Towards Solving Hard Problems via Test-Time Training

Jonas Hübotter

July 10, 2025

Part 1: Test-time scaling

Agents

Setting:

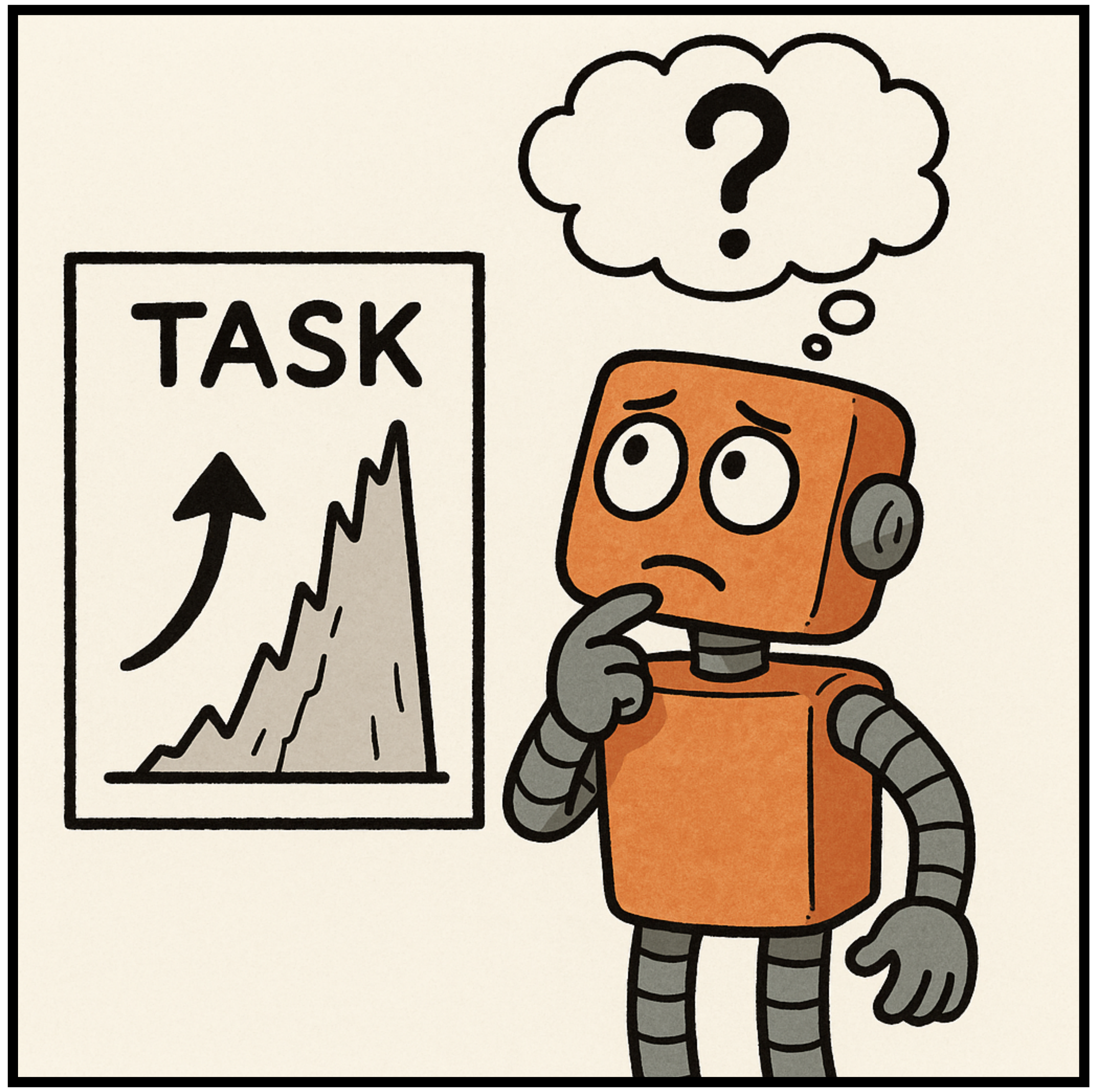
- We have an agent, pre-trained on many tasks (“train-time”)
- At “test-time” the agent is given a specific task to solve

Previously: compute is scaled at train-time by scaling size of agent & number of training tasks

Now: how can we effectively scale compute at test-time?

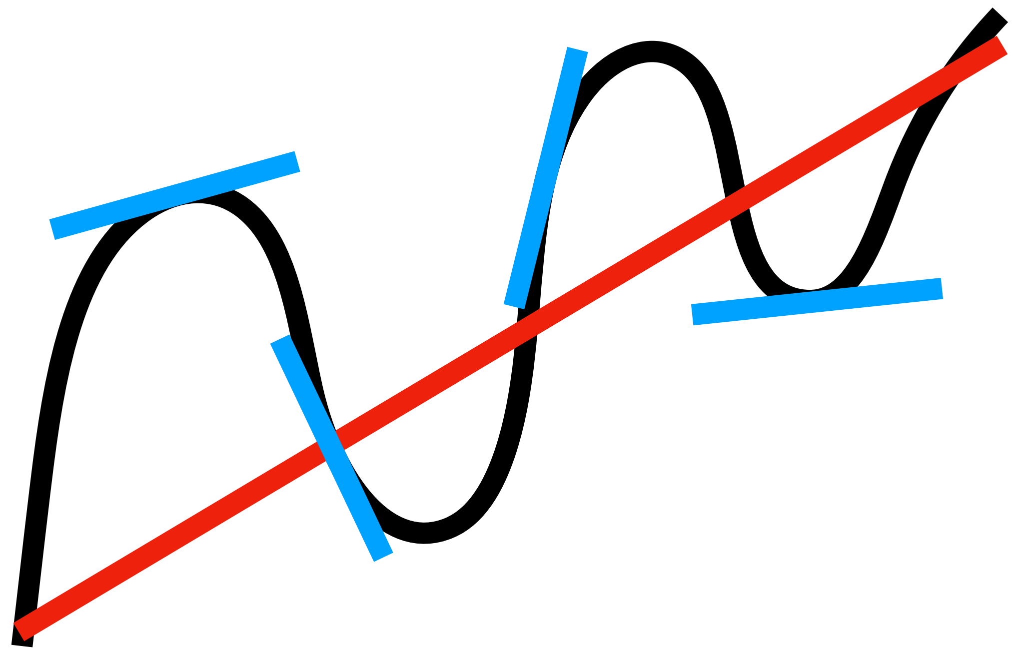
Test-time scaling

- **Test-time inference / search**
  - e.g., Best of N, Majority Voting, Beam Search
  - some, but often not significant improvements
- **Train-time reinforcement learning** (“reasoning”)
  - e.g., DeepSeek-R1, etc.
  - trains the model via RL to produce longer chains of thought
- **Test-time training (TTT)**
  - model is updated (“learns”) at test-time
  - towards specialization & deep exploration



Why does test-time training work?

- **Until recently:** focus on foundation models that generalize “zero-shot” to many tasks
  - Many tasks (perception, simple factual knowledge, etc.) can be solved 🎉
  - Problem: model’s need to be scaled to obtain the ability to solve additional tasks.
- **TTT: Specialization after Generalization**
  - Allows models to specialize to an individual task

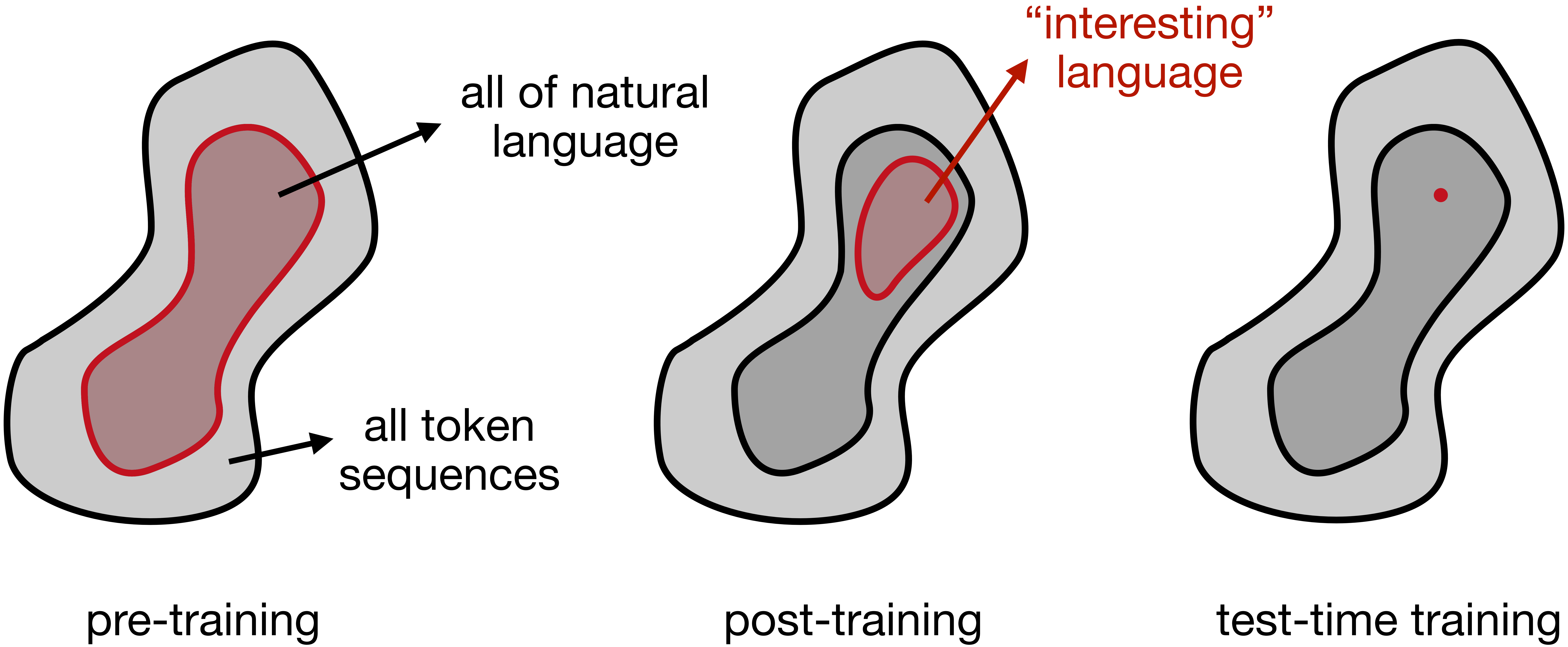


Transduction

(Vapnik; ’80s)

“When solving a problem of interest, do not solve a more general problem as an intermediate step. Try to get the answer that you really need but not a more general one.”

Test-time training vs "standard" post-training



Test-time training

Different approaches to TTT:

- Imitation learning
- Reinforcement Learning
  - Offline — based on existing experience
  - Online — interaction with an environment

Which data should the agent collect?

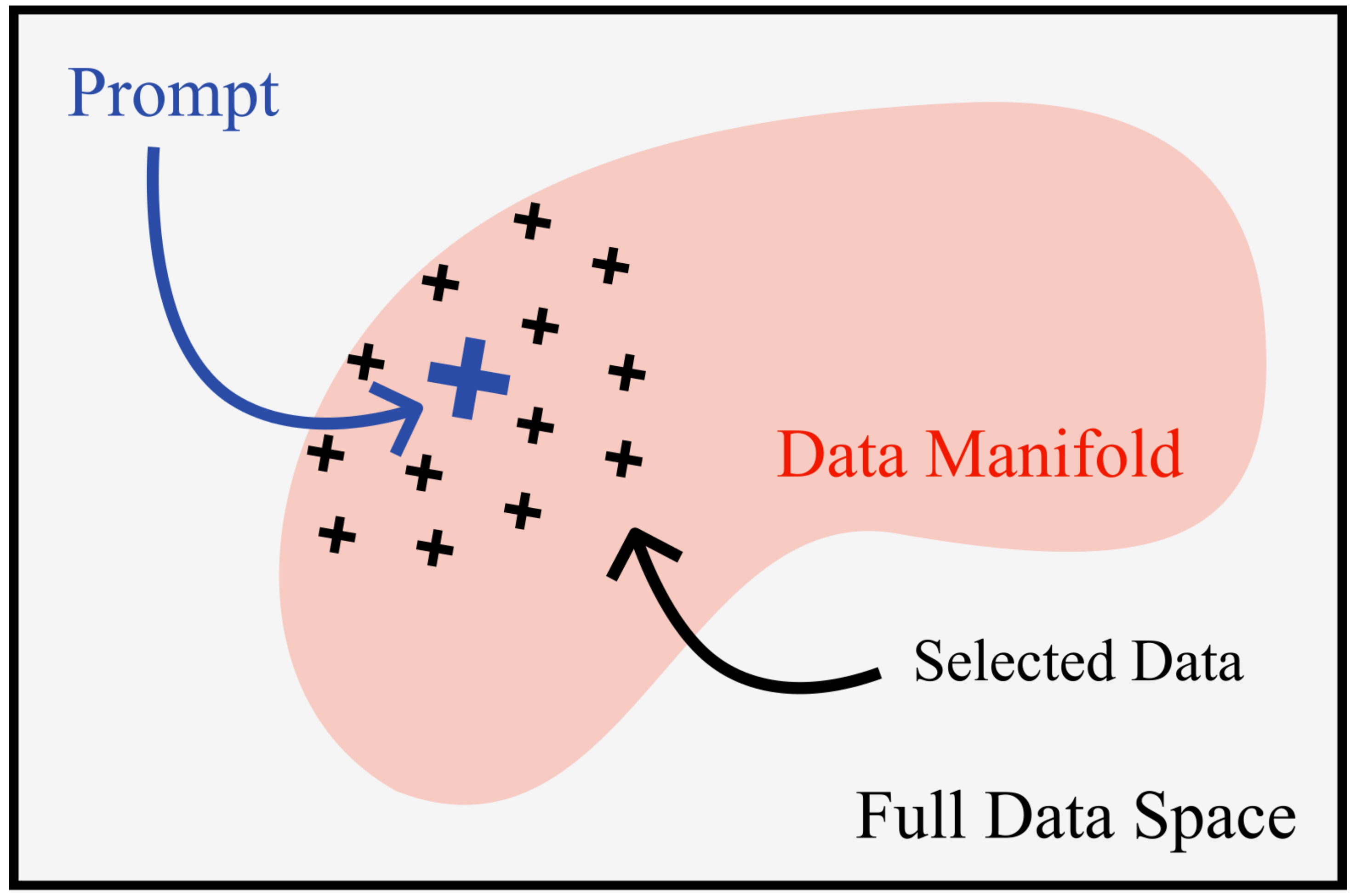
Different approaches to TTT:

• Imitation learning	Imitating (existing / new) data
• Reinforcement Learning	
• Offline — based on existing experience	Learning from existing experience
• Online — interaction with an environment	Collecting new experience

TTT Agents

Part 2: Test-time training agents

1. Imitating existing data



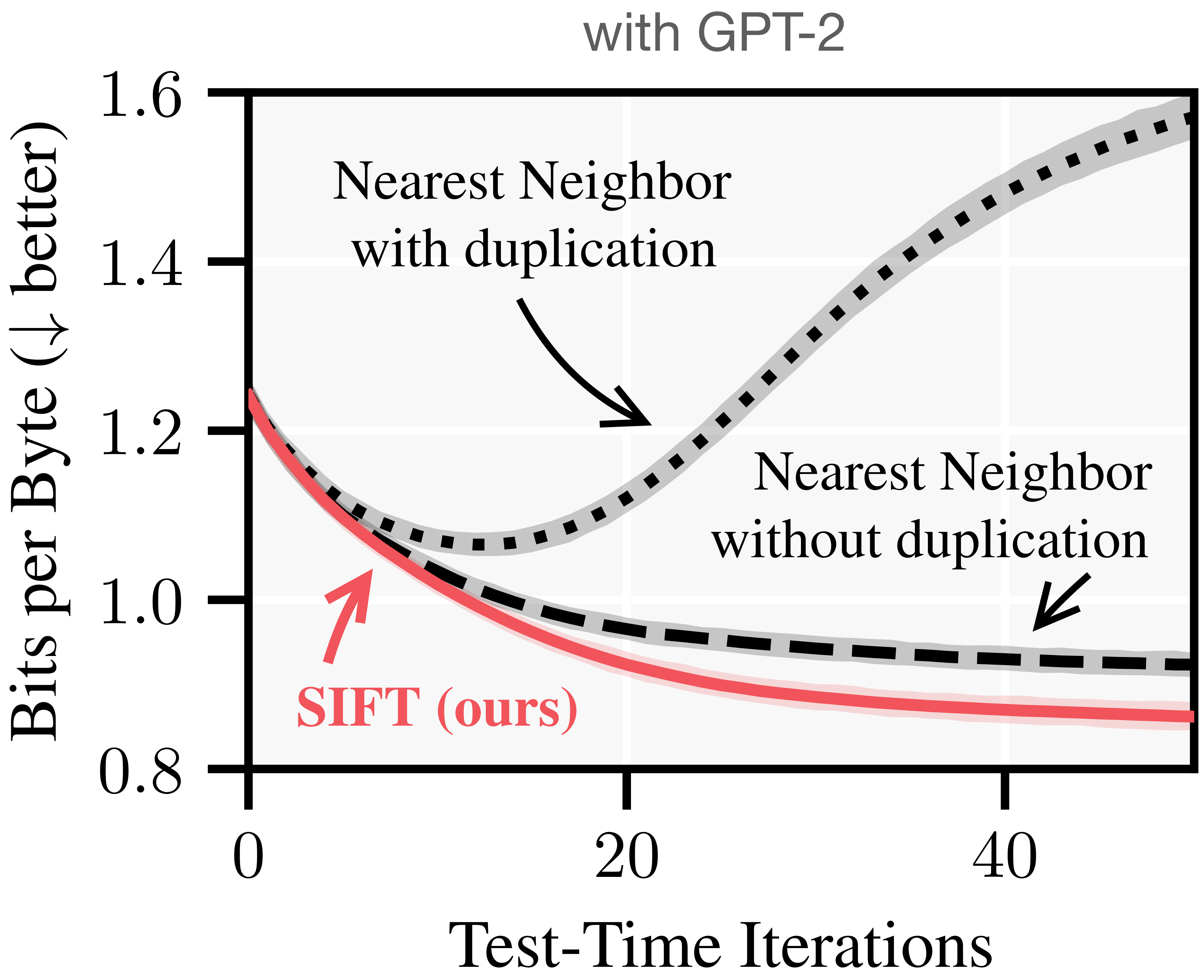
Selecting informative data for fine-tuning (SIFT):
Select data that maximally reduces "uncertainty" about how to solve the task.

1. given task x, find local data D_x (from memory)
2. fine-tune pre-trained model f on local data D_x to get **specialized model** f_x
3. predict $f_x(x)$

(**H**, Sukhija, Treven, As, Krause; NeurIPS '24; **H**, Bongni, Hakimi, Krause; ICLR '25)

Evaluation: language modeling on the Pile

Pile dataset
NIH Grants
US Patents
GitHub
Enron Emails
Common Crawl
ArXiv
Wikipedia
PubMed Abstr.
Hacker News
Stack Exchange
PubMed Central
DeepMind Math
FreeLaw
All

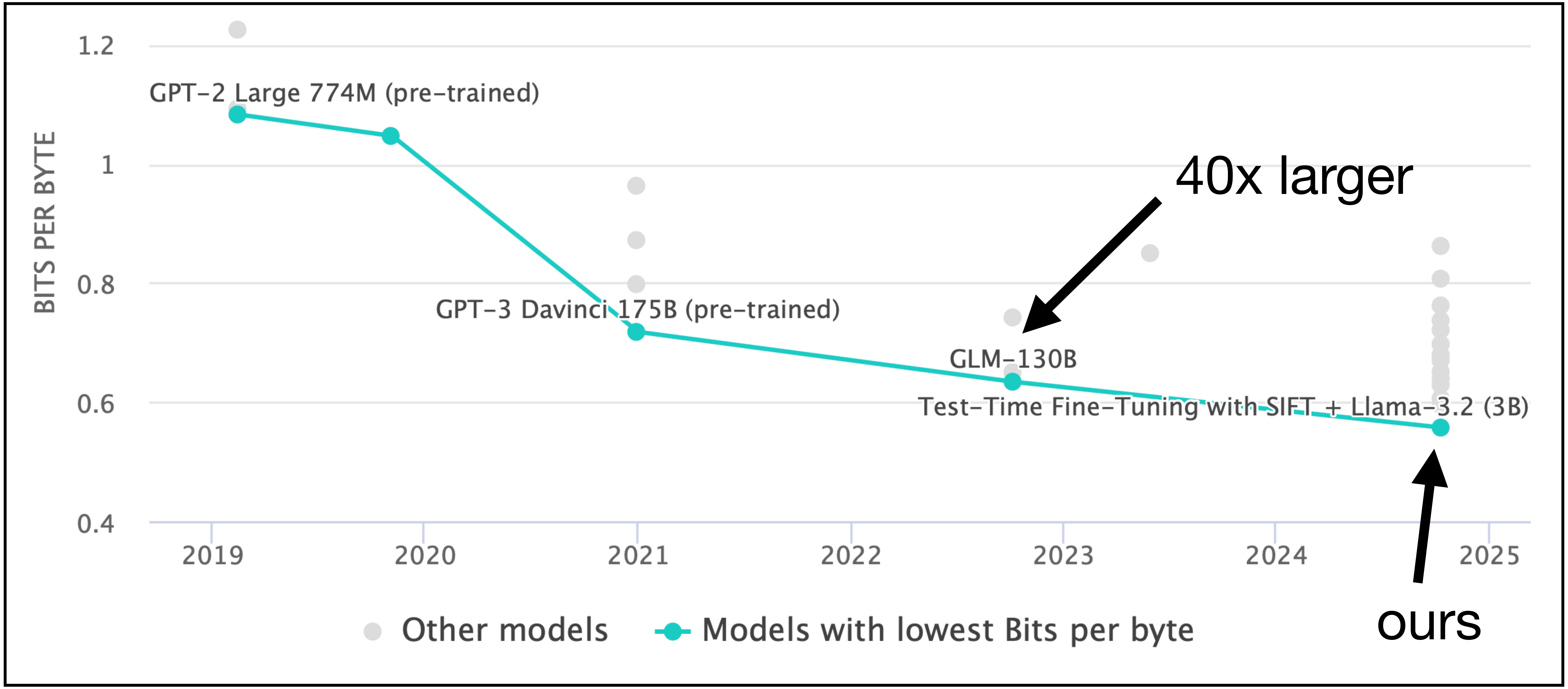


Observations

- larger relative gains with stronger base models
- larger relative gains with larger "memory"

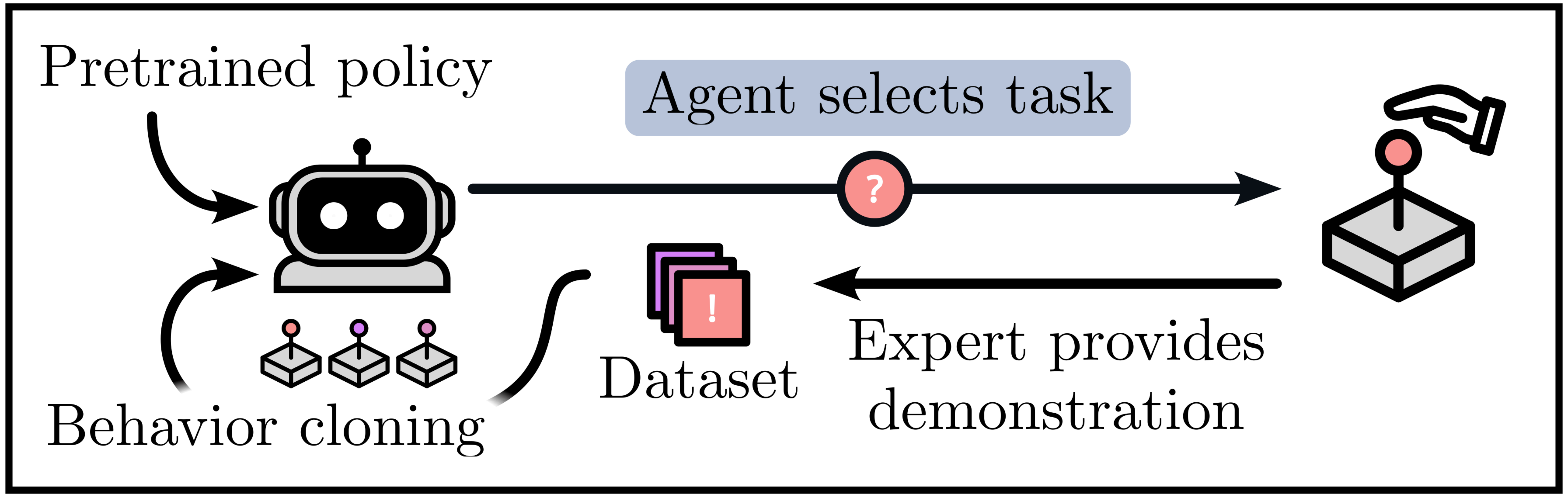
	US	NN	NN-F	SIFT	Δ
NIH Grants	93.1 (1.1)	84.9 (2.1)	91.6 (16.7)	**53.8** (8.9)	↓31.1
US Patents	85.6 (1.5)	80.3 (1.9)	108.8 (6.6)	**62.9** (3.5)	↓17.4
GitHub	45.6 (2.2)	42.1 (2.0)	53.2 (4.0)	**30.0** (2.2)	↓12.1
Enron Emails	**68.6** (9.8)	**64.4** (10.1)	91.6 (20.6)	**53.1** (11.4)	↓11.3
Wikipedia	67.5 (1.9)	**66.3** (2.0)	121.2 (3.5)	**62.7** (2.1)	↓3.6
Common Crawl	92.6 (0.4)	90.4 (0.5)	148.8 (1.5)	**87.5** (0.7)	↓2.9
PubMed Abstr.	88.9 (0.3)	87.2 (0.4)	162.6 (1.3)	**84.4** (0.6)	↓2.8
ArXiv	85.4 (1.2)	**85.0** (1.6)	166.8 (6.4)	**82.5** (1.4)	↓2.5
PubMed Central	**81.7** (2.6)	**81.7** (2.6)	155.6 (5.1)	**79.5** (2.6)	↓2.2
Stack Exchange	78.6 (0.7)	78.2 (0.7)	141.9 (1.5)	**76.7** (0.7)	↓1.5
Hacker News	**80.4** (2.5)	**79.2** (2.8)	133.1 (6.3)	**78.4** (2.8)	↓0.8
FreeLaw	**63.9** (4.1)	**64.1** (4.0)	122.4 (7.1)	**64.0** (4.1)	↑0.1
DeepMind Math	**69.4** (2.1)	**69.6** (2.1)	121.8 (3.1)	**69.7** (2.1)	↑0.3
All	80.2 (0.5)	78.3 (0.5)	133.3 (1.2)	**73.5** (0.6)	↓4.8

New SOTA on the Pile benchmark

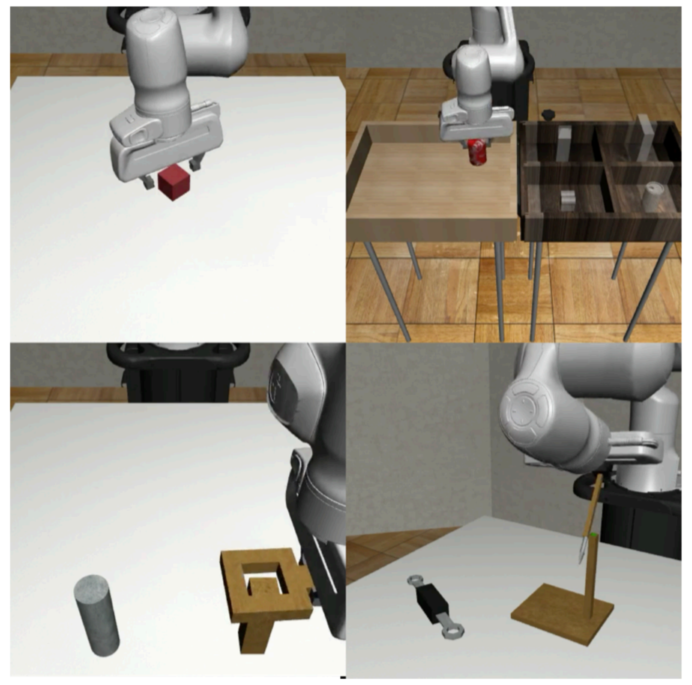


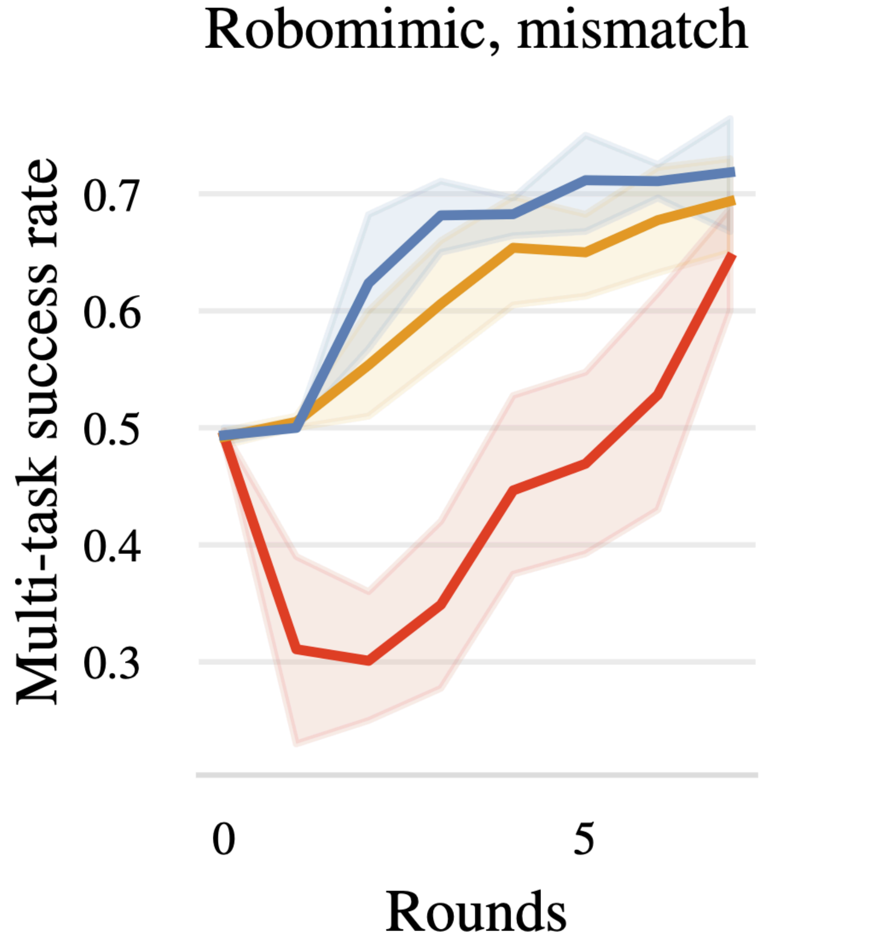
https://paperswithcode.com/sota/language-modelling-on-the-pile

2. Imitating data to be collected



Query data that maximally reduces the agent’s uncertainty about the test-time task <u>in expectation</u>.

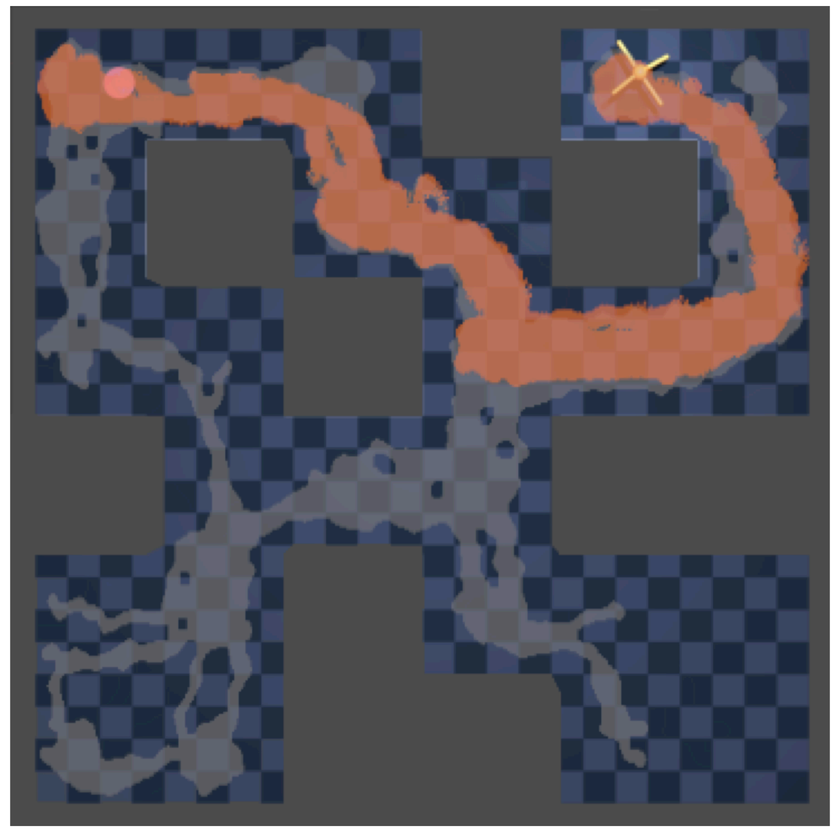




(Bagatella, **H**, Martius, Krause; ICML ’25)

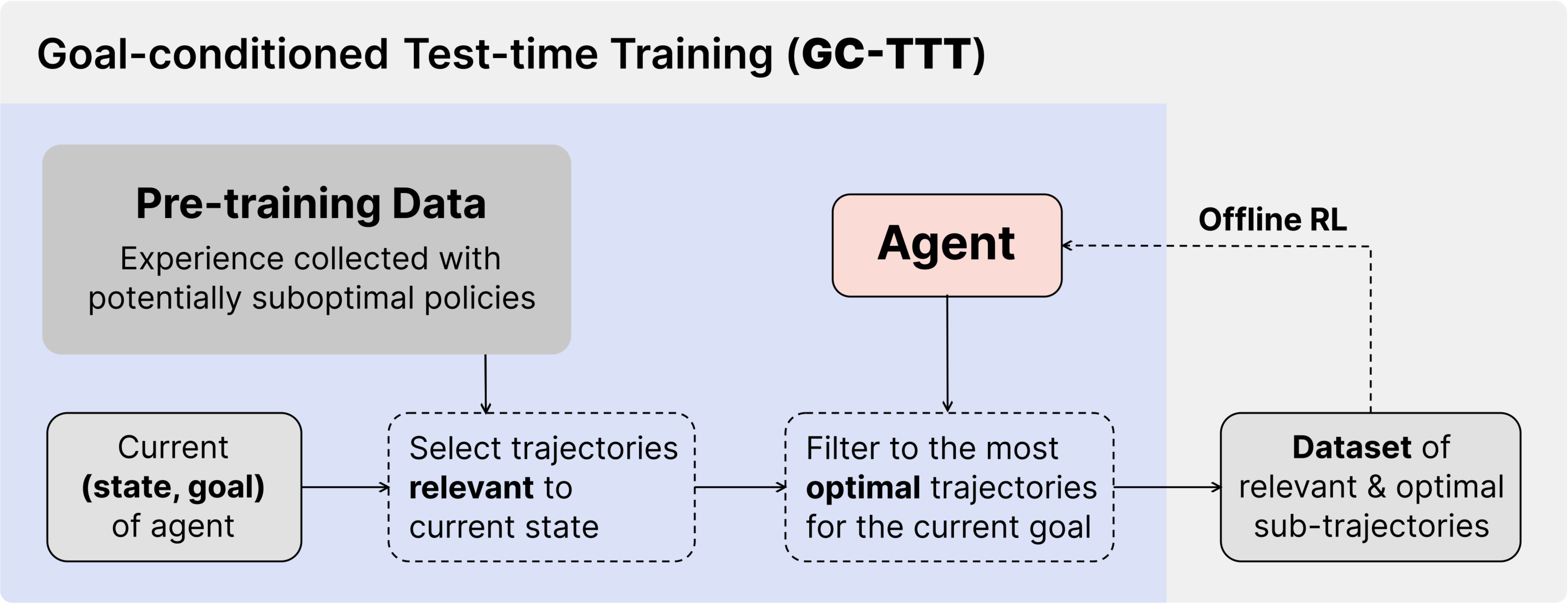
3. Learning from existing experience

- Experience is not from the optimal policy — should not be imitated directly.
- We need to infer the optimal policy from previous (suboptimal) experience.
- Instead of imitation learning (i.e., behavioral cloning / supervised learning) we do *offline* RL.



(Bagatella*, Albaba*, **H**, Martius, Krause; ICML PUT workshop ’25)

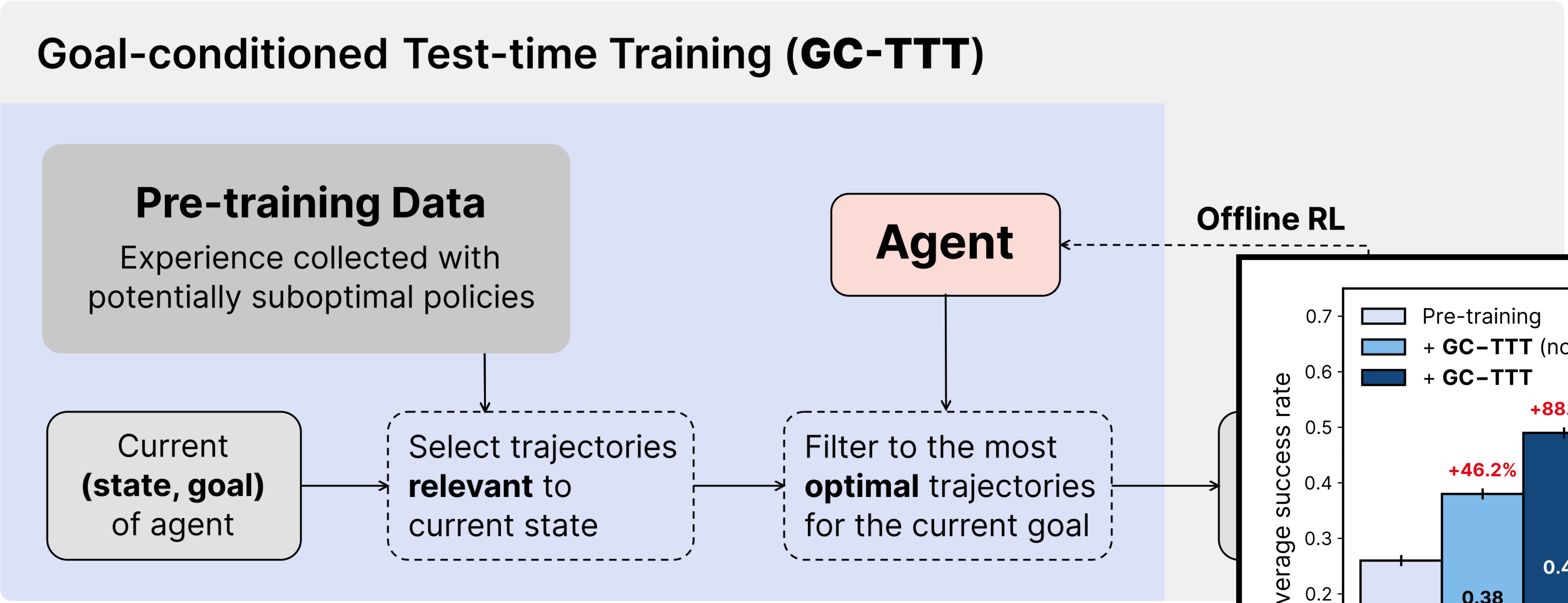
3. Learning from existing experience

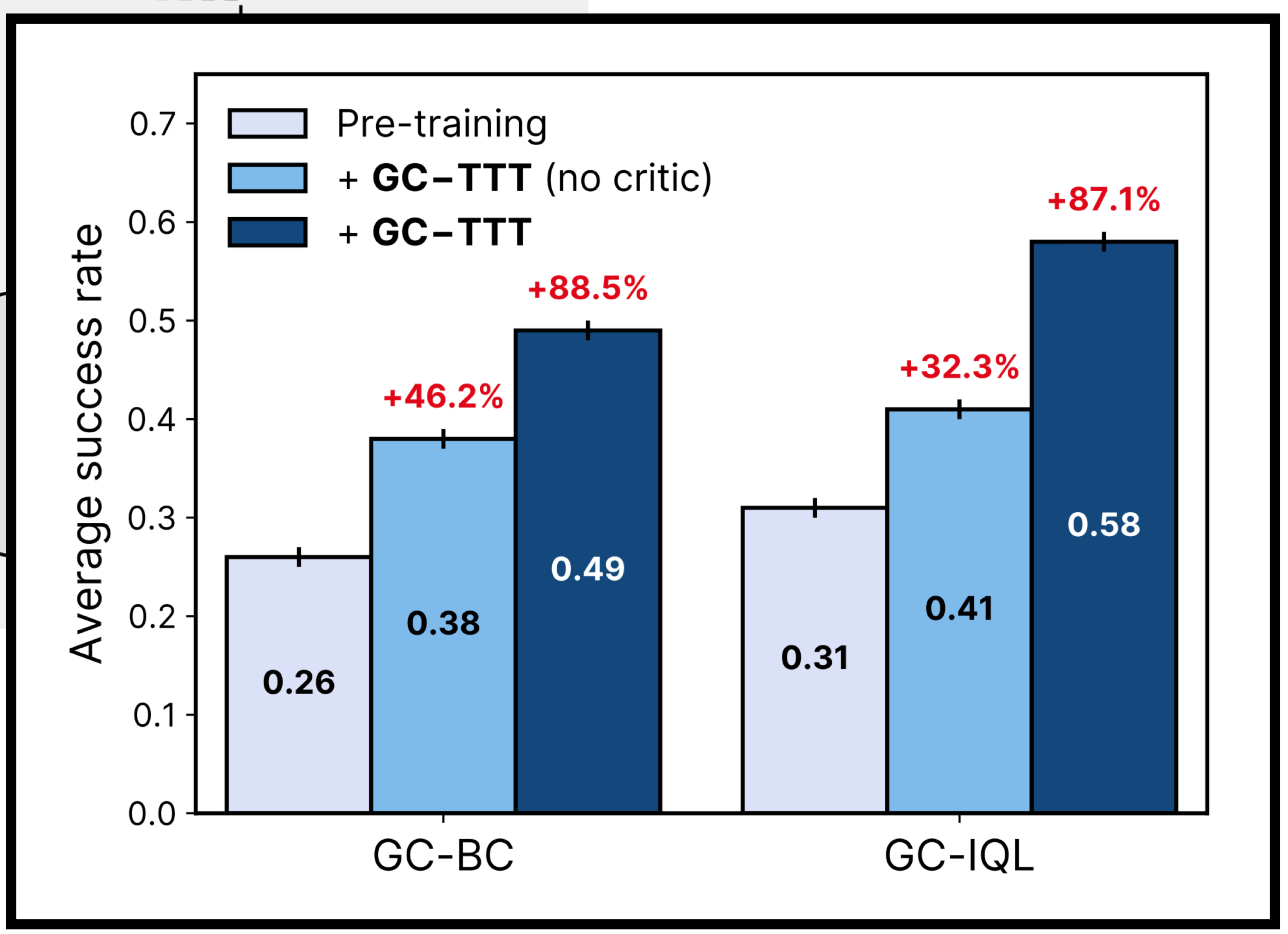


- Specializes the agent to the current goal & current state
- We do this recursively every K steps

(Bagatella*, Albaba*, **H**, Martius, Krause; ICML PUT workshop '25)

3. Learning from existing experience



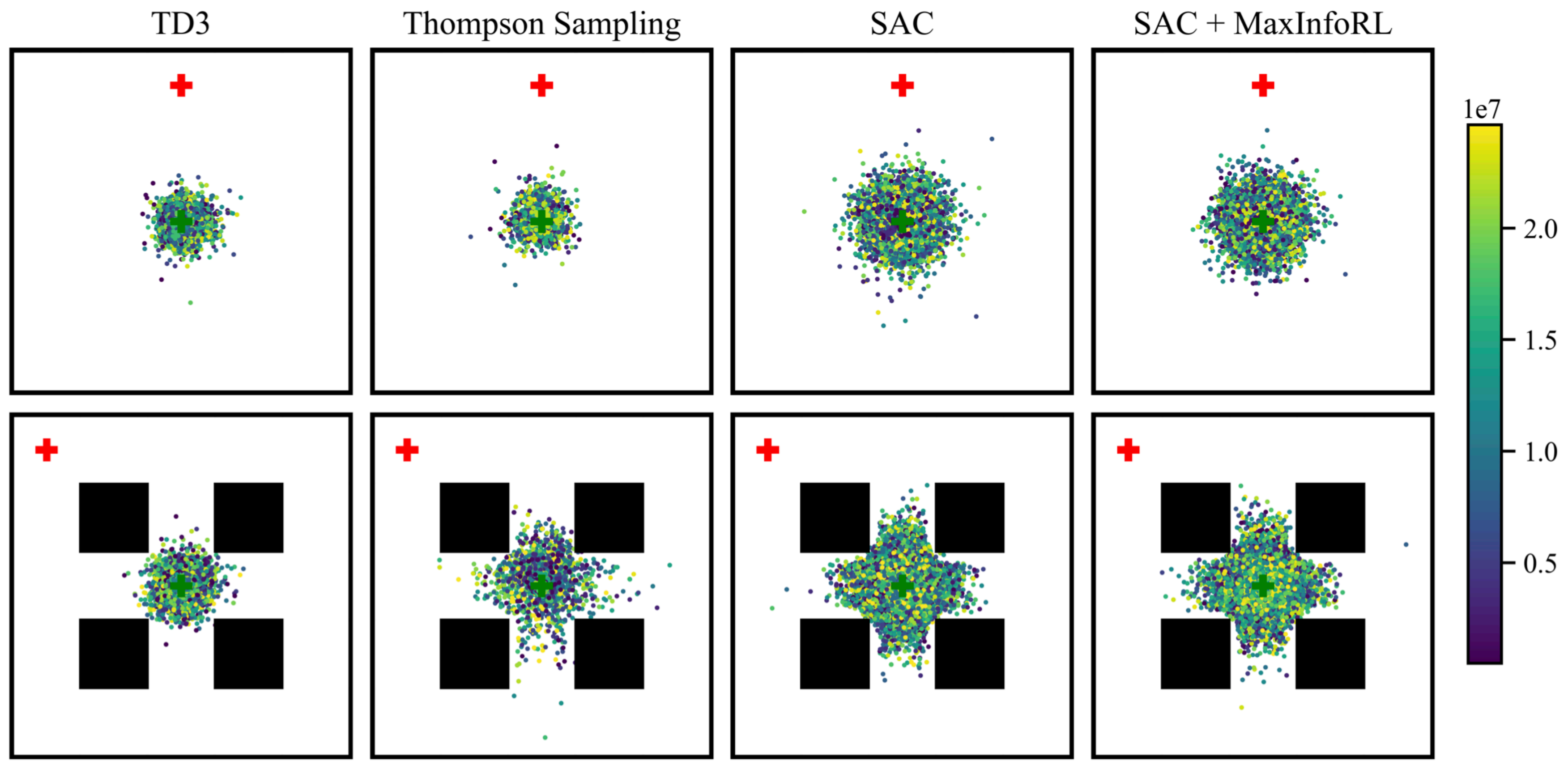


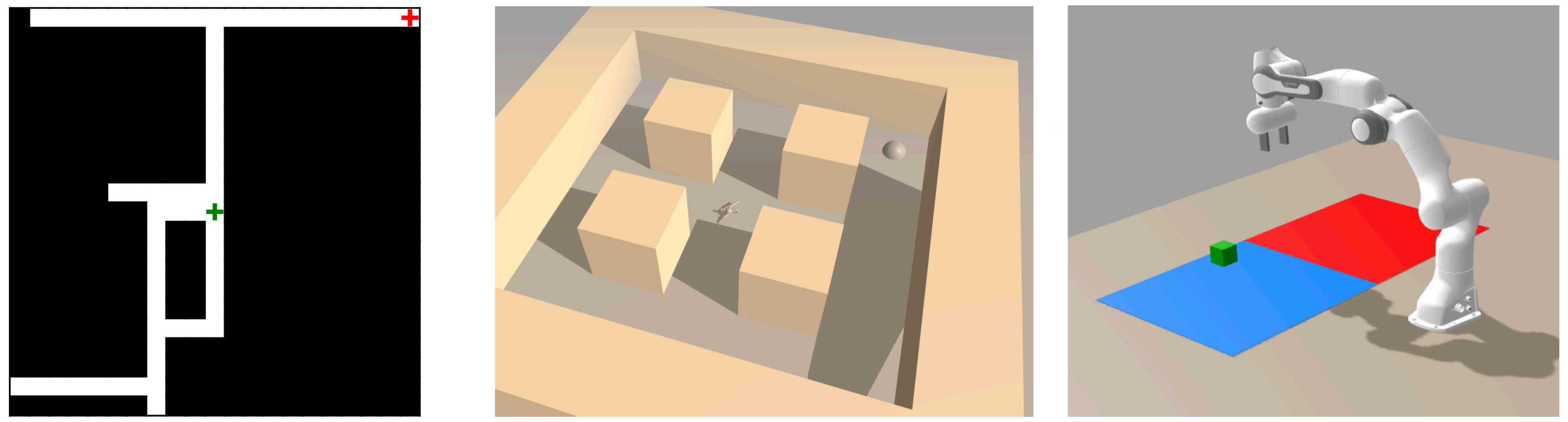
- Specializes the agent to the current goal & current state
- We do this recursively every K steps

(Bagatella*, Albaba*, **H**, Martius, Krause; ICML PUT workshop ’25)

4. Collecting new experience

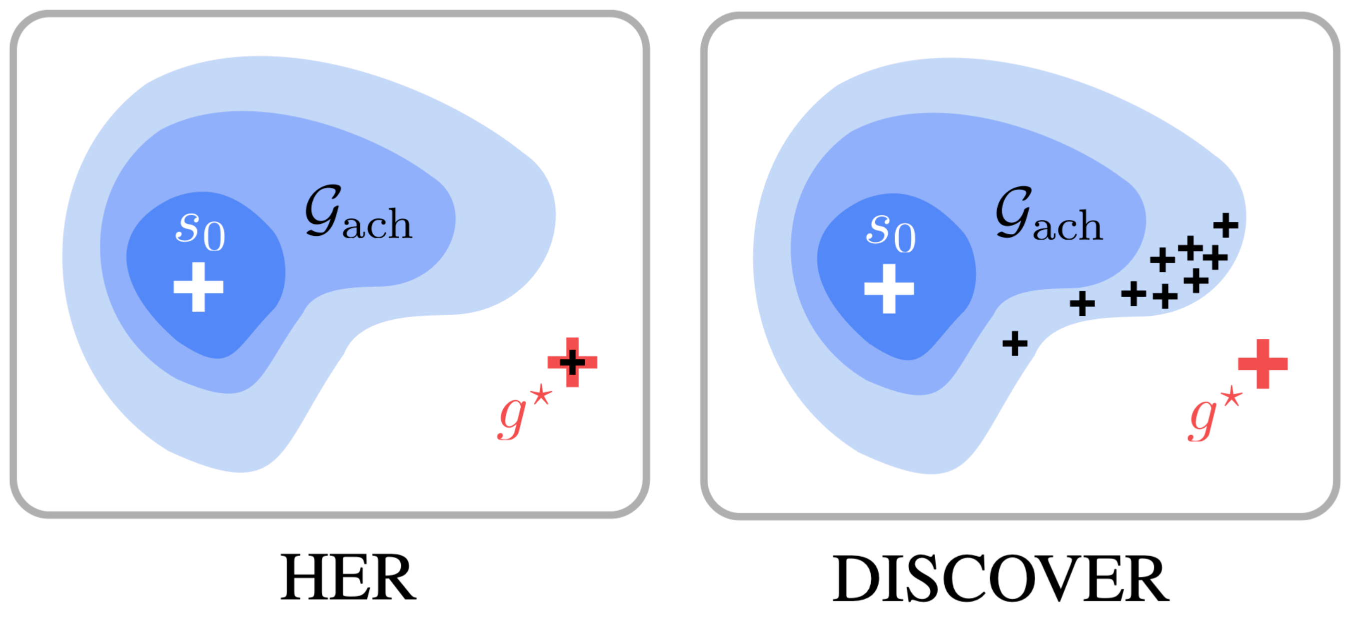
- Solving previously unsolved tasks requires the agent to obtain experience.
  - i.e., *online* interaction with an environment — called **TTRL**
- How should the agent select which experience to collect?
- Commonly in online RL, the agent simply attempts the test-time task.
- But if this task is not currently achievable, the agent never observes a reward:



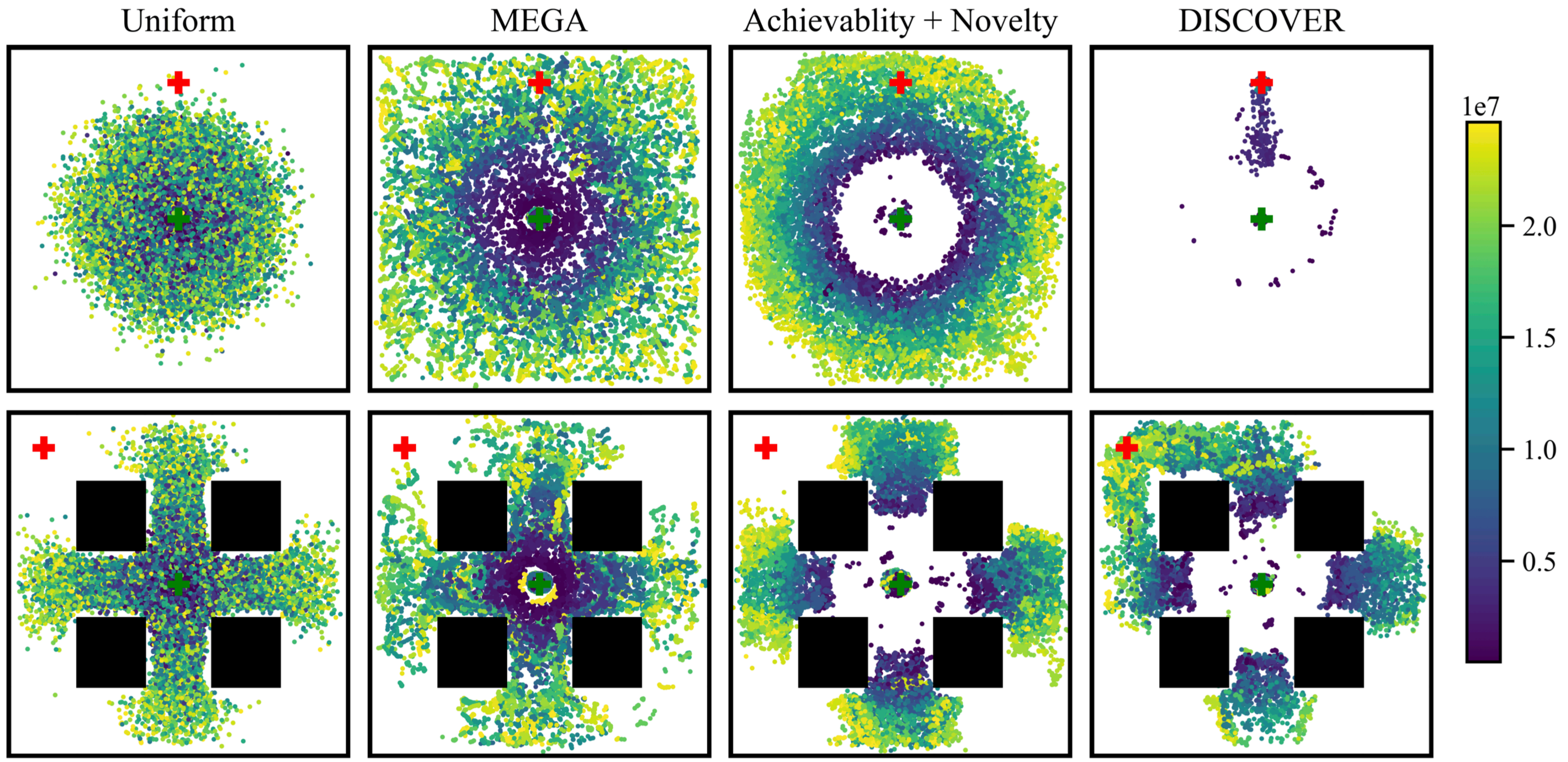


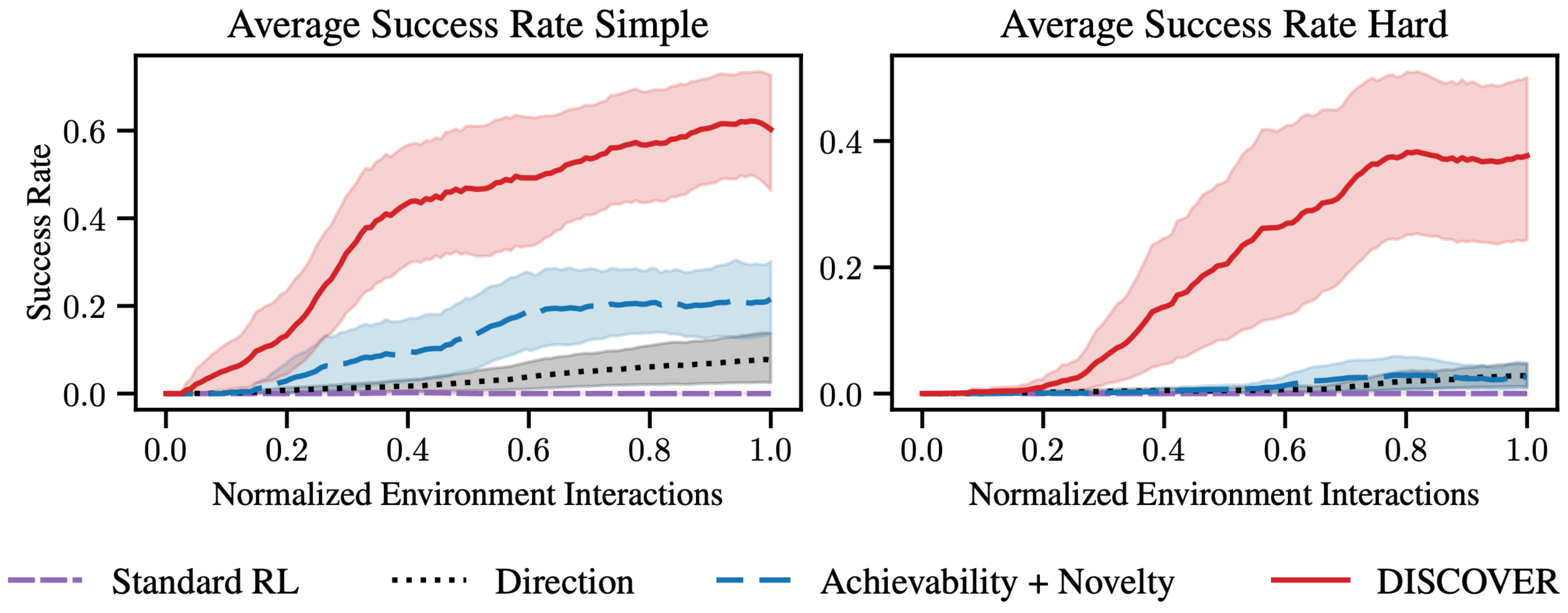
(a) Point maze (b) Ant maze (c) Arm

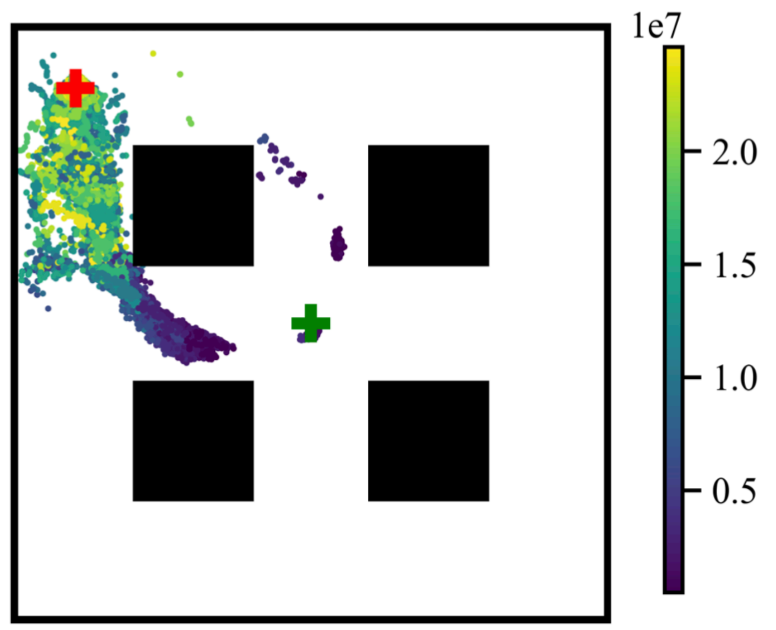
4. Collecting new experience



DISCOVER: Select intermediate tasks that are not "too easy", not "too hard", and relevant to the target task.







with prior

(Diaz-Bone*, Bagatella*, **H***, Krause; ICML EXAIT workshop '25)

Motivation: Going beyond short-term memory

- Train-time RL trains the model to produce longer chains of thought.
- This increases the context ("scratch memory") needed to solve a problem.
- While this is useful, eventually deep exploration attains experience that does not fit (or is inefficient to fit) into scratch memory.
- Why did we pre-train in the first place? Precisely to compress large amounts of experience.

$\Longrightarrow$ To learn completely new skills at test-time, we need to update the model's representations — simply expanding short-term memory is not enough!

Conclusion

- TTT is a method for specializing “foundation” models to individual tasks
- There are many open questions around TTT and TTT agents for hard tasks

Happy to discuss more

jonas.huebotter@inf.ethz.ch