Test-Time Training Agents to Solve Challenging Problems

Jonas Hübotter

November 5, 2025

About me

- Undergrad at TU Munich in CS and Math
- Masters at ETH Zurich in Theoretical CS and ML
- PhD Student at ETH Zurich with Andreas Krause

My interests:

examples later!

- Vertical: Solving “hard” tasks
- Horizontal: Specialization, RL, meta-learning & more

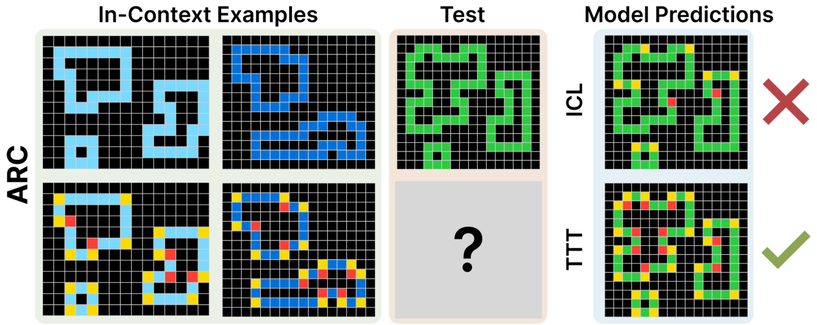
Thanks to my collaborators

I’ll acknowledge everyone along the way

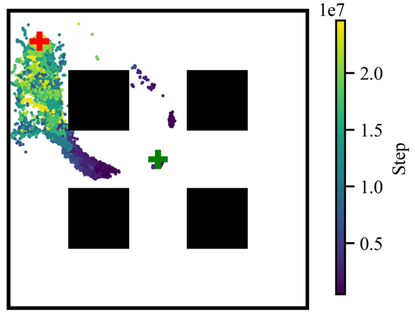
Agenda

1. TTT can enable models to outperform globally trained models *in-distribution*.
2. TTT enables models to continue learning & improving at test-time.

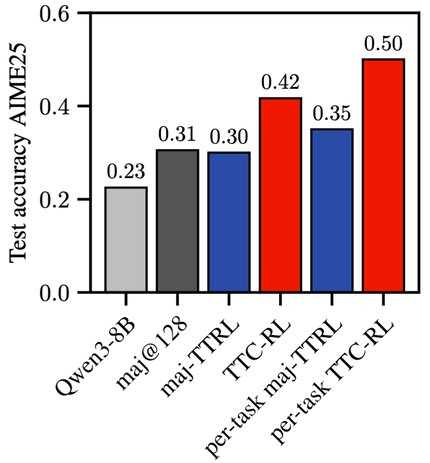
Applications of test-time training



Few-shot learning
(Akyürek et al.; ICML ’25)



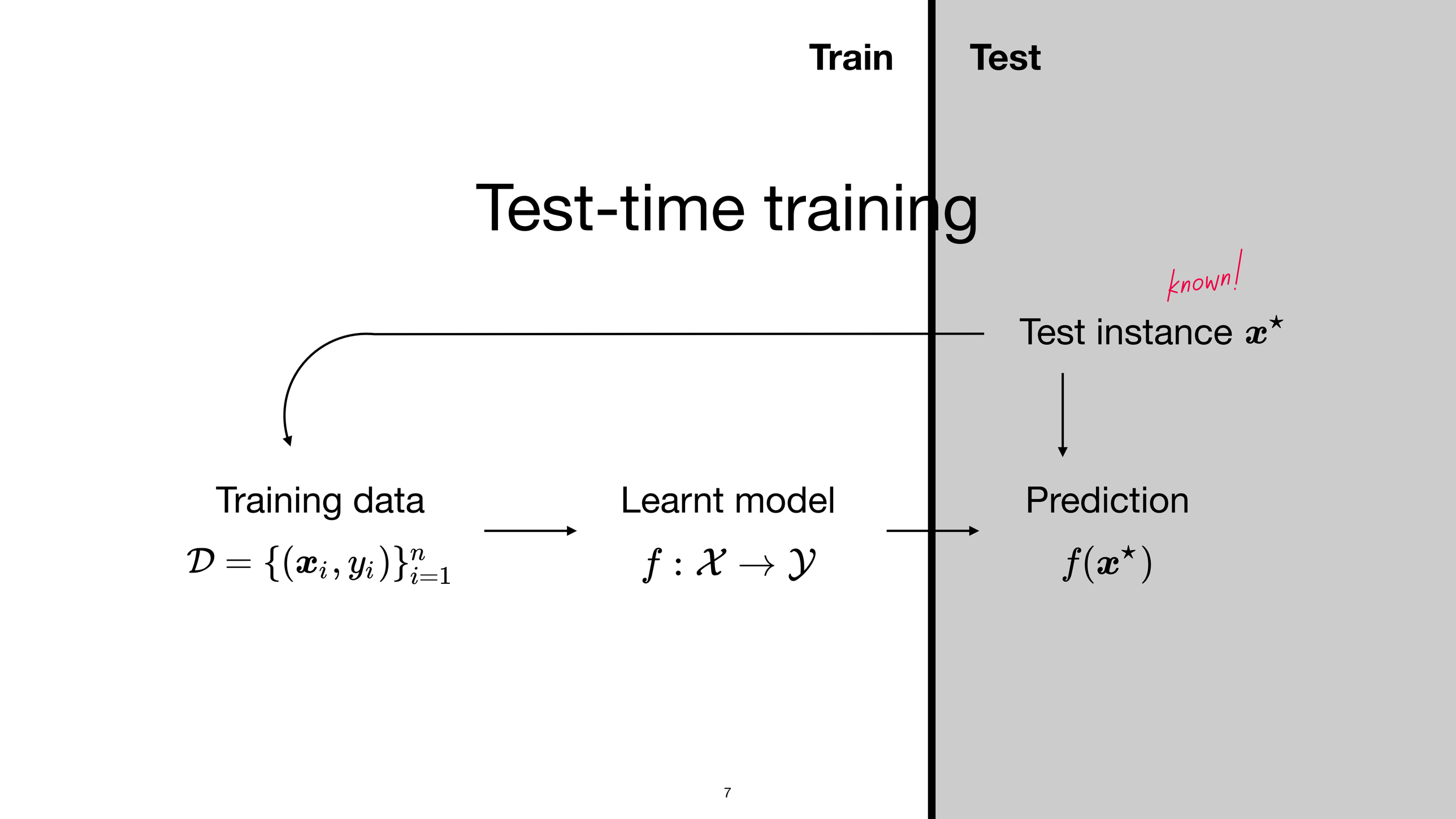
High-dimensional robotics tasks
(Diaz-Bone*, Bagatella*, **H***, Krause; NeurIPS ’25)



Reasoning
(**H***, Diaz-Bone*, Hakimi, Krause, Hardt; preprint)

Part 1: Test-time training

“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.” —*Vladimir Vapnik*


Train
Test
Test-time training
known!
Test instance $\boldsymbol{x}^\star$
Training data
$\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$
Learnt model
$f : \mathcal{X} \to \mathcal{Y}$
Prediction
$f(\boldsymbol{x}^\star)$

A working definition of test-time training

- Consider a dataset $D = \{(x_i, y_i)\}_i$
- Global training trains a single model f and predicts $f(x)$
- TTT trains a *specific* model f_x and predicts $f_x(x)$

Can TTT improve predictions even *in-distribution*
while using *only* D for specializing the model?

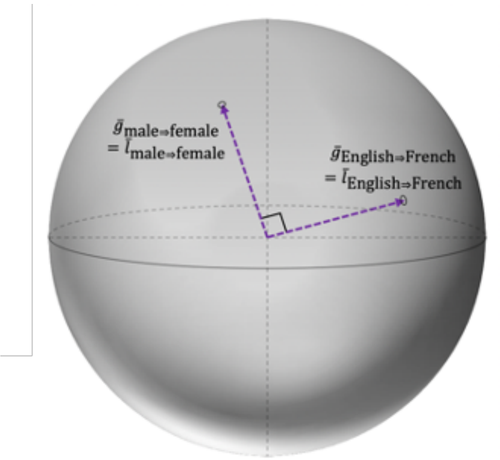
training on neighborhoods within the dataset

Why can TTT work?

CCFM @ NeurIPS ’25

- Assume existence of an s-sparse **concept space** $\Phi : \mathcal{X} \to \mathbb{R}^{d_1}$.
- Assume that the target is linear in the concept space: $f^\star(x) = \langle \Phi(x), w^\star \rangle$
  - This is called the **linear representation hypothesis**

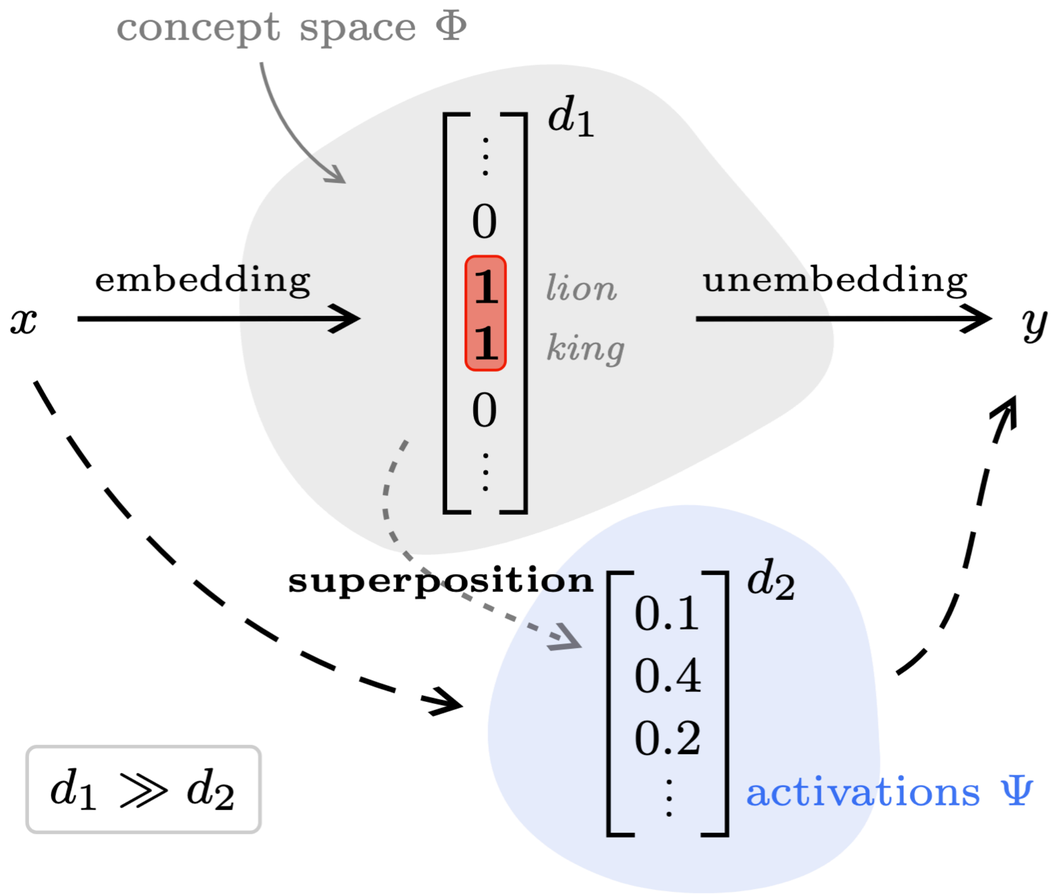
    Note: In classification, the logits are canonically parameterized as $\langle \Phi(x), w_c \rangle$ where w_c is the class-specific weight vector. The class probability is then given by $\mathbb{P}(c \mid x) \propto \exp(\langle \Phi(x), w_c \rangle)$.



Park et al.; ICML ’24

- Assume our model learns a dense, exponentially-smaller approximation of Φ which we call the **feature map** $\Psi : \mathcal{X} \to \mathbb{R}^{d_2}$ with $d_2 \ll d_1$

Note: Compressed sensing theory says that Ψ can represent exponentially many concepts in *superposition*: $d_1 \sim s \exp(d_2/s)$.



- Large concept space Φ
- Superimposed into the feature map Ψ

How does TTT behave?

We trained a sparse autoencoder for ImageNet to learn a concept space $\hat{\Phi}$

1. **Observation**: The learned features Ψ yield similar neighborhoods to neighborhoods in concept space $\hat{\Phi}$.

   also observed in prior work

2. **Observation**: Among a test point x and its neighborhood (in Ψ-space), $f^\star$ can be approximated by an s-sparse linear function in the concept space $\hat{\Phi}$.

   surprising!

3. **Observation**: TTT in Ψ-space implicitly adjusts coefficients based on only a few concepts relevant to the test task.

   predicted by prior work on implicit regularization in sparse recovery

How does TTT behave?

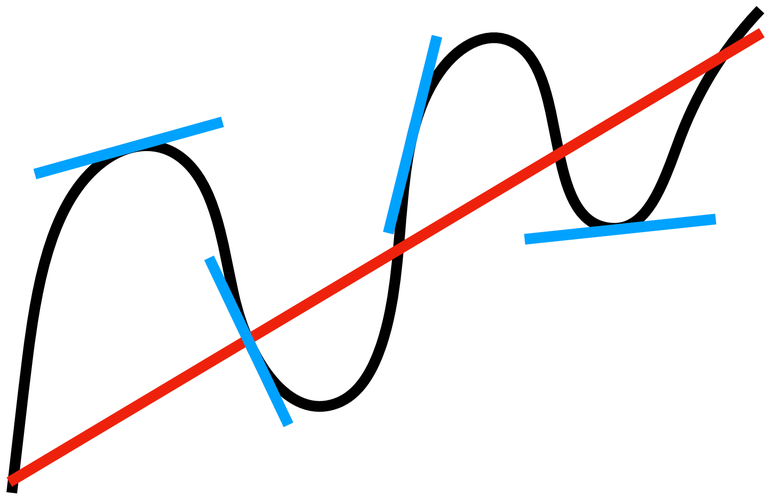
Under **Observations 1-3**, σ^2-subgaussian data & regularity conditions, for any x and sufficiently small neighborhood size k:

$$(f^\star(x) - \langle \Psi(x), \hat{v}_x^{TTT} \rangle)^2 \leq O\left(\frac{\sigma^2 s \log(d_1/s)}{k} \right)$$

the minimax optimal rate from sparse recovery

Under the same assumptions there exist an instance with $f^\star(\cdot) = 1$ where the error of the global model is $\mathbb{E}[(f^\star(x) - \langle \Psi(x), \hat{v}^{global} \rangle)^2] = 1 - d_2/d_1$

Takeaway: TTT *efficiently* learns the meaning of *exponentially many* concepts from data whereas global learning cannot disentangle them.



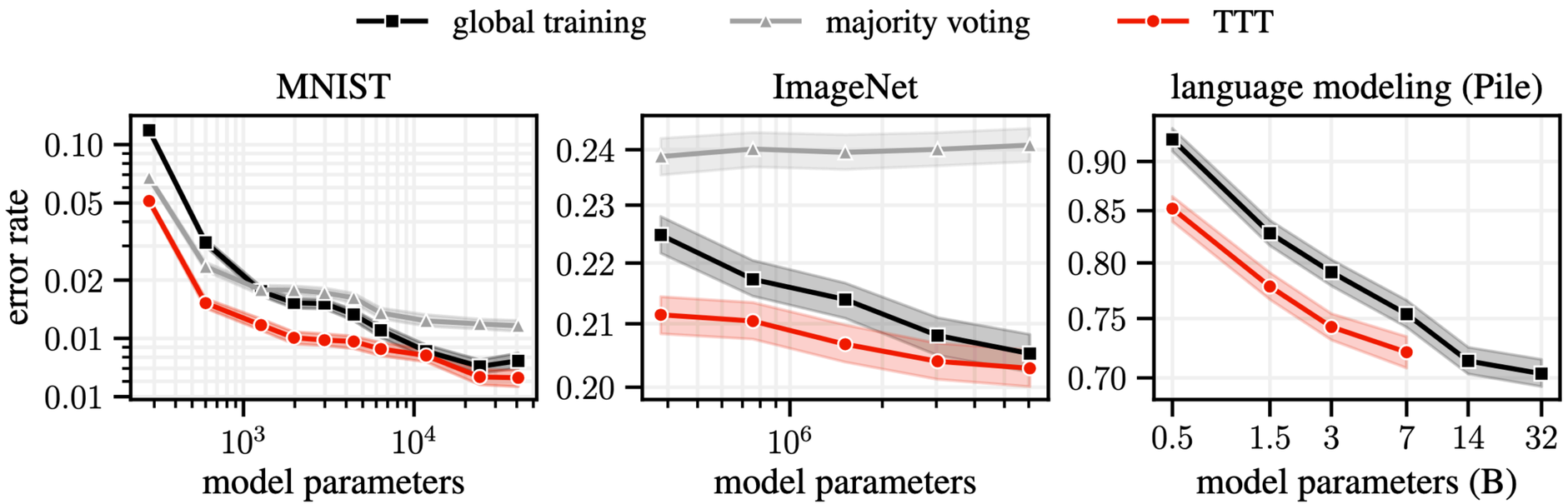
- TTT temporarily “forgets” irrelevant pre-trained knowledge
- This “frees up” capacity to learn relevant concepts at a higher resolution

“specialization after generalization”

learning meaning of concepts

learning concepts

When does (simple) TTT work?



Takeaway: TTT locally improves predictions for underparameterized models, but its improvement diminishes as models become overparameterized.

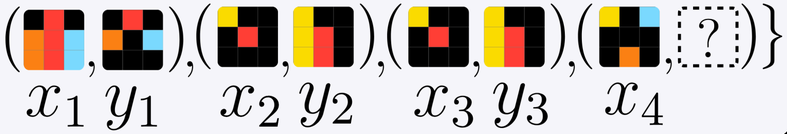
Part 2: Test-time training agents

“I am still learning.” —*Michelangelo, 87 years old*

Warm up: Few-shot learning

- Each test task comes with demonstrations
- Performing TTT on a fine-tuned Llama3 8B:

Test Task



Leave One Out

$\mathbf{LM}(\cdot|x_1, y_1, x_2, y_2, x_3)$
$\mathbf{LM}(\cdot|x_2, y_2, x_3, y_3, x_1)$
$\mathbf{LM}(\cdot|x_1, y_1, x_3, y_3, x_2)$

+ augmentations

Data

Loss on:
All Outputs

$\hat{y}_1$ $\hat{x}_2$ $\hat{y}_2$ $\hat{x}_3$ $\hat{y}_3$

LM

x_1 y_1 x_2 y_2 x_3

Loss
next-token prediction

Task Specific **K Tasks**

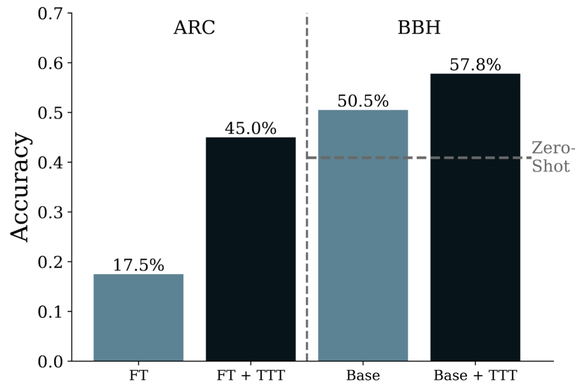
Task 1 $\mathbf{LM}(\cdot \mid d_{\text{TTT}}^{(1)}, x_{\text{test}}^{(1)}, \theta^{(1)})$

Task 2 $\mathbf{LM}(\cdot \mid d_{\text{TTT}}^{(2)}, x_{\text{test}}^{(2)}, \theta^{(2)})$

⋮

Task k $\mathbf{LM}(\cdot \mid d_{\text{TTT}}^{(k)}, x_{\text{test}}^{(k)}, \theta^{(k)})$

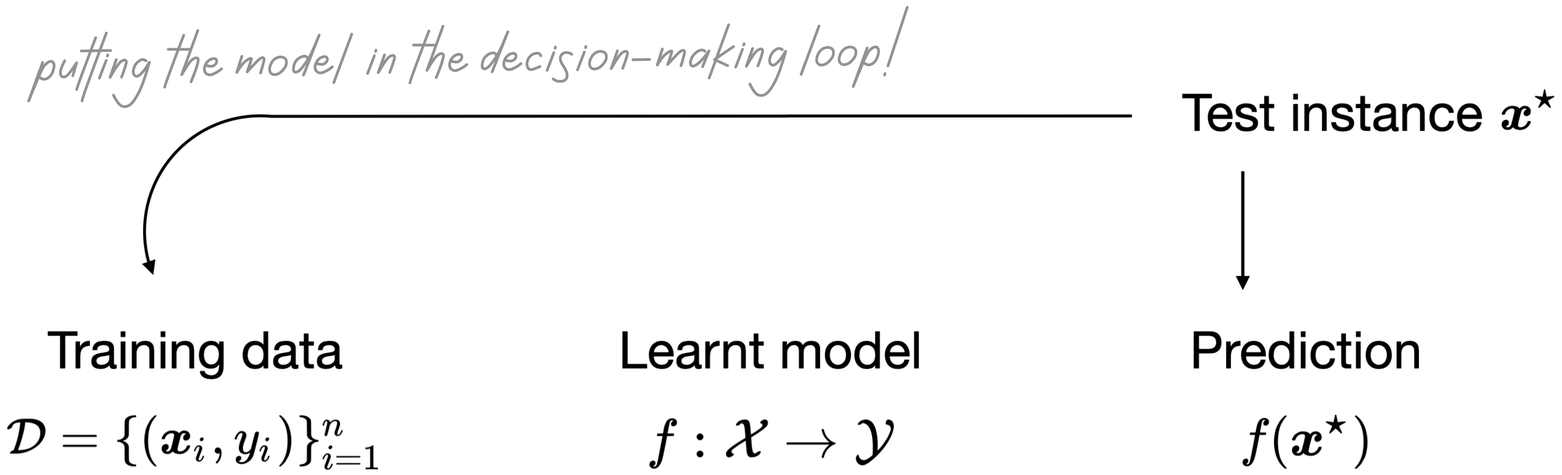
Task-specific models



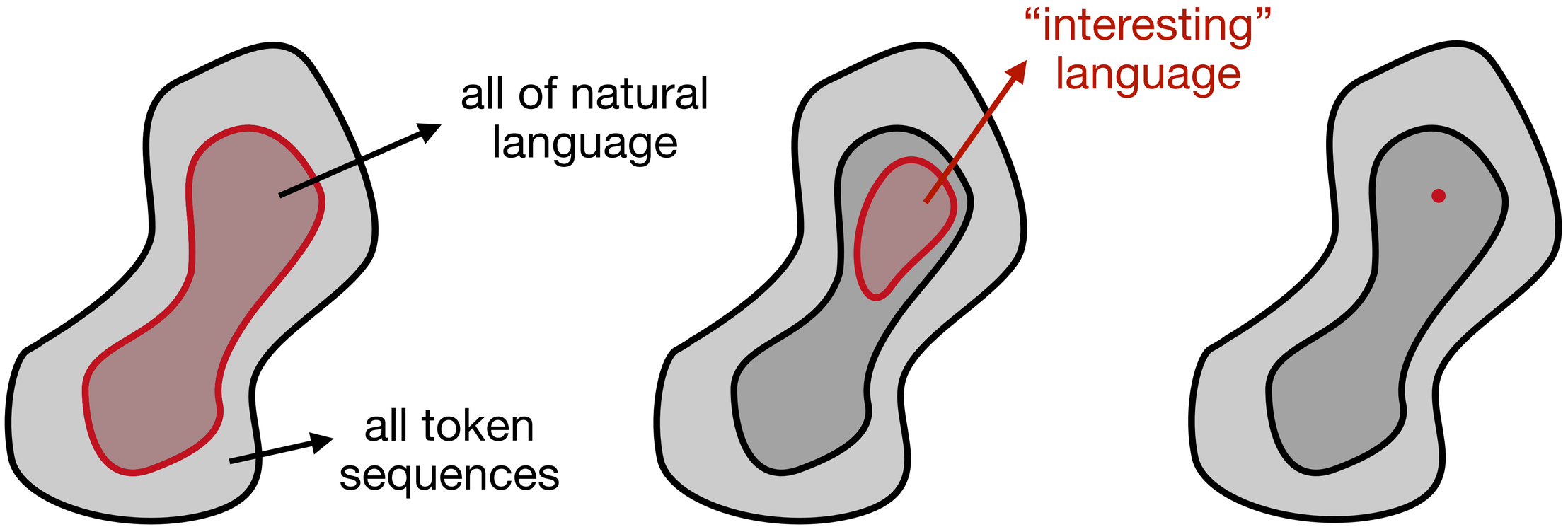
Test-time training **agents**

If task-specific data is not given to us:

- Can we select it from training data?
- Can we generate it ourselves through interaction within an environment?



Why is automatic data selection necessary?



pre-training
human-curated

post-training
human-curated

test-time training
self-curated(!)

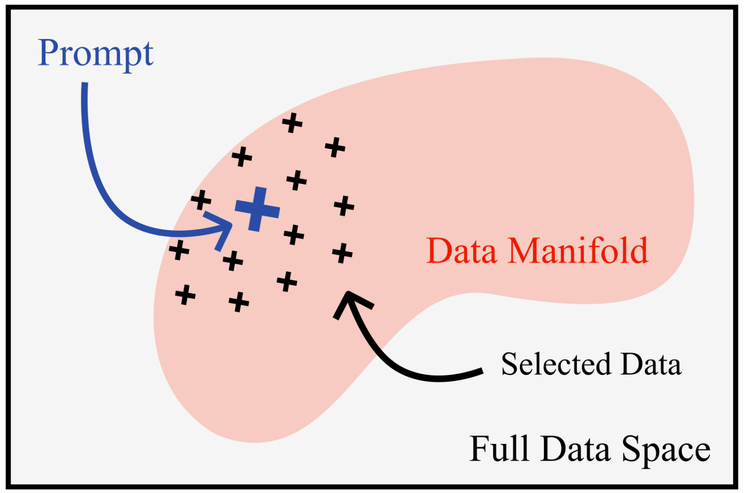
Step 1: Imitating existing data from D

ICLR ‘25

Selecting informative data for fine-tuning (SIFT):

Select data that maximally reduces “uncertainty” about how to solve the task:

$$\text{obs}_n = \text{argmax}\, I(f(x); \text{obs}_n \mid \text{obs}_{<n})$$



Simple TTT procedure:

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

1) Estimating uncertainty

- Making this tractable…

  Surrogate model: approximate model f as logit-linear model in a known representation space

  → linear representation hypothesis

- **Error bound:** $d_{\mathrm{TV}}(f_n(x), f^\star(x)) \leq \beta(\delta)\ \sigma_n(x)$ (with prob. $1 - \delta$)

  error scaling uncertainty

→ $\sigma_n(x)$ measures uncertainty about response to x!

2) Minimizing uncertainty

- SIFT: minimizes uncertainty about prediction for input $x^\star$

$$D_{x^\star} = X_n \cup \{x_{n+1}\} \quad \text{with } x_{n+1} = \operatorname*{argmin}_x \sigma_{X_n \cup \{x\}}(x^\star)$$

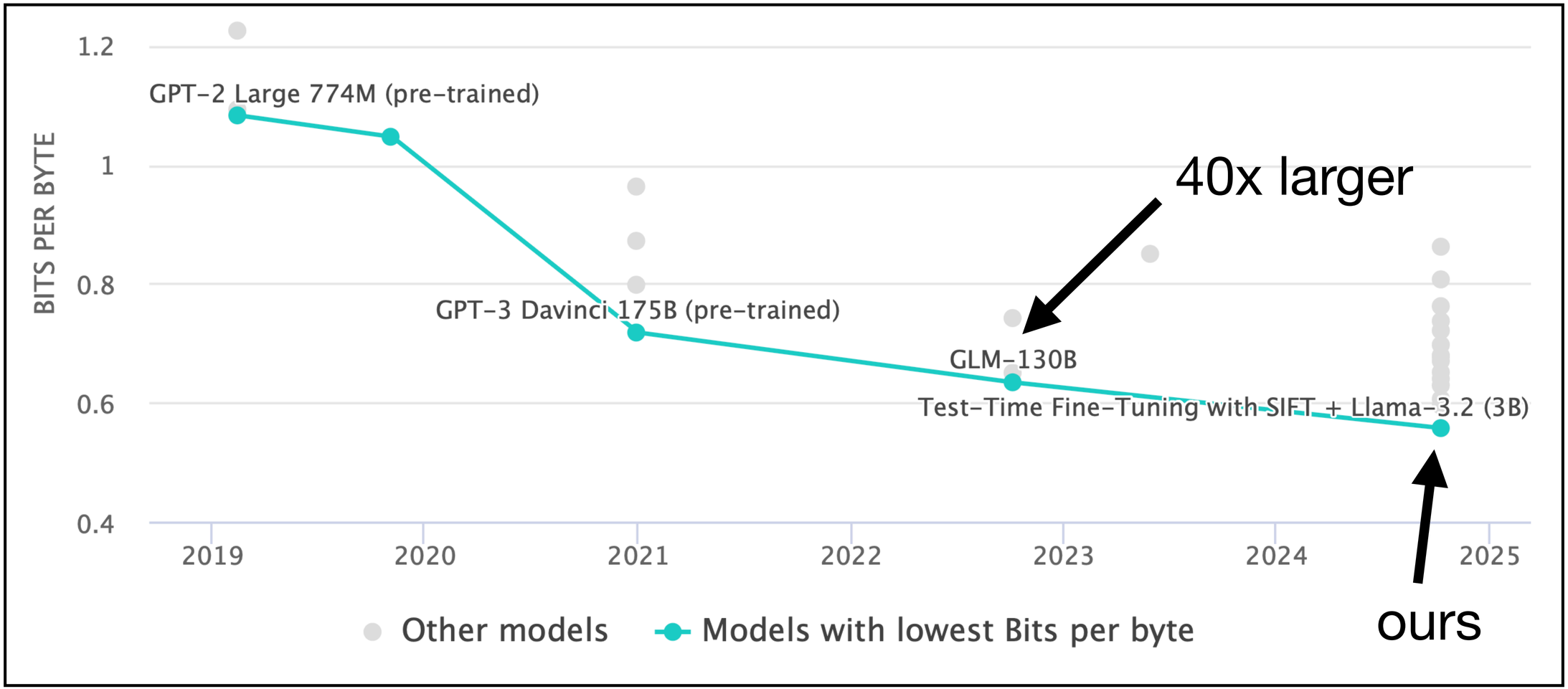
- convergence of uncertainty is guaranteed!

$$\sigma_n(x^\star) \to \sigma_\infty(x^\star)$$

irreducible uncertainty

Takeaway: SIFT guarantees convergence to the smallest possible "uncertainty" given the *available* data and *pre-learned* representations.

New SOTA on the Pile benchmark

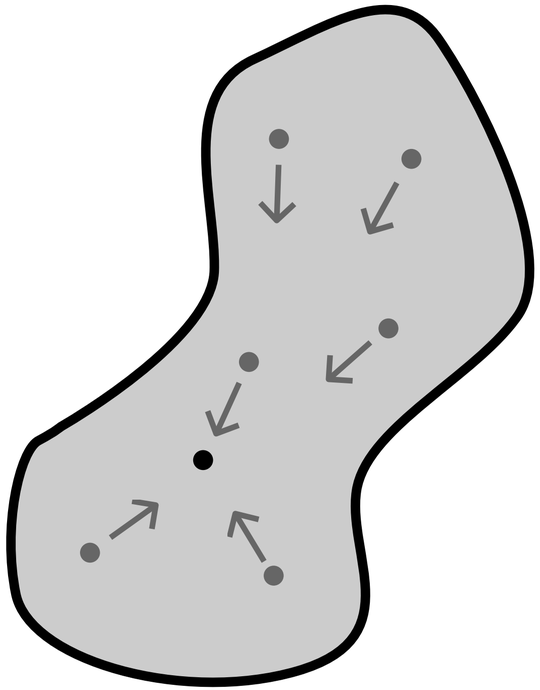


https://paperswithcode.com/sota/language-modelling-on-the-pile

Takeaway: SIFT guarantees convergence to the smallest possible "uncertainty" given the *pre-learned* representations and *available* data.

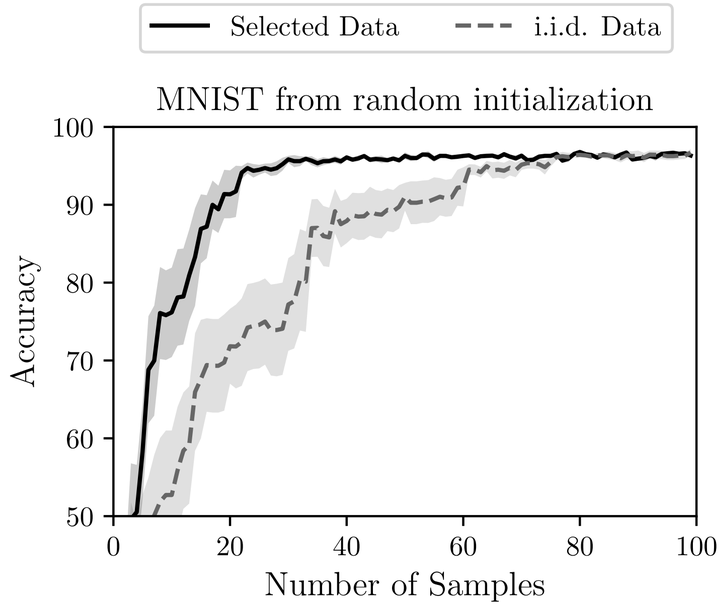
- Can we learn representations over time?
- Can we obtain new data over time?

Q: Can we learn representations over time?



representations

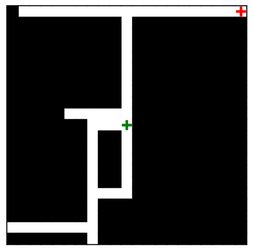
Strong representations can be bootstrapped!



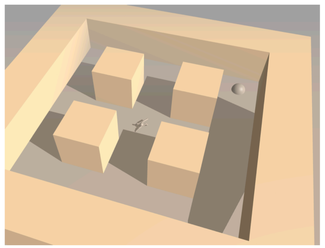
NeurIPS ‘24

Solving “challenging” tasks

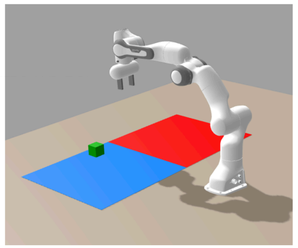
- One family of challenging tasks are *sparse-reward* tasks
- Such tasks require chaining particular actions before observing *any* reward
- **Standard RL approach**: Repeatedly attempt test task



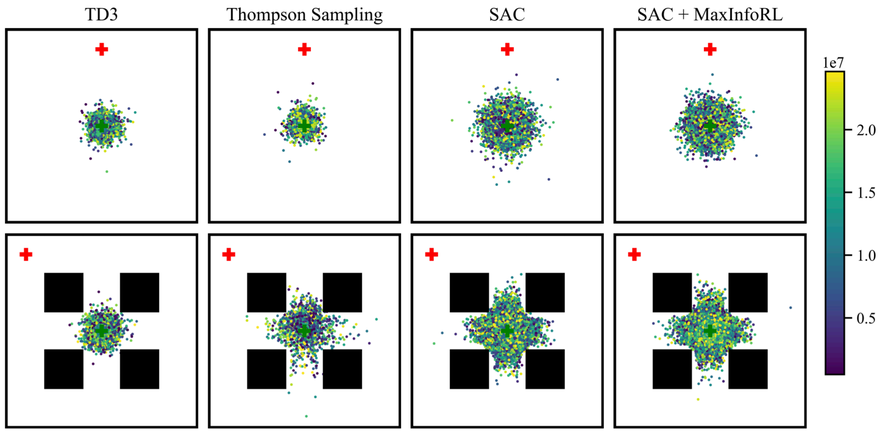
(a) Point maze



(b) Ant maze



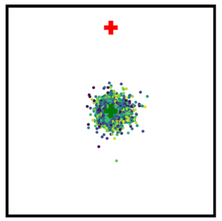
(c) Arm



never observes any reward signal!

Step 2. Collecting new experience

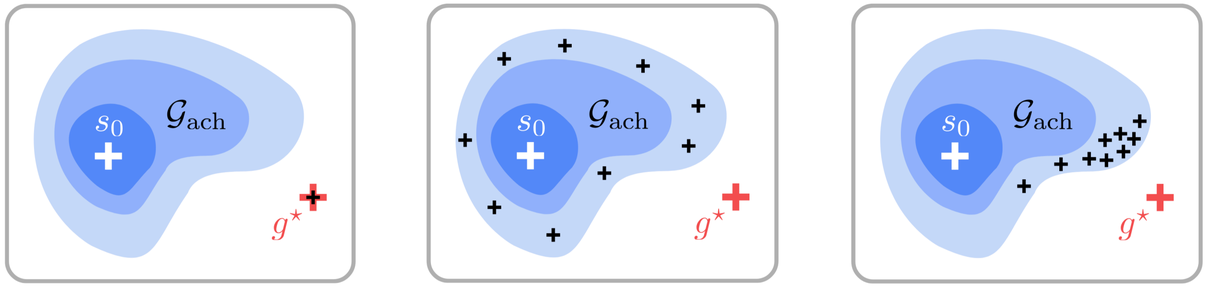
NeurIPS ‘25



Solving previously unsolved tasks requires obtaining *new* experience

→ online interaction with an environment

We ask: What if the agent uses its current understanding of the task landscape to set itself *informative* tasks?



Standard RL | Global exploration | **DISCOVER**

How to determine which tasks to attempt?

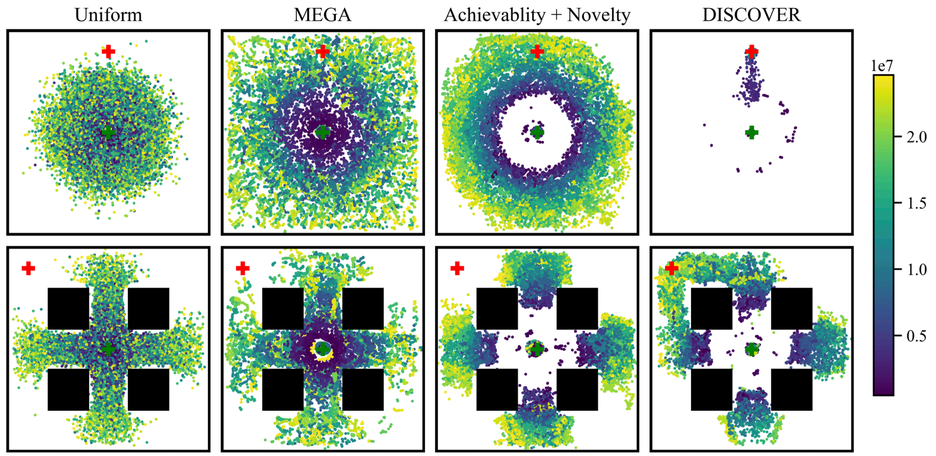
- We learn a task-conditioned value function $V_\theta(g, g')$ which measures the "similarity" of tasks g and g'

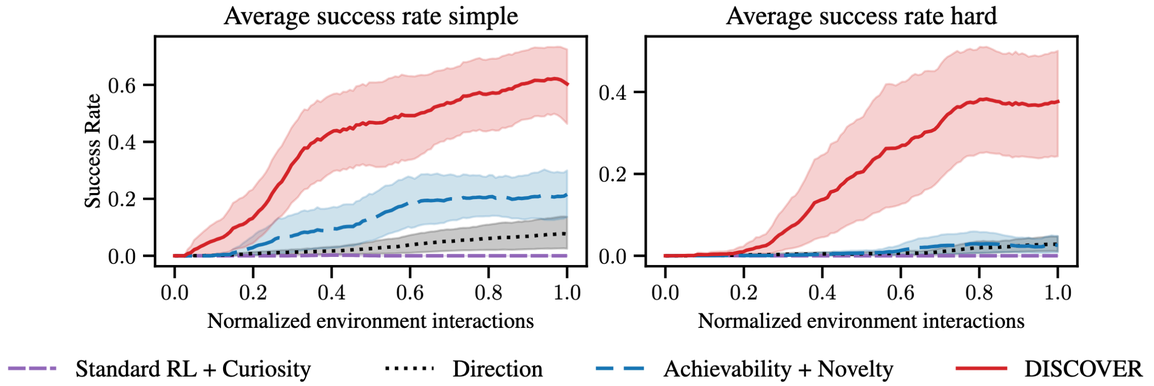
$$\text{task}_n = \underset{g}{\text{argmax}}\ \alpha_n \cdot \text{optimistic } V_\theta(s_0, g) + (1 - \alpha_n) \cdot \text{optimistic } V_\theta(g, g^\star)$$

not too hard & diverse

relevant to test task

- The parameter α_n can be auto-tuned to have 50% task achievability





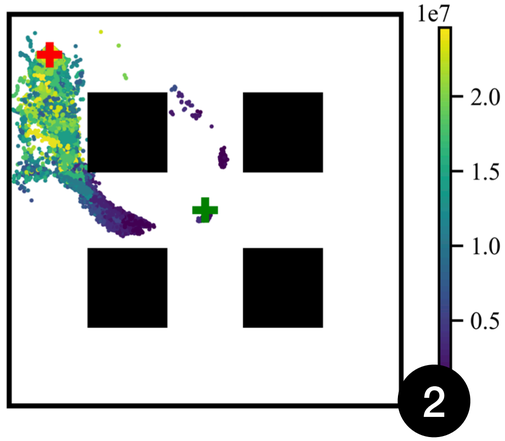
Observations from DISCOVER

1. Targeted exploration is crucial in high-dimensional task spaces
2. DISCOVER can leverage prior knowledge about the task space

Dim.	2	3	4	5	6
HER	∞	∞	∞	∞	∞
MEGA	4.8	∞	∞	∞	∞
Ach. + Nov.	5.2	∞	∞	∞	∞
DISCOVER	**2.9**	**3.1**	**7.4**	**5.4**	**18.7**

1

Required #steps (M) for reaching 10% success rate in point mazes of varying dimensions



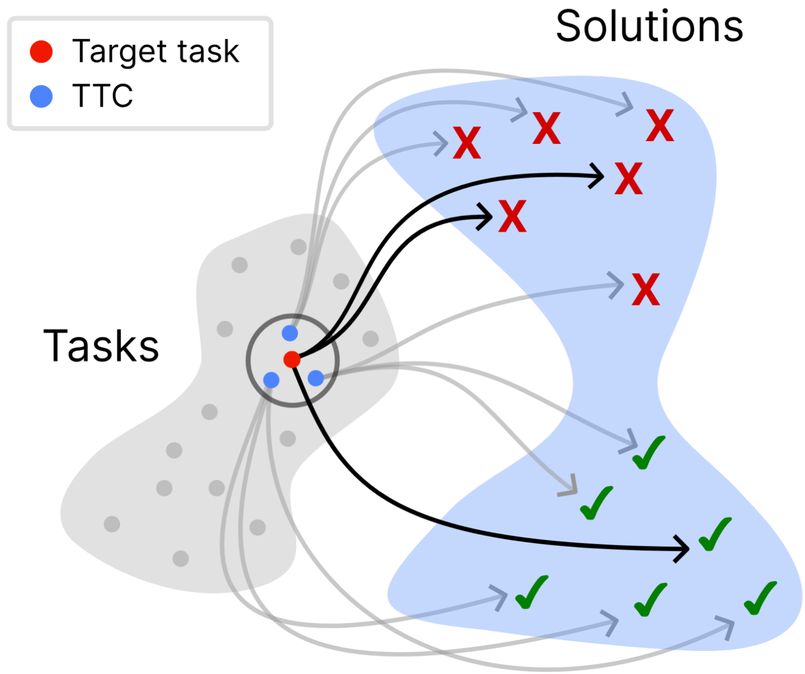
2

Using a pre-trained value function from point maze in ant maze with same layout

Can TTT improve reasoning?

Preprint

- Given a test task, an LLM self-curates a **test-time curriculum (TTC)** of similar tasks for practicing
- The TTC is adaptively selected from a corpus (with SIFT) to balance similarity to the test task and diversity
- We train on the TTC via GRPO

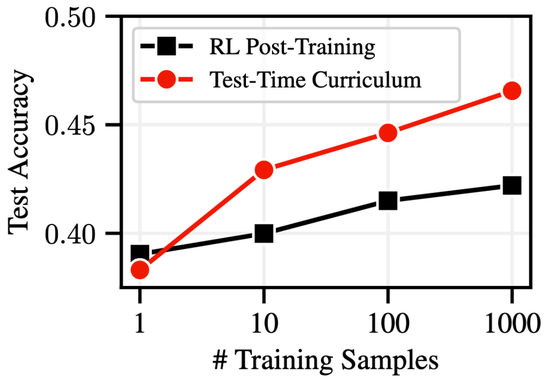


Main results

We treat each benchmark as a set of test tasks, and train on the TTC with RL

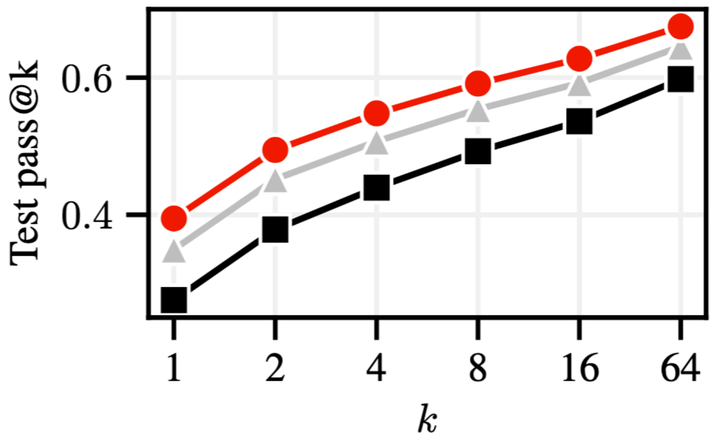
Model	AIME24	AIME25	MATH500	Codeforces	CodeElo	LCB^{v6}	GPQA-D
Qwen3-8B	21.67	23.33	69.55	20.85	13.73	20.61	49.11
+ RL post-training	41.67	38.33	82.50	27.83	22.67	25.95	56.47
+ TTC-RL	**50.83**$^{+29.2}$	**41.67**$^{+18.3}$	**85.10**$^{+15.6}$	**33.35**$^{+12.5}$	**29.34**$^{+15.6}$	**27.29**$^{+6.7}$	**58.38**$^{+9.3}$
Qwen3-4B-Instruct-2507	52.50	40.83	72.00	26.70	20.27	21.56	61.93
+ RL post-training	55.83	**47.50**	86.30	28.39	21.18	25.95	**62.82**
+ TTC-RL	**60.00**$^{+7.5}$	45.83$^{+5.0}$	**88.50**$^{+16.5}$	**34.99**$^{+8.3}$	**27.20**$^{+6.9}$	**26.91**$^{+5.4}$	61.93$^{+0.0}$
Qwen3-8B-Base	15.83	14.17	63.10	9.92	6.67	11.26	29.70
+ RL post-training	22.50	20.83	76.85	17.46	9.97	**18.51**	42.77
+ TTC-RL	**30.00**$^{+14.2}$	**21.67**$^{+7.5}$	**78.15**$^{+15.1}$	**17.84**$^{+7.9}$	**11.33**$^{+4.7}$	17.94$^{+6.7}$	**45.94**$^{+16.2}$

Pass@1 accuracy on reasoning tasks of
- base model
- model after global RL post-training
- TTC-RL

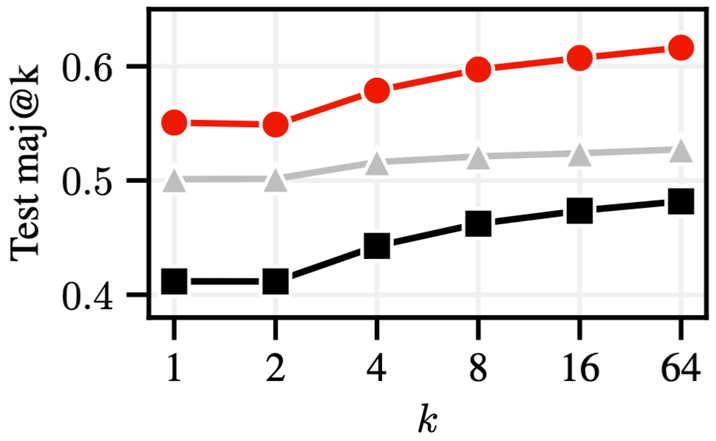


Takeaway: TTC-RL consistently achieves a higher pass@1 than general-purpose RL post-training & learns from significantly fewer attempts.

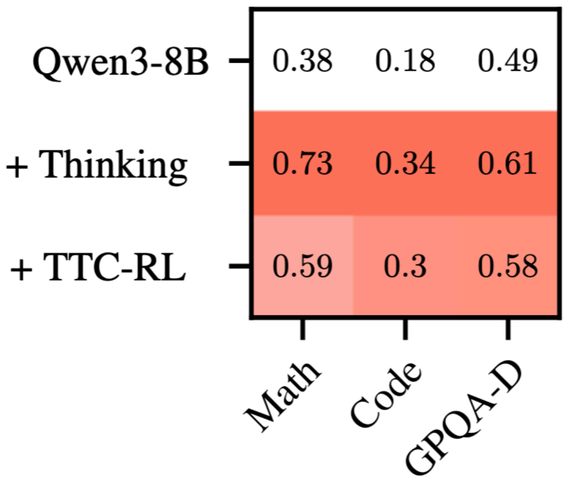
TTC-RL with additional test-time scaling



Comparison to **majority voting**:

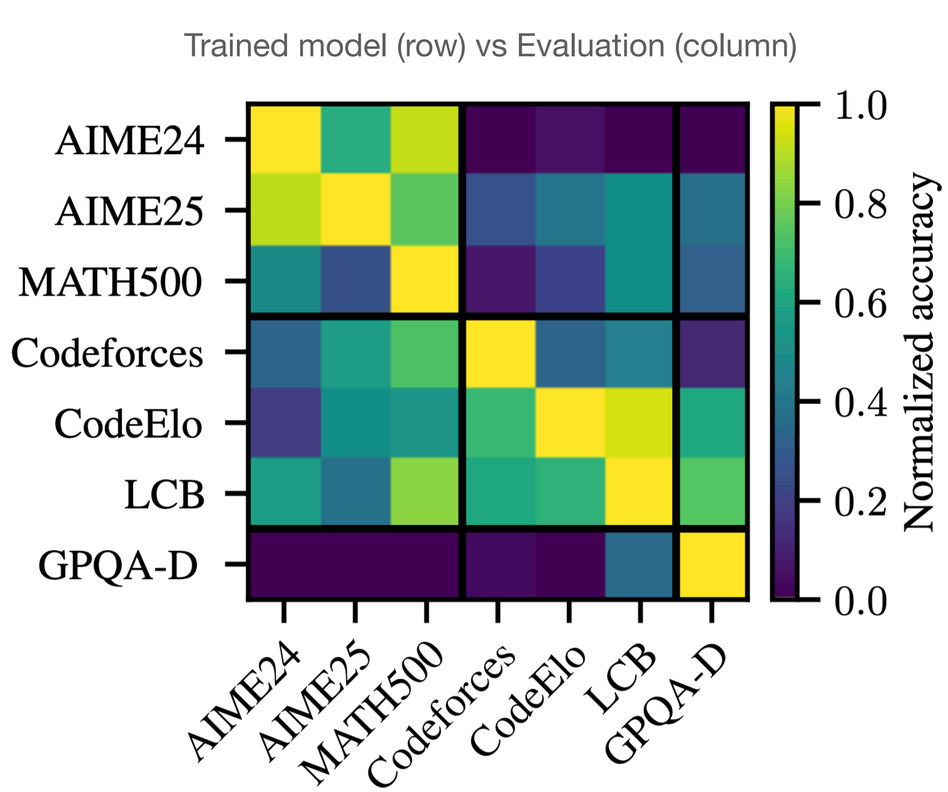


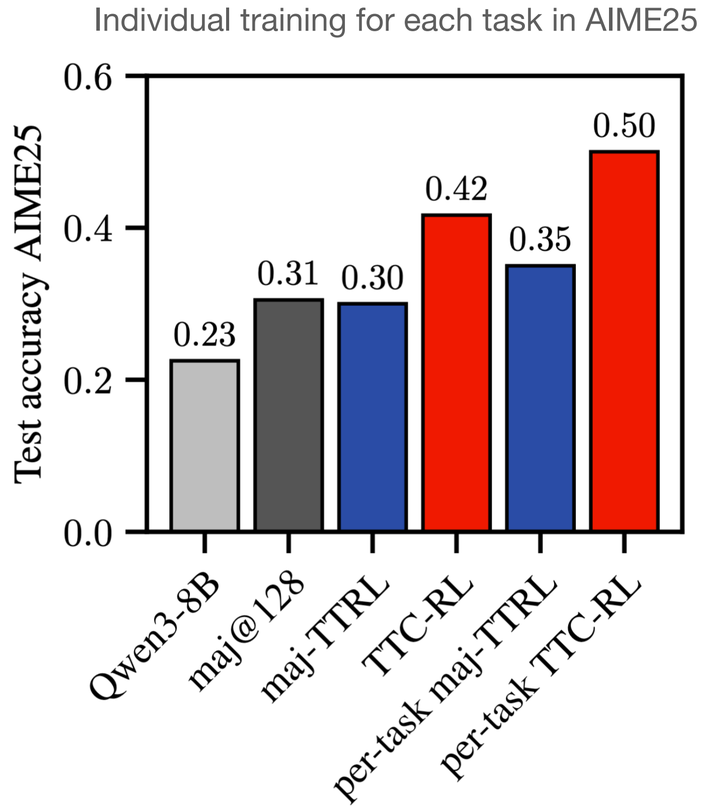
Comparison to **scaling thinking**:
8k context + TTC-RL vs 30k context



Takeaway: TTCs can complement existing methods for test-time scaling such as majority voting & reasoning in-context.

Do TTCs specialize models?





Takeaway: TTCs effectively specialize models to their target tasks.

Conclusion

TTT is a class of methods for specializing “foundation” models to individual tasks

1. TTT can enable models to outperform globally trained models *in-distribution*.
2. TTT enables models to continue learning & improving at test-time.

Happy to talk more!

jonas.huebotter@inf.ethz.ch