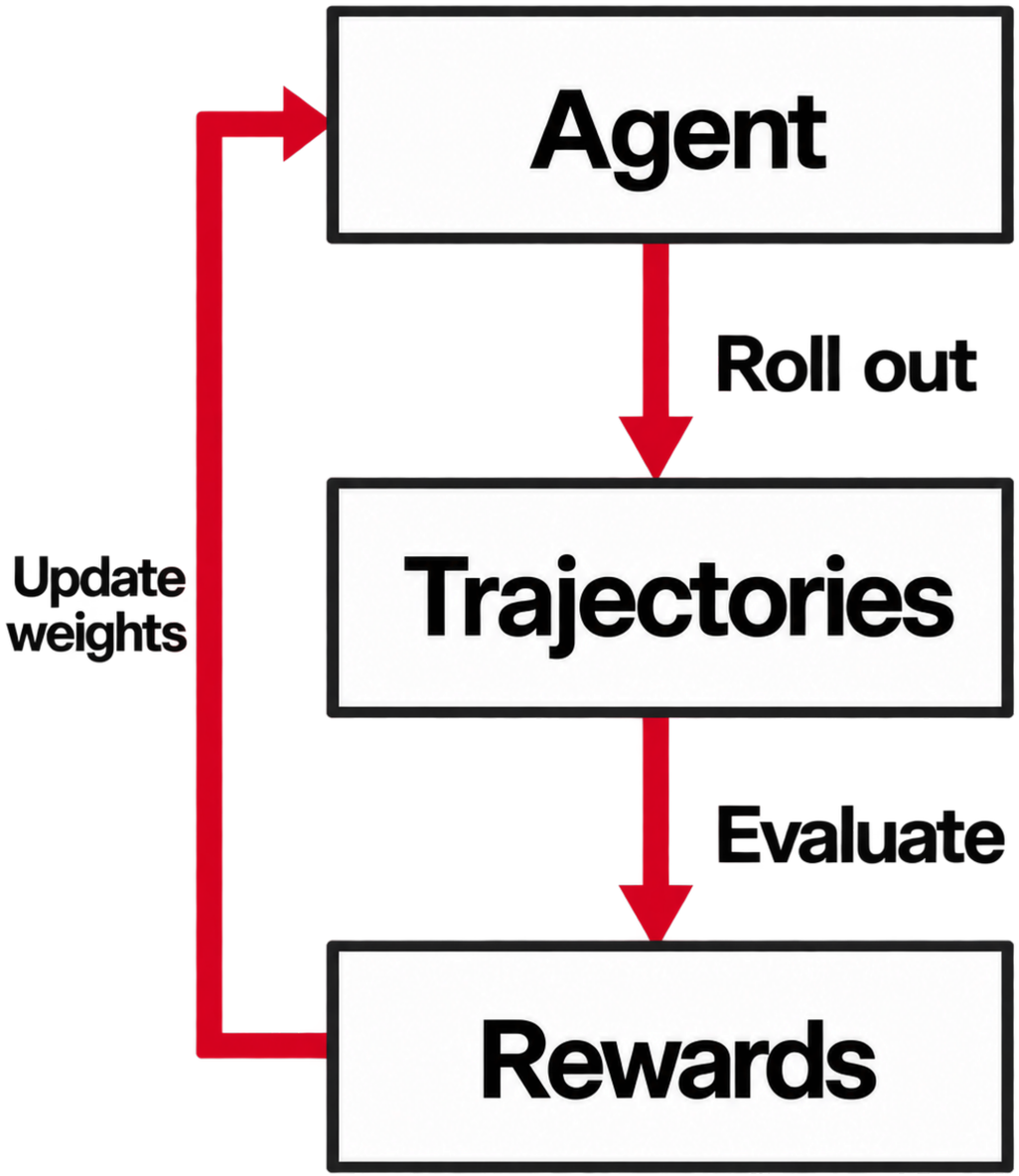
Finetuning LLama2-13B on GPT-4 rollouts makes it a better agent than prompted GPT-3.5

Image Credit: https://arxiv.org/pdf/2310.05915

Reinforcement Learning

RL For Agents

- Rollout **student** on agentic tasks.
- Reward the student for successfully completing the task.
- Focus on varying environments and task complexity.
- Verifiability is key.



OpenThoughts-Agent RL environment task_1087

TASK

Locate all files (case-insensitive) whose names end with .example.com under the current directory but ignore anything inside the beta subdirectory; capture both stdout and stderr of the command into a provided capture file for inclusion in the generated task.

AGENT

```
find . -type f -iname '*.example.com' ! -path './beta/*' > /output/command_capture.txt 2>&1
wc -c /output/command_capture.txt
```

TERMINAL

```
48 /output/command_capture.txt
```

AGENT

I should verify the contents of the capture file to confirm it contains the expected file listings.

```
cat /output/command_capture.txt
```

TERMINAL

```
./alpha/one.example.com
./alpha/two.example.com
```

AGENT

```
"task_complete": true
```

VERIFIER

```
cmp -s "$EXPECTED" "$OUTPUT"
```

Exact match to reference output

Terminal reward **1**

External reward verifies if the task was completed successfully.

Run several times within a group and several groups to a batch.

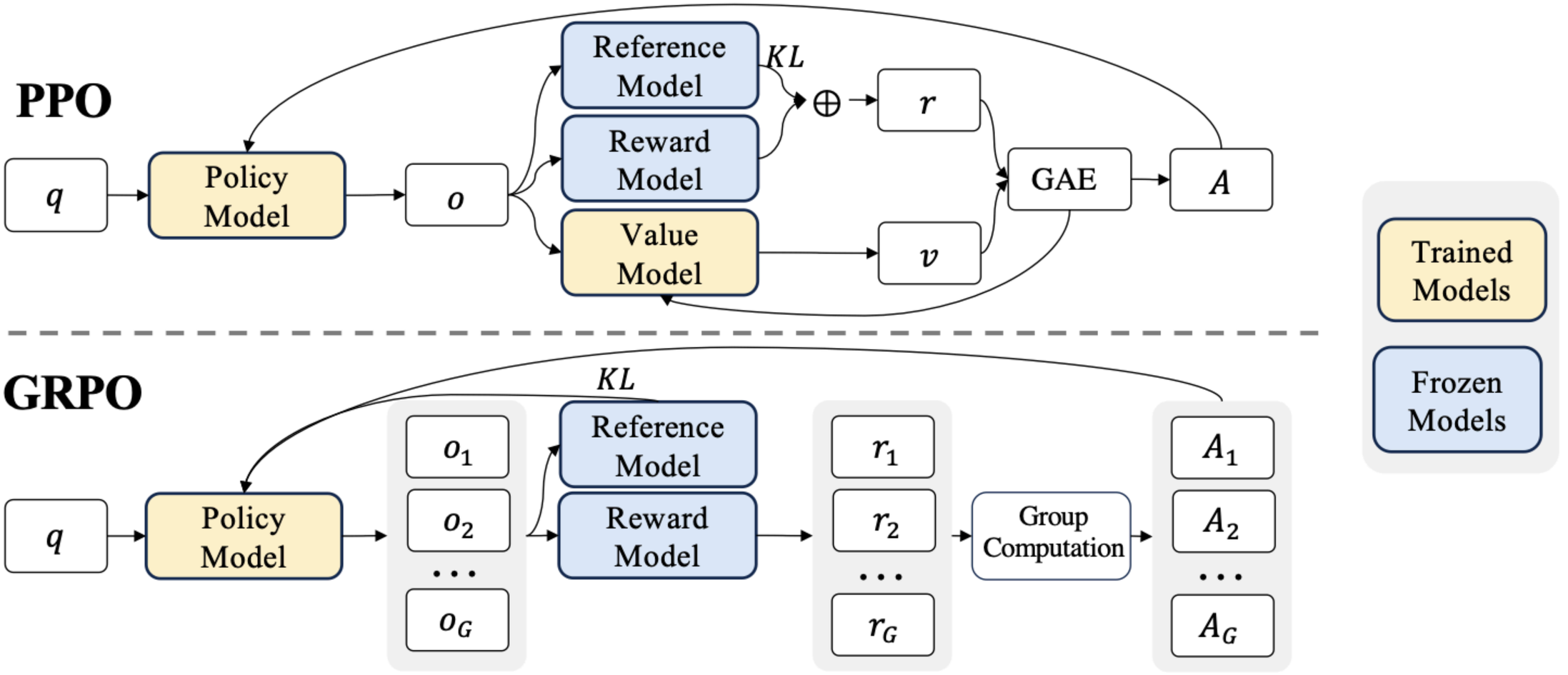


Figure 4 | Demonstration of PPO and our GRPO. GRPO foregoes the value model, instead estimating the baseline from group scores, significantly reducing training resources.

Image Credit: https://arxiv.org/pdf/2402.03300

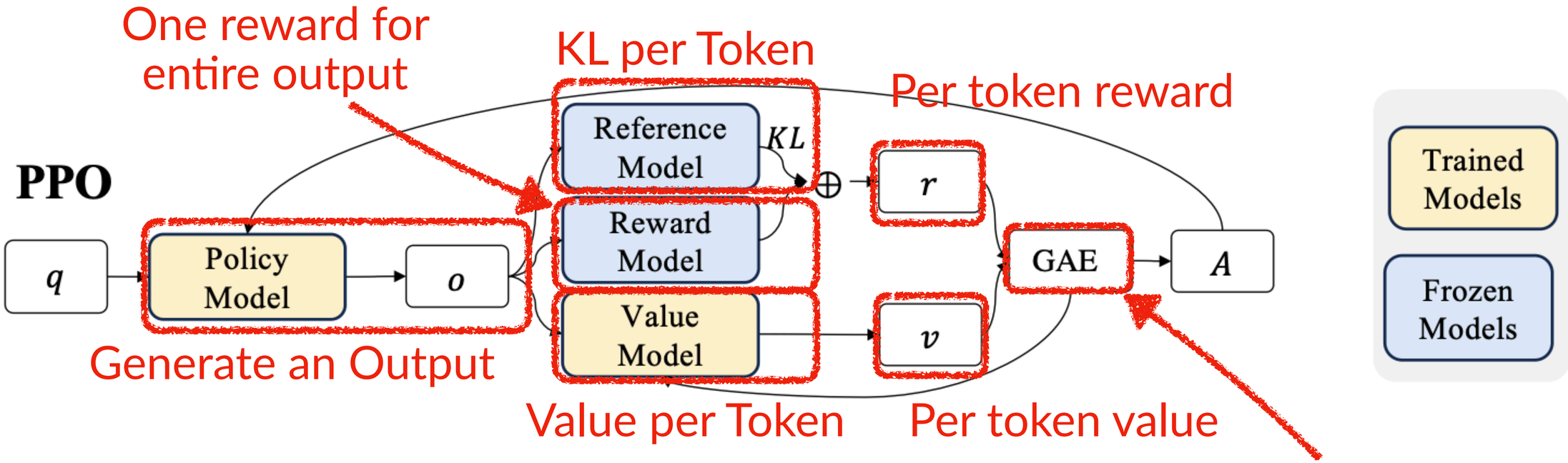
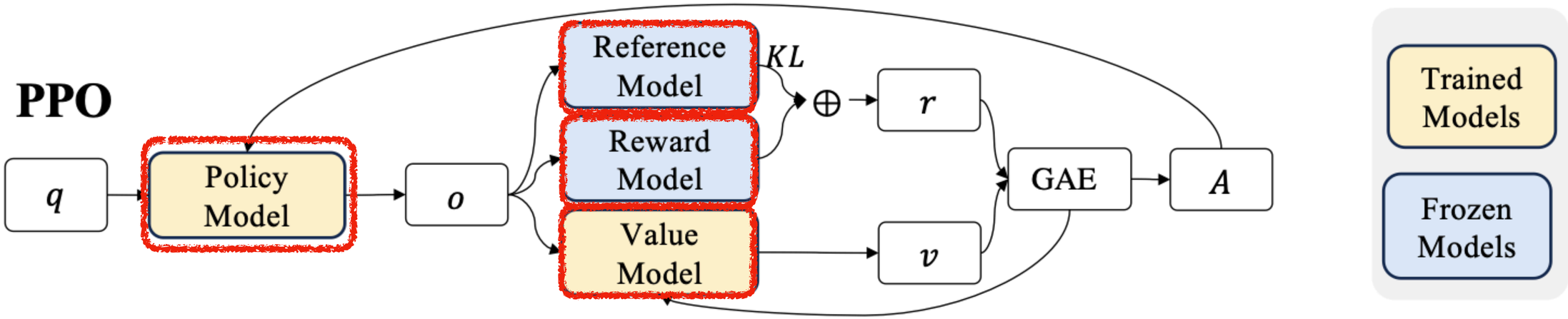
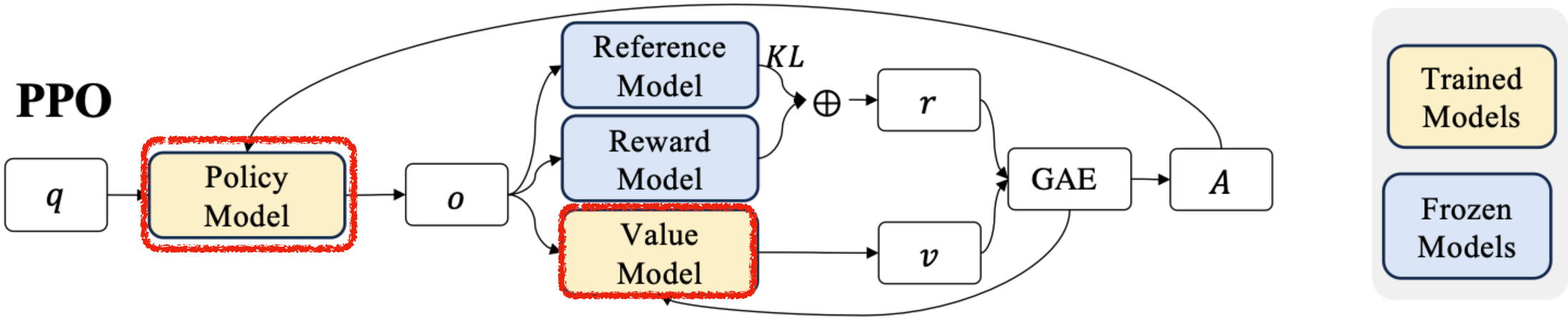


Image Credit: https://arxiv.org/pdf/2402.03300



1. **Four Models** to keep in memory!

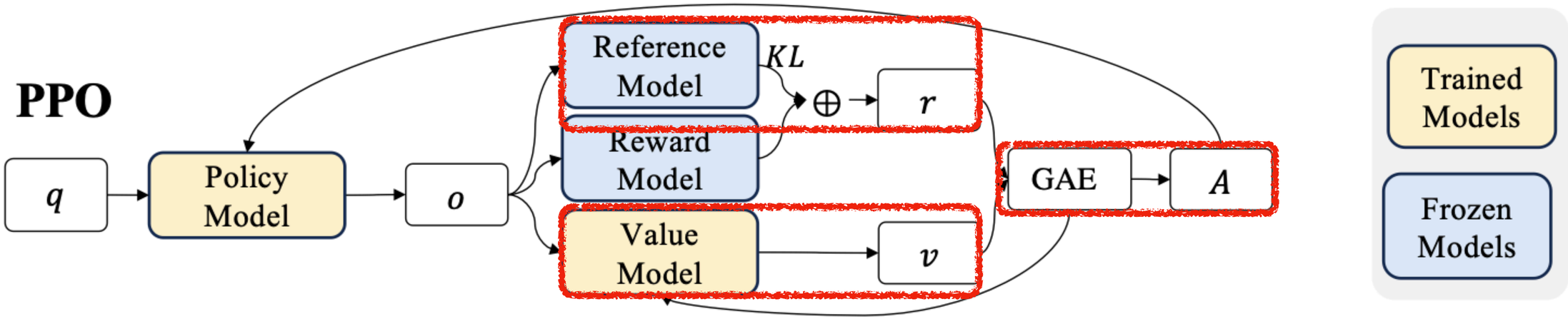
Image Credit: https://arxiv.org/pdf/2402.03300



1. **Four Models** to keep in memory!

2. **Two Models** to train every step!

Image Credit: https://arxiv.org/pdf/2402.03300



1. **Four Models** to keep in memory!

2. **Two Models** to train every step!

3. Several (expensive) **per token steps**!

Image Credit: https://arxiv.org/pdf/2402.03300

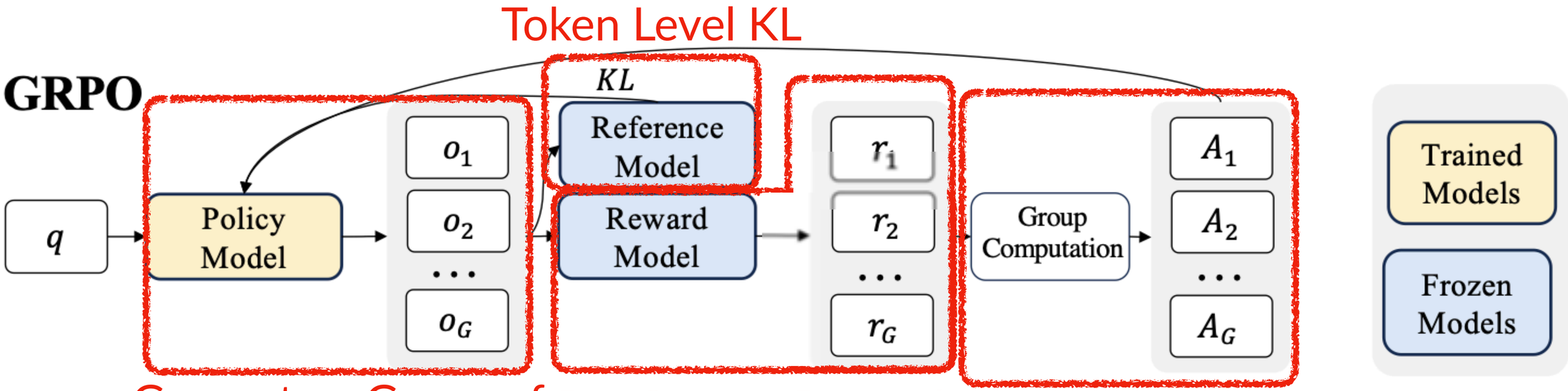
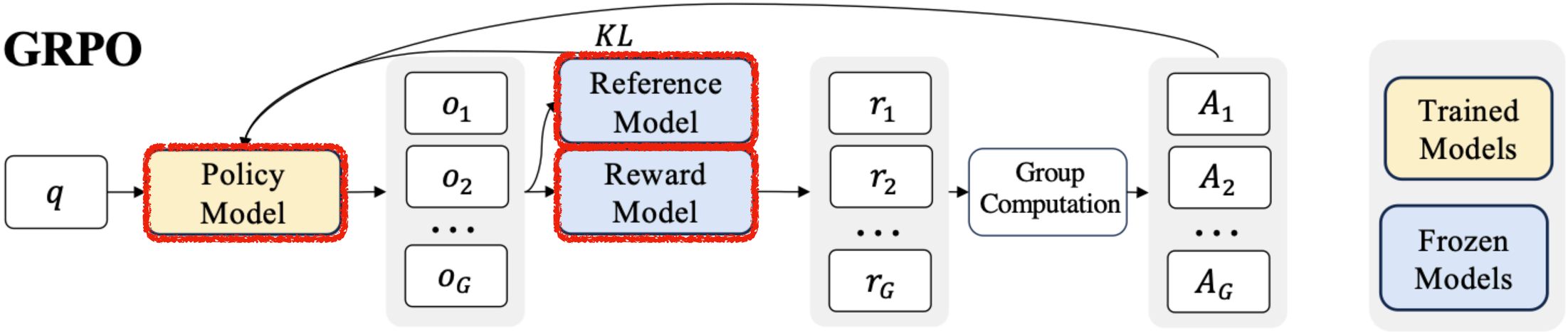
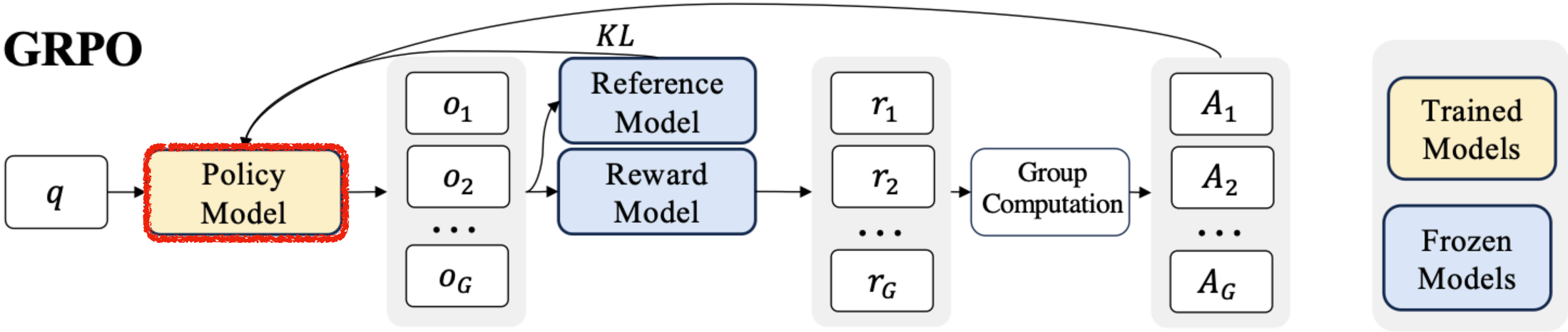


Image Credit: https://arxiv.org/pdf/2402.03300



1. **Three Models** to keep in memory! Also "Reward Model" doesn't have to be a model.

Image Credit: https://arxiv.org/pdf/2402.03300

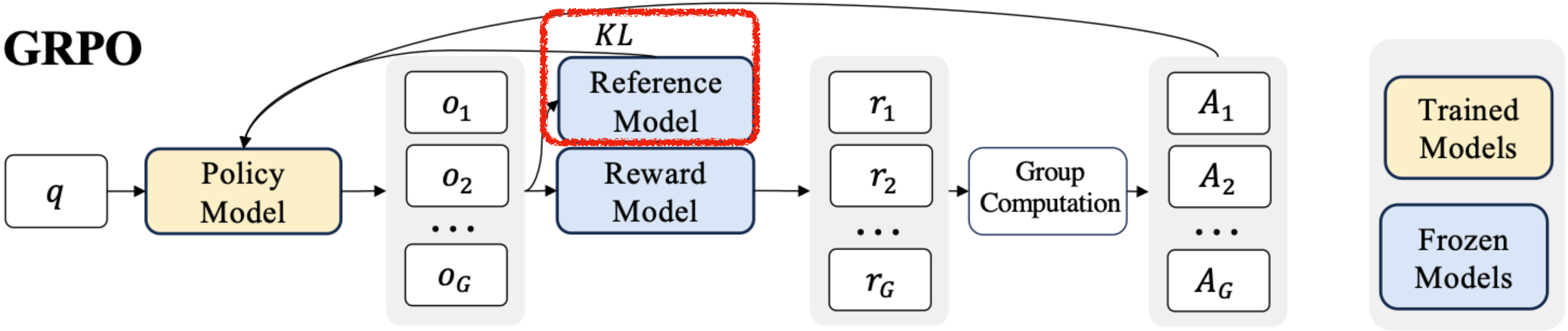


1. **Three Models** to keep in memory! Also "Reward Model" doesn't have to be a model.

2. **One Model** to train every step!

Image Credit: https://arxiv.org/pdf/2402.03300

Token Level KL

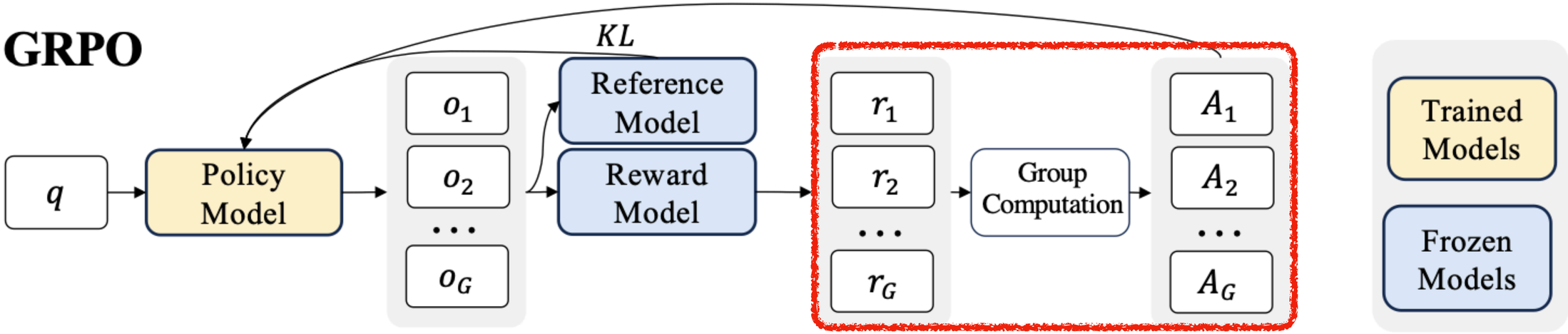


1. **Three Models** to keep in memory! Also "Reward Model" doesn't have to be a model.

2. **One Model** to train every step!

3. **Only KL** is per token, everything else is sequence level!

Image Credit: https://arxiv.org/pdf/2402.03300

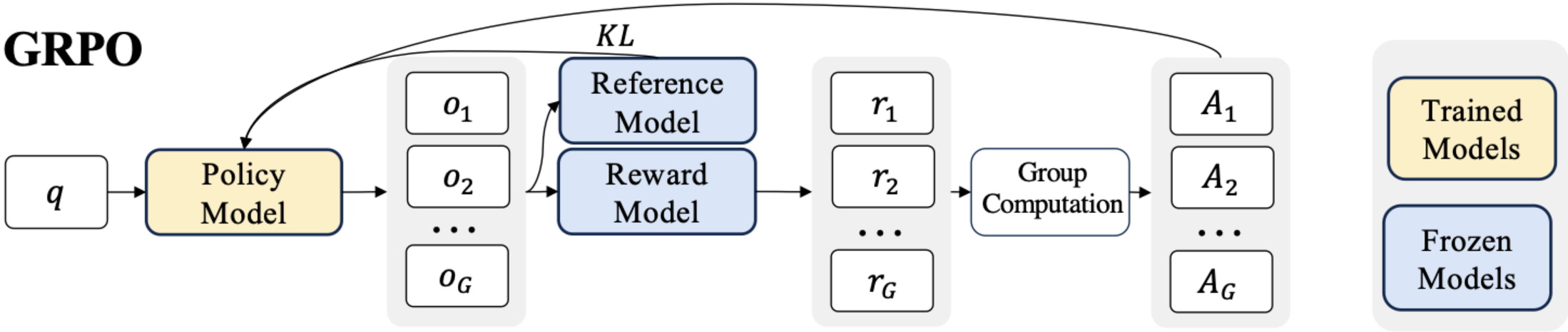


Advantage Calculation

$$\hat{A}_{i,t} = \frac{R_i - \text{mean}(\{R_1, \ldots, R_G\})}{\text{std}(\{R_1, \ldots, R_G\})}$$

Compute the z-score for each output reward!

Image Credit: https://arxiv.org/pdf/2402.03300



Expectation over a prompt pulled from our dataset, and a group of sampled outputs for that prompt.

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(O|q)]$$

Don't deviate too far from reference

$$\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left\{\min\left[\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q,o_{i,<t})}\hat{A}_{i,t}, \text{clip}\left(\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q,o_{i,<t})}, 1-\varepsilon, 1+\varepsilon\right)\hat{A}_{i,t}\right] - \beta\mathbb{D}_{KL}\left[\pi_\theta||\pi_{ref}\right]\right\}$$

Avg over group

Avg over tokens in response

How important is this token based on our new policy compared to our old

Advantage

Don't over update on any one token.

Image Credit: https://arxiv.org/pdf/2402.03300

Real World Example

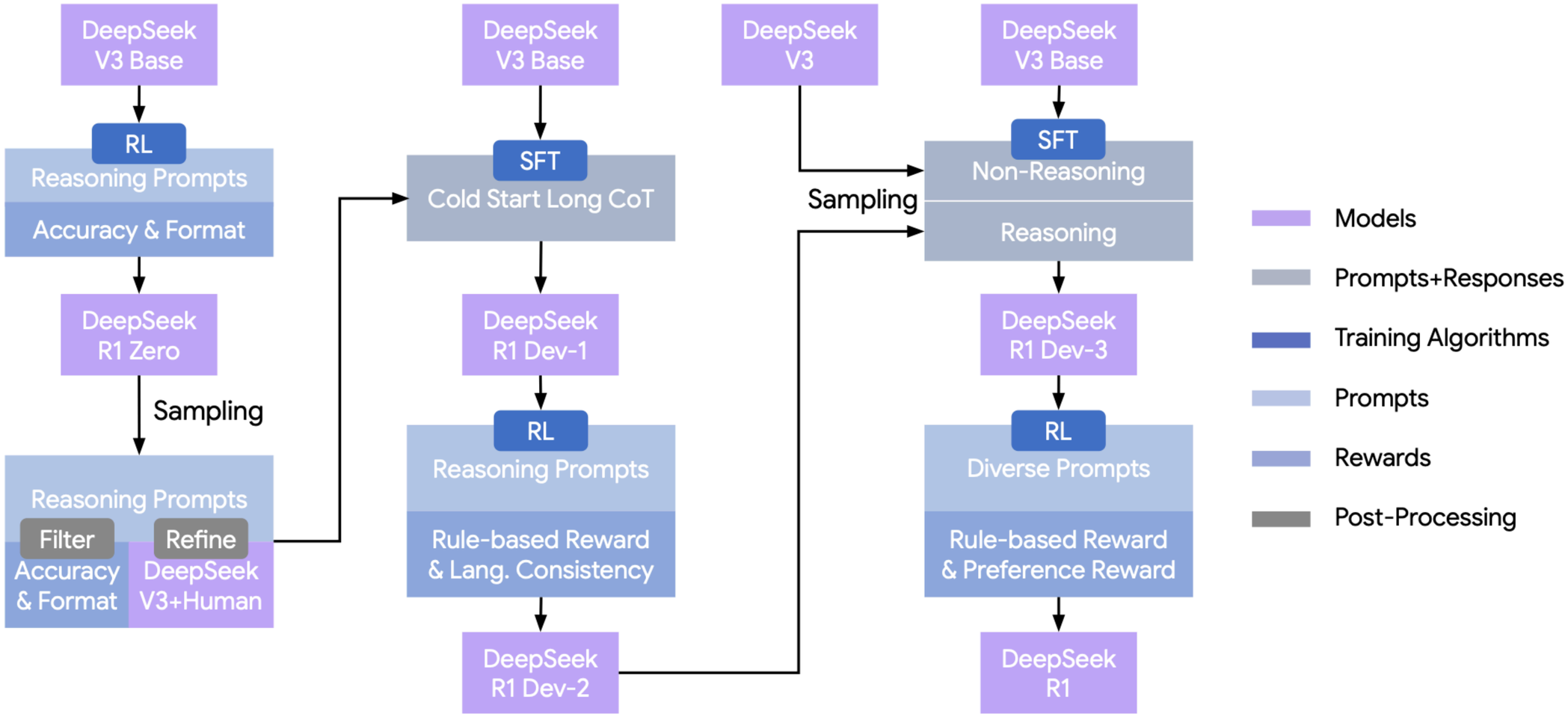


Image Credit: https://arxiv.org/pdf/2501.12948

GRPO vs PPO for DeepSeekR1

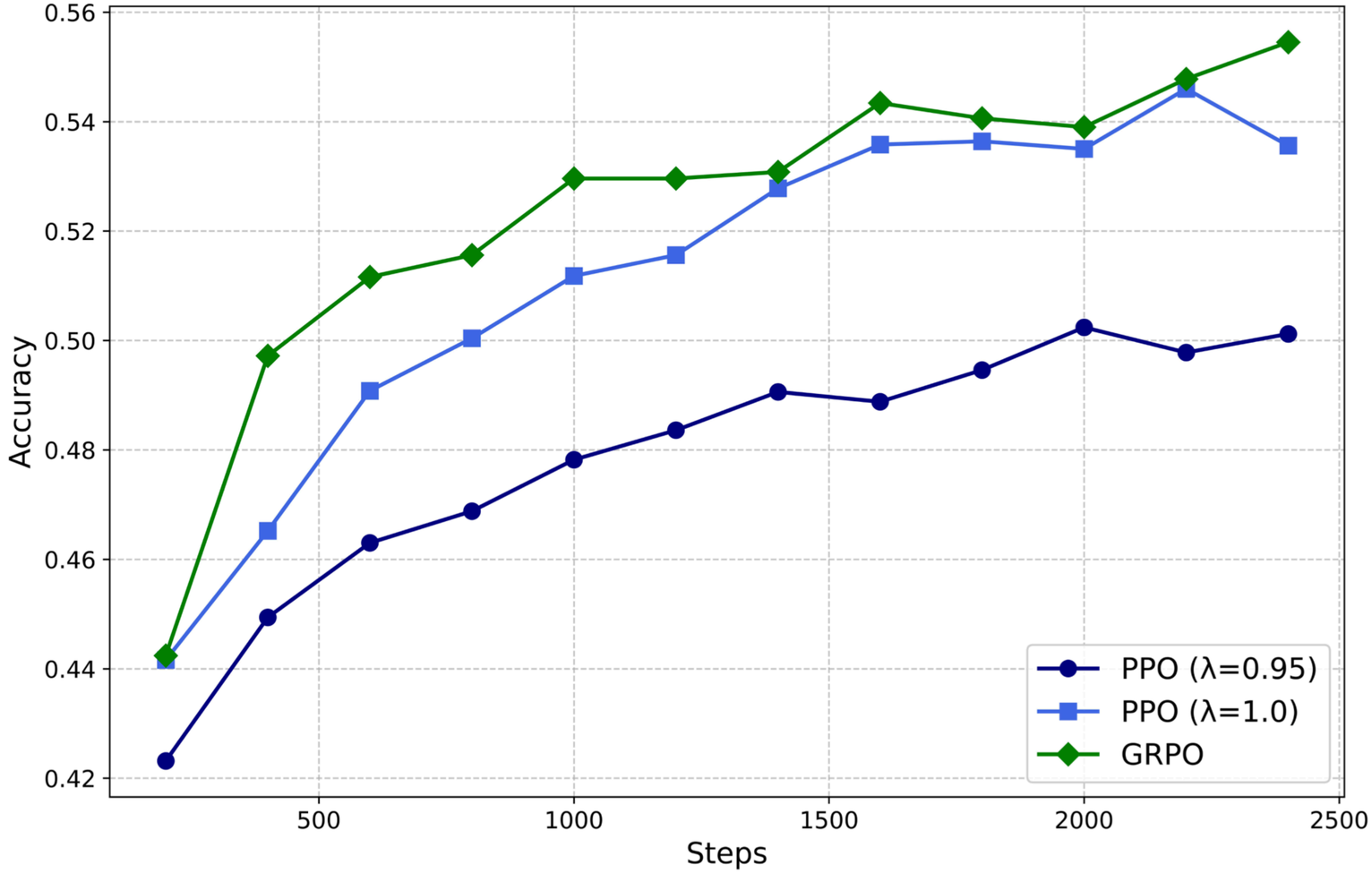


Figure 4 | Performance of PPO and GRPO on the MATH task.

Image Credit: https://arxiv.org/pdf/2501.12948

On-Policy Distillation



It's handy to get chess coaching on a game you actually played rather than a game you just observed.

Image Credit: https://thinkingmachines.ai/blog/on-policy-distillation/

SFT vs RL vs On Policy Distillation

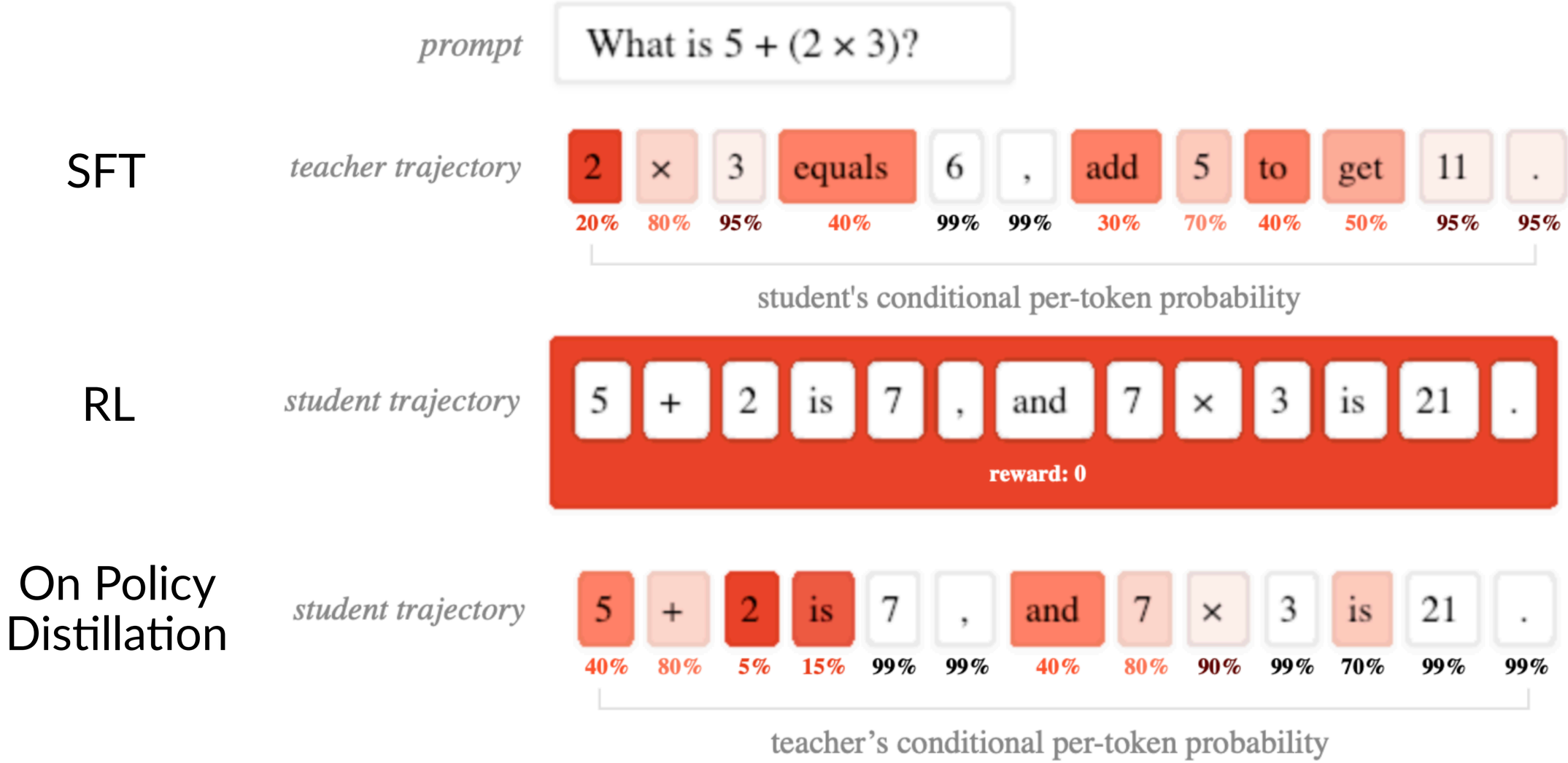


Image Credit: https://thinkingmachines.ai/blog/on-policy-distillation/

SFT vs RL vs On Policy Distillation

Method	Sampling	Reward signal
Supervised finetuning	off-policy	dense
Reinforcement learning	on-policy	sparse
On-policy distillation	on-policy	dense

Image Credit: https://thinkingmachines.ai/blog/on-policy-distillation/

On Policy Distillation Loss

$$\mathrm{KL}\left(\pi_\theta || \pi_{\text{teacher}}\right) = \mathbb{E}_{x\sim\pi_\theta}\left[\log \pi_\theta(x_{t+1}|x_{1..t}) - \log \pi_{\text{teacher}}(x_{t+1}|x_{1..t})\right]$$

Expectation over response sampled from policy

Student logprobs minus teacher logprobs for each token

Image Credit: https://thinkingmachines.ai/blog/on-policy-distillation/

On Policy Distillation Example

Choose the most realistic answer. Beth places four whole ice cubes in a frying pan at the start of the first minute, then five at the start of the second minute and some more at the start of the third minute, but none in the fourth minute. If the average number of ice cubes per minute placed in the pan while it was frying a crispy egg was five, how many whole ice cubes can be found in the pan at the end of the third minute?

A. 30 B. 0 C. 20 D. 10 E. 11 F. 5

(Start of assistant response truncated)

Step 3: Total ice cubes in the pan **at the end of the third minute**

This means all ice cubes placed **by the end of minute 3** — that is, minute 1, 2, and 3.

So:

4 (min 1) + 5 (min 2) + 11 (min 3) = **20**

Note: The ice cubes are not removed during the process — the pan is "frying a crispy egg", but we are only counting how many were added. It seems the ice cubes stay in the pan after being added.

Based on the context, we assume that ice cubes are not melted or removed during the frying process — otherwise, we would have to account for melting, which is not mentioned.

Therefore, the total number of ice cubes **in the pan at the end of the third minute** is:

4 + 5 + 11 = **20**

Final Answer:

C. 20

Correct answer: C. 20<|im_end|>

Image Credit: https://thinkingmachines.ai/blog/on-policy-distillation/

Real Model Example

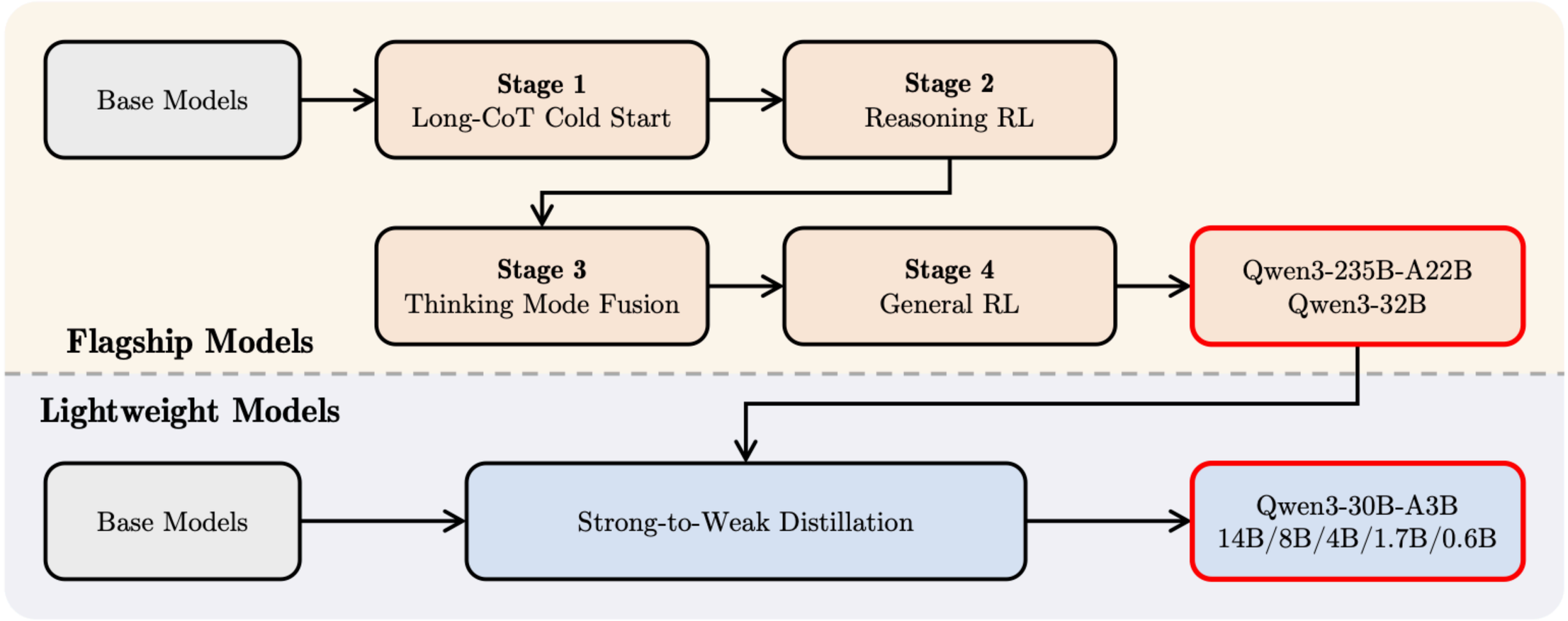


Figure 1: Post-training pipeline of the Qwen3 series models.

Image Credit: https://thinkingmachines.ai/blog/on-policy-distillation/

On Policy Distillation Results

Method	AIME'24	GPQA-Diamond	GPU Hours
Off-policy distillation	55.0%	55.6%	Unreported
+ Reinforcement learning	67.6%	61.3%	17,920
+ On-policy distillation	74.4%	63.3%	1,800

Figure 8: From Qwen3 Technical Report, Table 21.

On-policy distillation was more efficient and produced better models than RL.

Image Credit: https://thinkingmachines.ai/blog/on-policy-distillation/

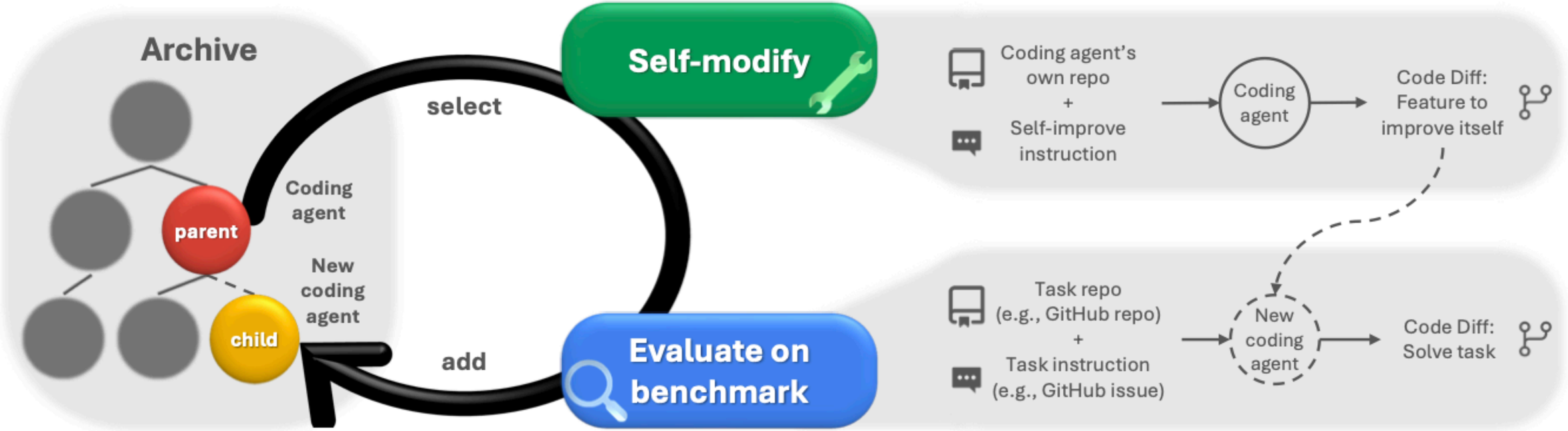
Weight Optimization Summary

Method	Trajectory source	Learning signal	Training objective
Supervised fine-tuning	Collected demonstrations	Reference tokens and actions	Maximize the likelihood of demonstrated behavior
Reinforcement learning	Student-generated trajectories	Task rewards	Maximize expected reward
On-policy distillation	Student-generated trajectories	Teacher token probabilities at student-generated prefixes	Minimize divergence from the teacher on student-generated trajectories

Optimization with Agents

Self Improvement

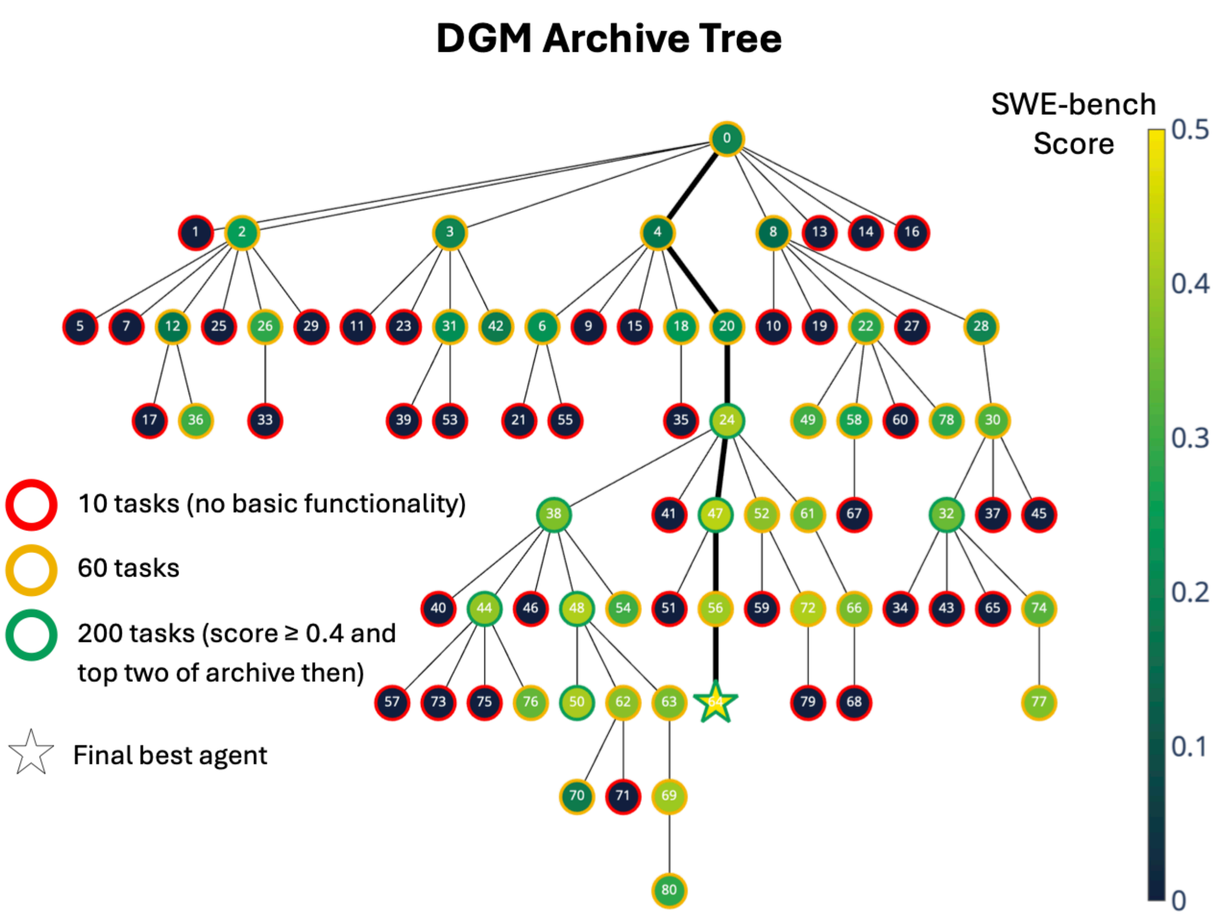
Darwin Gödel Machine [Zhang et al., 2026]



Select parents to self-modify proportionally to performance and inversely proportional to the number of children.

Image Credit: https://arxiv.org/pdf/2505.22954

Darwin Gödel Machine [Zhang et al., 2026]



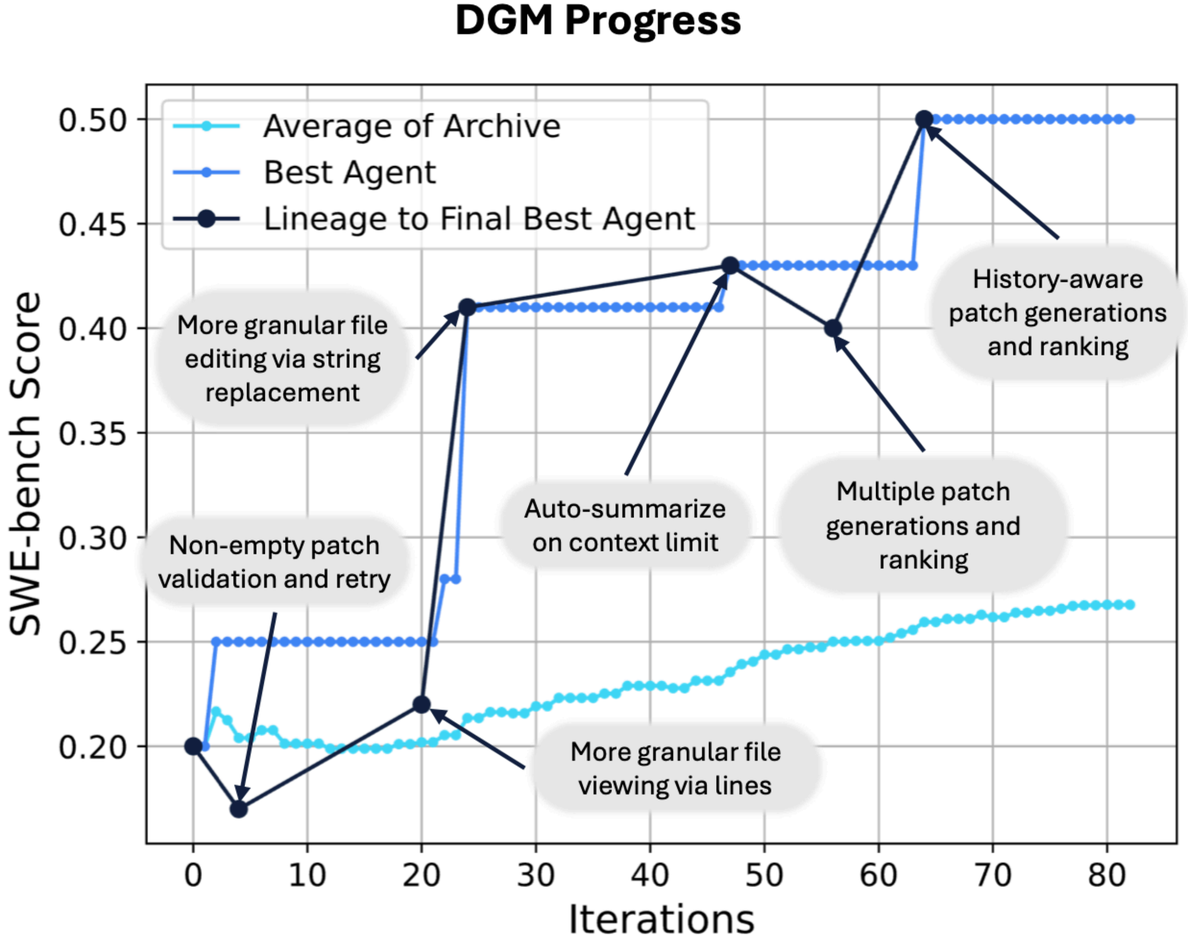
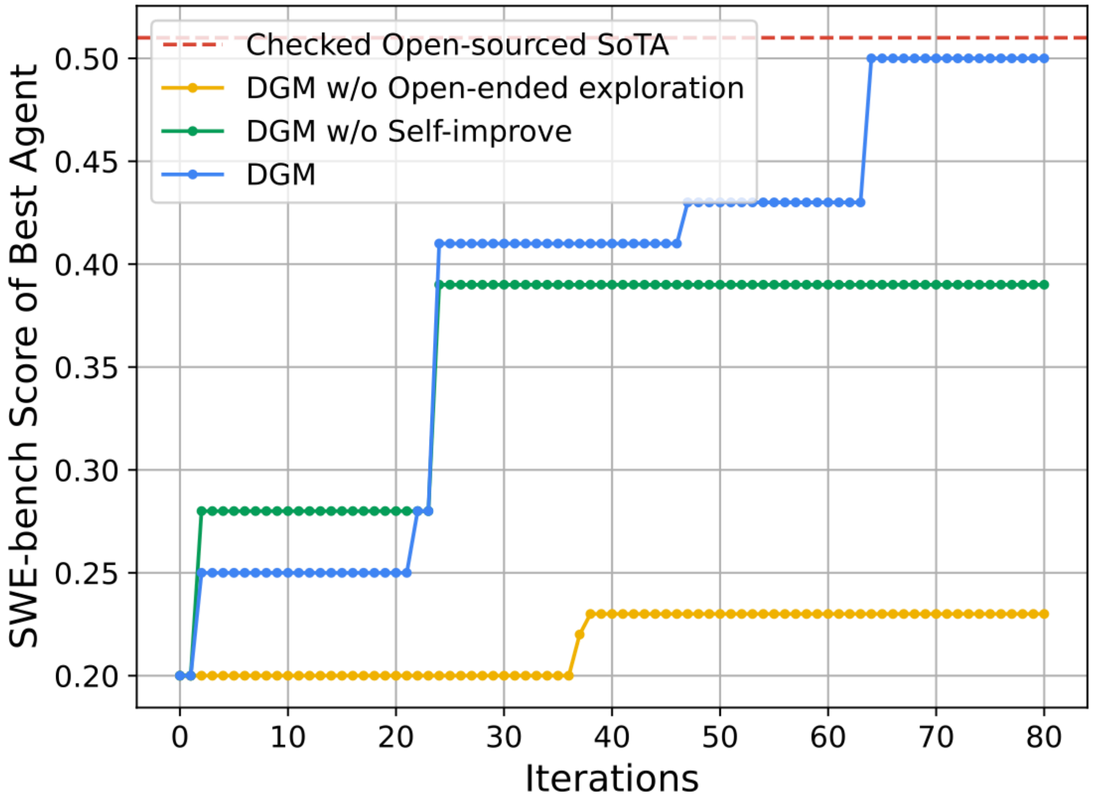
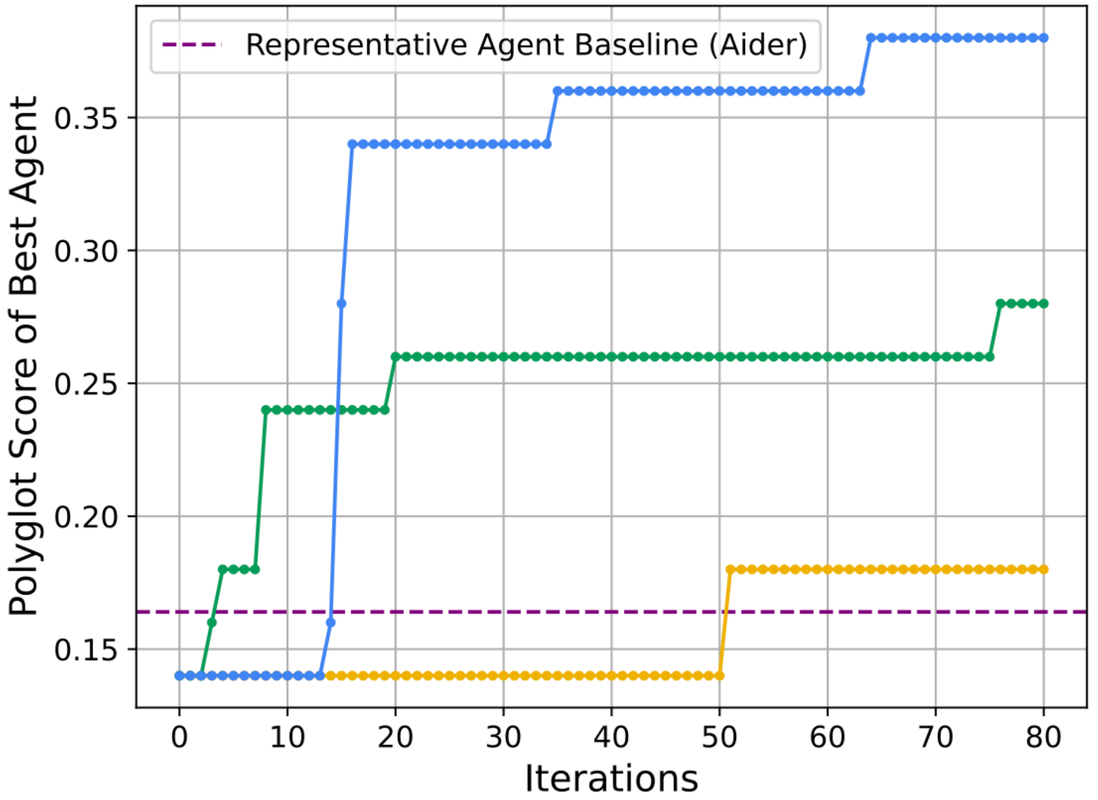


Image Credit: https://arxiv.org/pdf/2505.22954

Darwin Gödel Machine [Zhang et al., 2026]

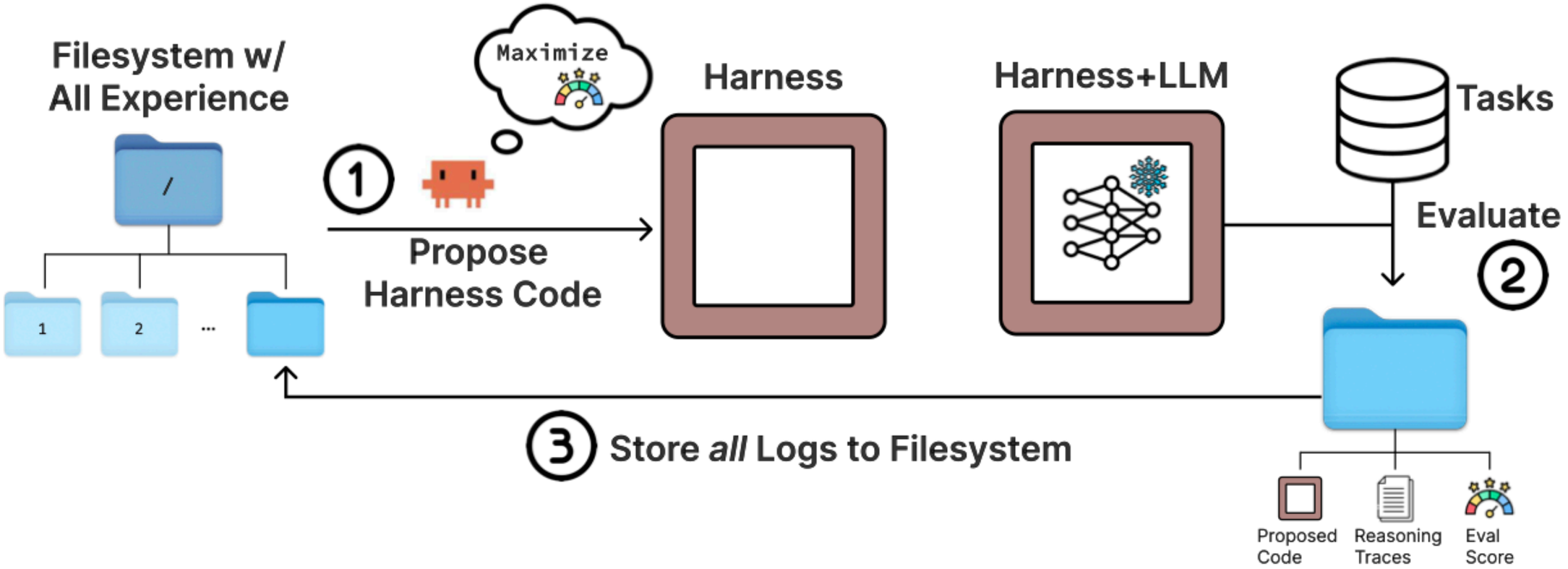




Both the self-improvement (parent modifies itself rather than meta-agent) AND the exploration (returning to prior parents instead of always building on latest) were critical.

Image Credit: https://arxiv.org/pdf/2505.22954

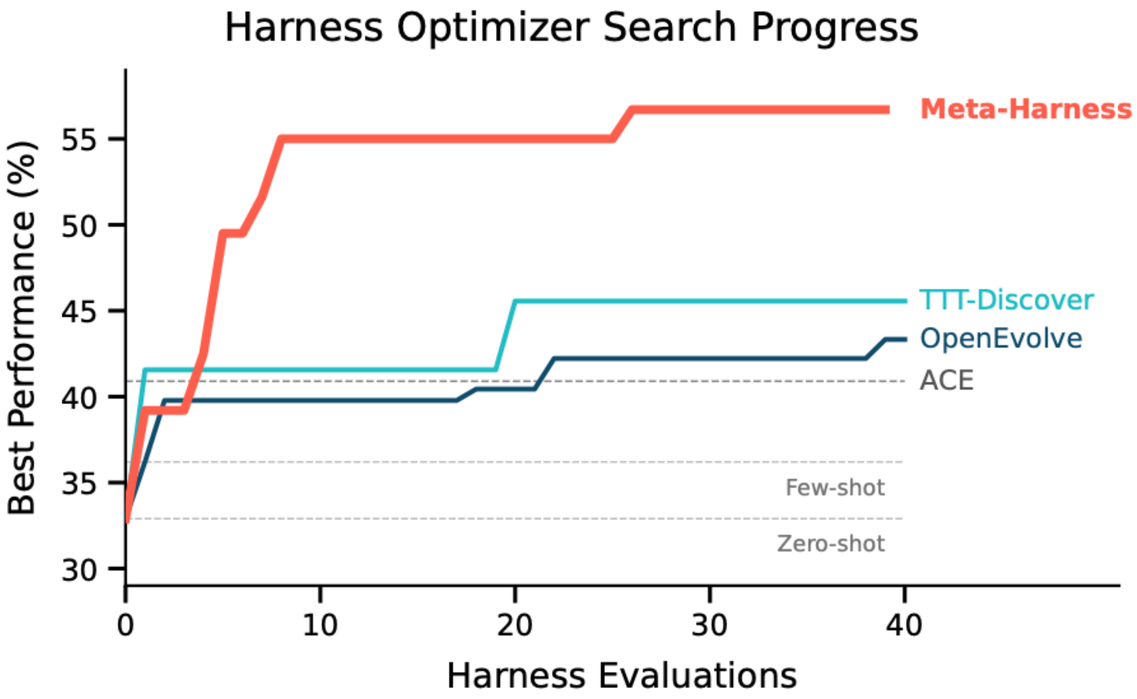
Meta-Harness

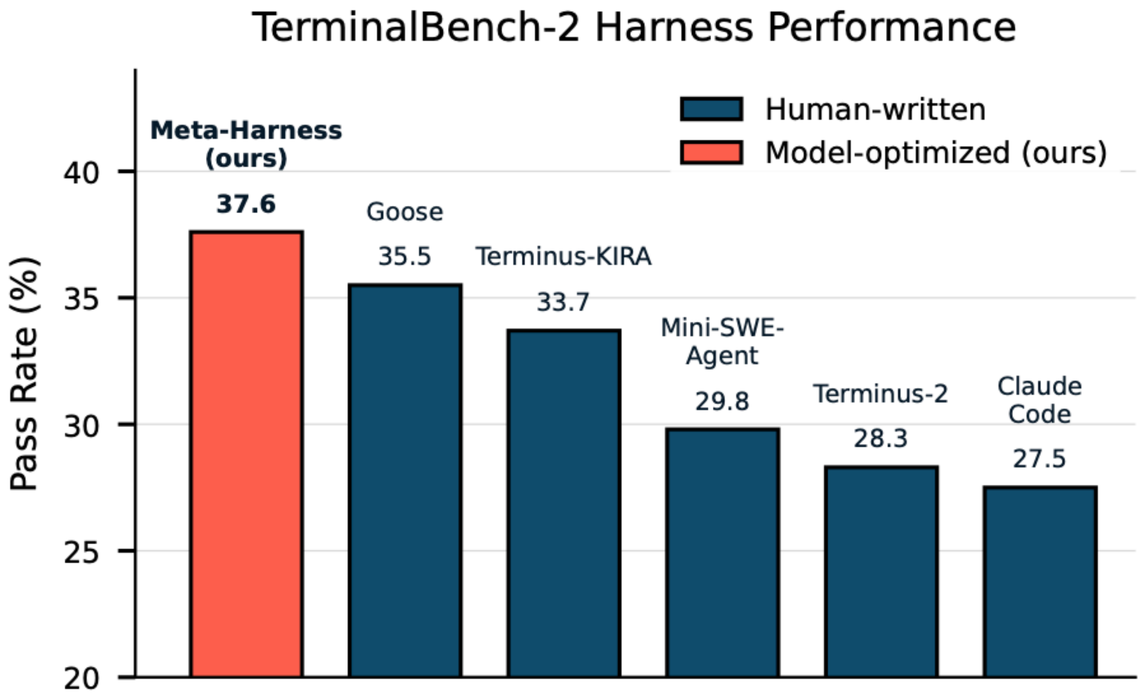


Allow a meta agent to see ALL prior experience and decide what to edit next.

Image Credit: https://arxiv.org/pdf/2603.28052

Meta-Harness



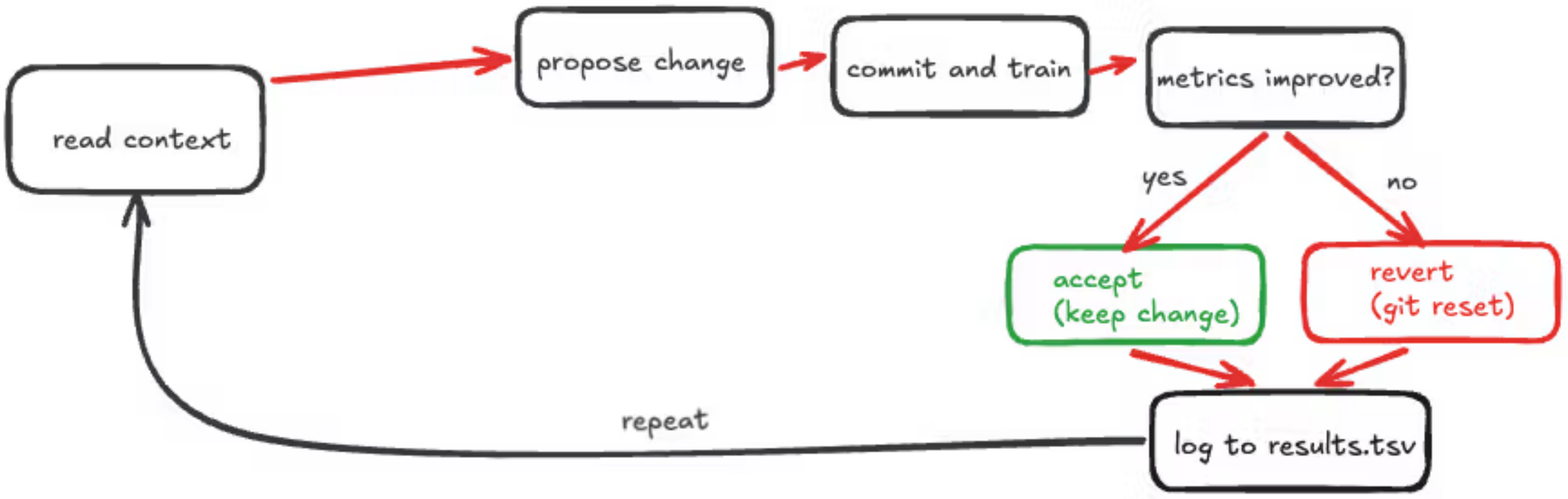


Over time the meta-agent can use the context to build better and better harnesses for agents to solve domain-targeted problems.

Image Credit: https://arxiv.org/pdf/2603.28052

Auto Research (Optimize Anything)

Auto-Research



Give agent access to repository. Let it brainstorm, write a change. Run experiments to see if the change was positive. If metrics improve, build upon it, otherwise reset. Log the findings.

Image Credit: https://www.cerebras.ai/blog/how-to-stop-your-autoresearch-loop-from-cheating

Auto-Research Prompt

The experiment loop

The experiment runs on a dedicated branch (e.g. `autoresearch/mar5` or `autoresearch/mar5-gpu0`).

LOOP FOREVER:

1. Look at the git state: the current branch/commit we're on
2. Tune `train.py` with an experimental idea by directly hacking the code.
3. git commit
4. Run the experiment: `uv run train.py > run.log 2>&1` (redirect everything — do NOT use tee or let output flood your context)
5. Read out the results: `grep "^val_bpb:\|^peak_vram_mb:" run.log`
6. If the grep output is empty, the run crashed. Run `tail -n 50 run.log` to read the Python stack trace and attempt a fix. If you can't get things to work after more than a few attempts, give up.
7. Record the results in the tsv (NOTE: do not commit the results.tsv file, leave it untracked by git)
8. If val_bpb improved (lower), you "advance" the branch, keeping the git commit
9. If val_bpb is equal or worse, you git reset back to where you started

The idea is that you are a completely autonomous researcher trying things out. If they work, keep. If they don't, discard. And you're advancing the branch so that you can iterate. If you feel like you're getting stuck in some way, you can rewind but you should probably do this very very sparingly (if ever).

...

Image Credit: https://github.com/karpathy/autoresearch

Auto-Research

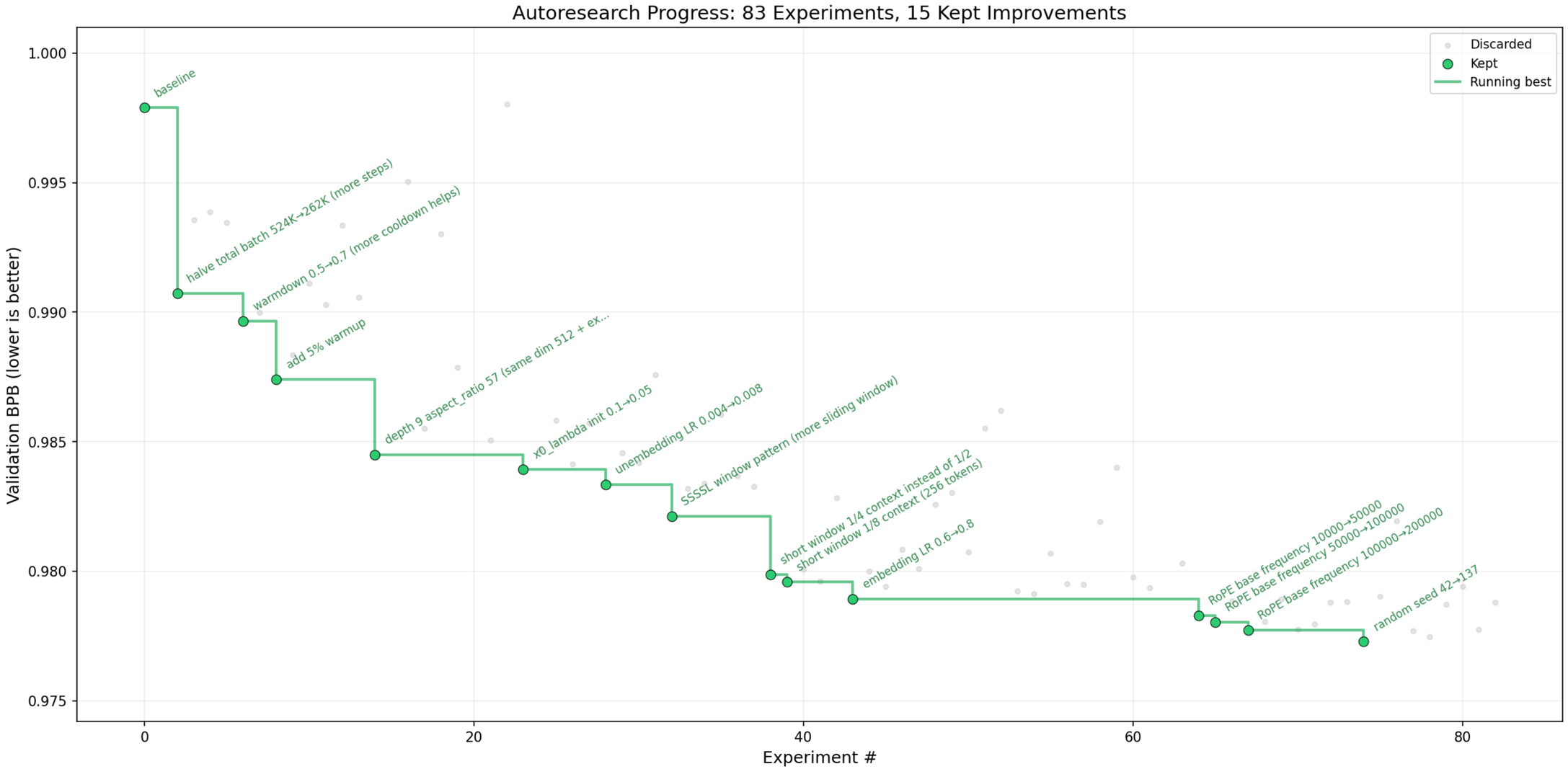
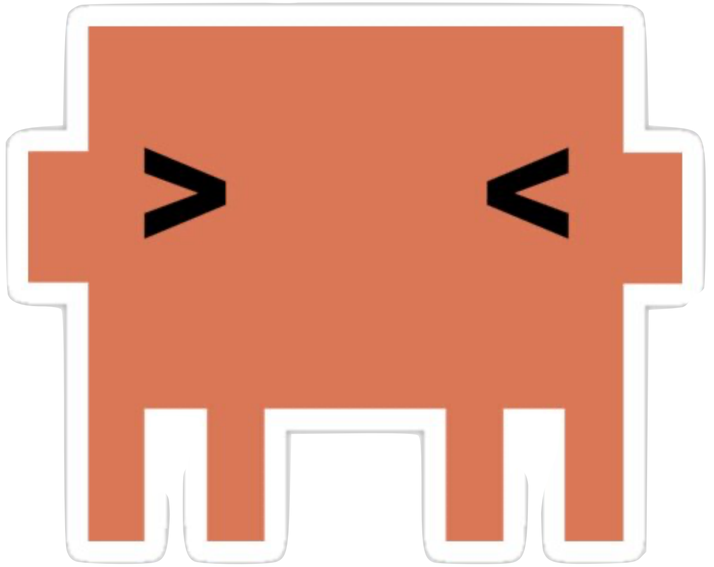


Image Credit: https://github.com/karpathy/autoresearch

Optimization with Agents Summary

- Agents are relatively capable optimizers.
- Set them up in an environment where they can log results, ideate, and execute, and they are pretty good at optimizing anything that runs on software.
- The metric/evals becomes the most critical part to control.



Which Strategy to Use

BetterTogether [Soylu et al., 2024]

Algorithm 1 `BetterTogether`: Optimizing LM programs by alternating prompt and weight optimization steps, instantiated in Algorithm 2

Input: **Program** $\Phi_{\langle\Theta,\Pi\rangle} = \Phi_\Theta \odot \Phi_\Pi$,
with module weights $\Theta = [\theta_1, \ldots, \theta_{|\Phi|}]$
and module prompts $\Pi = [\pi_1, \ldots, \pi_{|\Phi|}]$
Training Set X and **Metric** μ

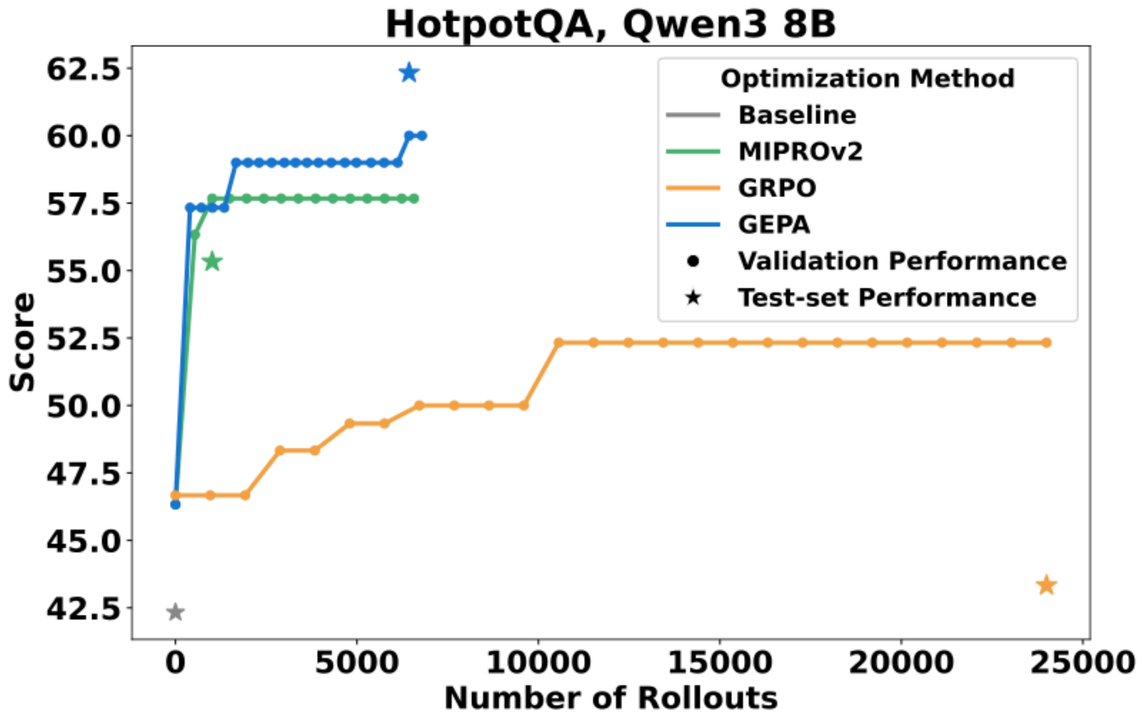
function BETTERTOGETHER($\Phi_{\langle\Theta,\Pi\rangle}, X, \mu$)
  $\Pi' \leftarrow$ OPTIMIZEPROMPTS($\Phi_{\langle\Theta,\Pi\rangle}, X, \mu$)
  $\Theta' \leftarrow$ FINETUNEWEIGHTS($\Phi_{\langle\Theta,\Pi'\rangle}, X, \mu$)
  $\Pi'' \leftarrow$ OPTIMIZEPROMPTS($\Phi_{\langle\Theta',\Pi\rangle}, X, \mu$)
  return $\Phi_{\langle\Theta',\Pi''\rangle}$
end function

Strategy	mistral-7b-instruct-v0.2			llama-2-7b-chat			llama-3-8b-instruct		
	HotPotQA	GSM8K	Iris	HotPotQA	GSM8K	Iris	HotPotQA	GSM8K	Iris
Baseline Strategies									
Vanilla Zero-shot	17.2	40.3	26.0	13.2	24.0	0.0	31.6	72.7	48.0
Prompt Optimization (Π)	33.8	46.4	57.3	33.3	26.0	56.7	**46.9**	77.9	79.3
Weight Optimization (Θ)	22.9	40.7	29.3	12.2	24.0	–	34.8	75.1	37.3
$\Pi \rightarrow \Pi$	33.8	47.7	59.3	32.6	24.7	64.0	46.5	77.6	**82.0**
$\Theta \rightarrow \Theta$	24.0	42.8	38.0	13.0	24.1	–	34.4	44.1	39.3
BetterTogether Strategies									
$\Pi \rightarrow \Theta$	36.3	47.3	30.7	32.7	**27.3**	26.7	42.8	77.6	44.0
$\Theta \rightarrow \Pi$	33.0	**48.3**	**66.7**	34.2	26.6	–	43.6	**78.9**	78.7
$\Pi \rightarrow \Theta \rightarrow \Pi$	**37.6**	46.8	52.7	**34.8**	26.3	**65.3**	46.7	77.0	79.3

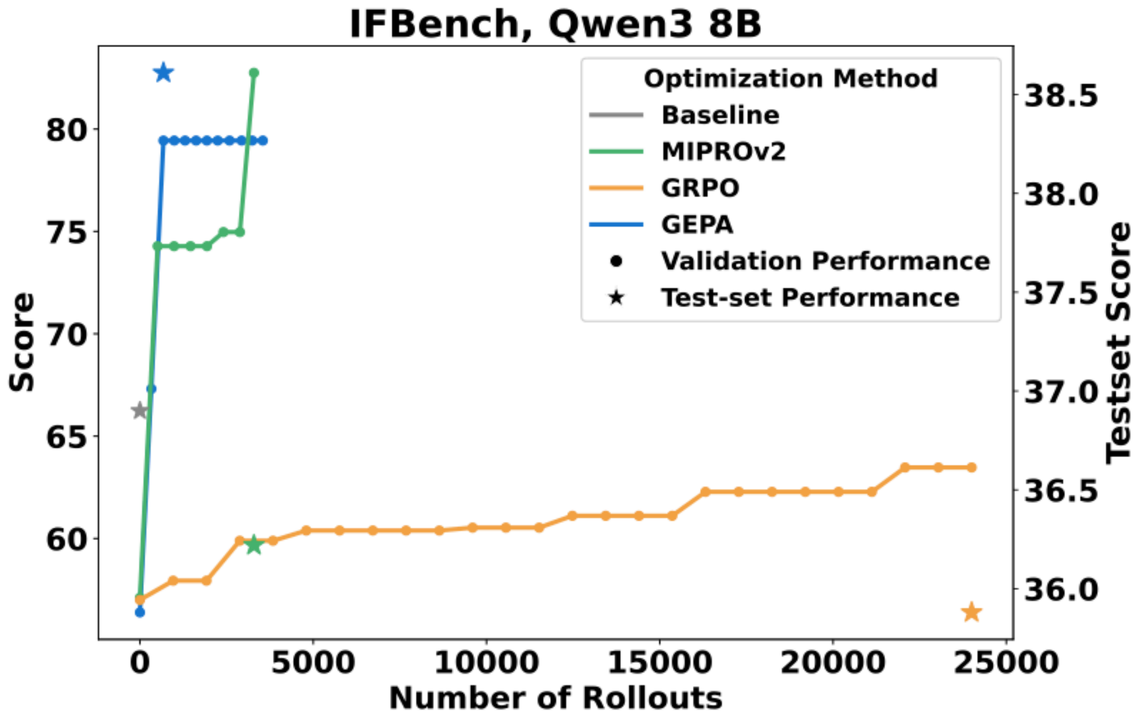
Sometimes it is best to optimize the prompts, then the weights, then the prompts again.

Image Credit: https://aclanthology.org/2024.emnlp-main.597.pdf

GEPA [Agrawal et al., 2026]



(a) HotpotQA, Qwen3 8B

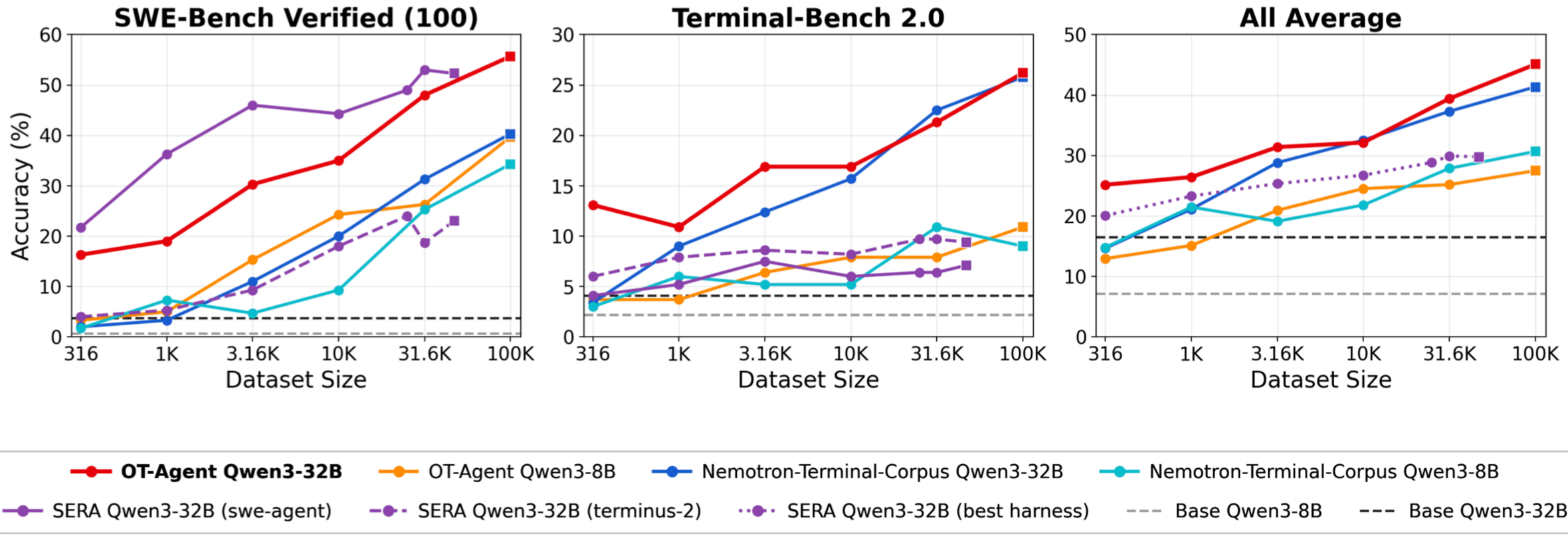


(b) IFBench, Qwen3 8B

Prompt optimization is much more sample-efficient than RL.

Image Credit: https://arxiv.org/pdf/2507.19457

OpenThoughts-Agent [Raoof et al., 2026]

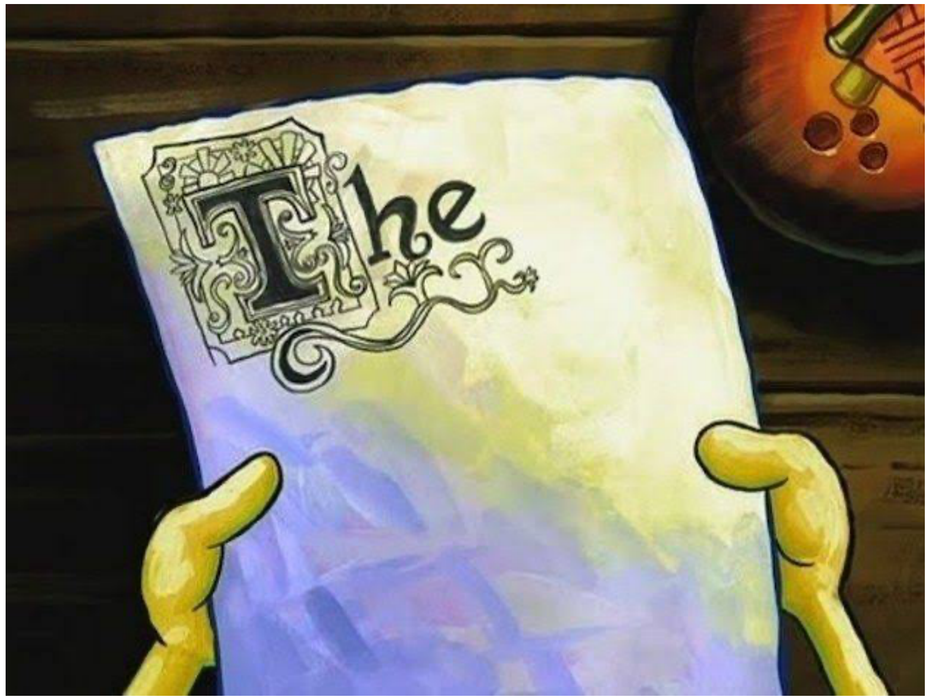


Weight optimization is more data hungry, but can effectively grow with thousands and thousands of examples.

Image Credit: https://arxiv.org/pdf/2606.24855

Reasons to use context optimization

- **Optimization for API Access models.** Some strong models do not provide fine-tuning access.
- **Need to rerun optimization frequently and quickly.** Context optimization runs faster and on simpler infrastructure than weight.
- **Environment interactions are expensive.** Context optimization is more sample efficient.



Reasons to use weight optimization

- **Context Optimization fails.** Especially for small models sometimes instruction following skills are minimal.
- **You have a capable teacher and want a cheaper model**. Distilling directly into a student model is often effective.
- **The context is getting too long**. Context optimization adds more complexity to the prompt. Sometimes it is useful to use weight optimization to avoid context-bloat from prompt optimization.



Summary

SAS vs MAS	Orchestration	Communication	Coordination	Failure Modes

Summary

- [x] Conditions for Optimization
- [x] Context Optimization
- [x] Weight Optimization
- [x] Optimization with Agents
- [x] Which Strategy to Use



Guest Lecture next **Monday**!