Research Statement

Amrith Setlur (Carnegie Mellon University)

Scaling ML pretraining on web data has produced impressive results, such as models that can win gold medals at IMO [A] and accurately predict protein structures [20]. However, when faced with tasks that are not well-represented in available datasets (*e.g.*, time-series data in finance [25]), models can falter and hallucinate incorrect answers [B]. These shortcomings echo long-standing failures in deep learning where models fail arbitrarily under atypical inputs (*e.g.*, adversarial perturbations [5] or distribution shifts [8]). In contrast, humans solve unseen challenges and reason about uncertainties with more time available to "think" and "interact" with the test environment. This hints at an alternative paradigm of learning, where trained models can spend more computation on difficult test instances and thus perform better on them, *i.e.*, models that can adapt at test-time.

My work aims to develop foundation tools for test-time adaptation (TTA) that enable a new generation of generalist and effective AI systems. The success of many real-world systems is driven by some primitive forms of TTA: YouTube adapts per-user recommendations on the fly [C], AlphaGo and AlphaFold implement a form of test-time search [19], and modern large language models (LLMs) like OpenAI-o3/DeepSeek-R1 [18] scale test-time compute to reason more deeply, "think" for longer, and sample more tokens. While these systems benefit from TTA, their core AI components are still trained using ML paradigms which produce models that do not adapt ("static predictors"), and this often limits TTA to fixed adaptation procedures with ad hoc design choices (*e.g.*, recommender systems still handcraft user features). My work identifies and solves key barriers that exist to realize the full potential of the TTA paradigm. First, we need expressive models that can easily implement sophisticated test-time procedures (*e.g.*, self-verifying a prediction, backtracking). Second, we lack training data that can teach models "how to adapt" (*e.g.*, how to search through different solutions for a math problem), as opposed to "what to predict" directly (final answer). Third, we don't have scalable algorithms for TTA and the bitter lesson [D] tells us that, for an algorithm to work well, it must be effective at scaling data and compute (like scaling maximum likelihood estimation (MLE) on web data). To address these issues, my work thus far has considered ***how to collect training data and develop learning algorithms for TTA*** (Figure 1).

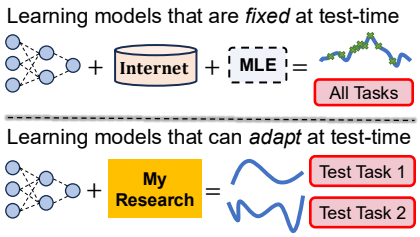

Figure 1: Learning to adapt at test-time.

To begin with, I demonstrated how TTA can overcome longstanding ML challenges like robustness to distribution shifts [17, 3, 32, 13, 14, 4] and data privacy [15, 26]. My recent works [34, 35] use principled frameworks to build LLMs that adapt at test-time and outperform some industry trained models. Concretely, my work answers:

- **What training data teaches adaptation?** Most datasets are suited for old-school static prediction, *i.e.*, data that maps inputs to labels. But for TTA, we need data that explains how to arrive at the prediction, or what information to seek before making one. For this, I explored the avenue of *synthetic/model-generated* data to expose patterns of desired test-time behaviors like querying, checking, and refining [16, 6, 31, 34].
- **Which learning algorithms enable TTA?** First, I studied *parameterizations* (*e.g.*, modern day LLMs/VLMs) that let a fixed model ingest arbitrary context (*e.g.*, few-shot exemplars) and adapt freely. Second, I used insights from algorithms that train static predictors to develop *scalable* algorithms for TTA (my work was the first to show the benefits of scaling RL [33] and use it to train LLMs that can implement test-time search [35]). Third, I pushed on *test-time efficiency and information-seeking*: allocating compute where it matters (*e.g.*, fewer tokens in LLMs) and proactively querying for informative feedback before answering [33, 35, 27, 10, 34, 21].

Impact of my work. My work on data [16] and algorithms [34, 33] for TTA is built upon by empirical [12, 24] and theory communities [29]. My blog [E] (38k+ views) was the first to formalize TTA as a meta-RL optimization problem and generated strong interest from teams at Cursor, ServiceNow, and Meta, who reached out for discussion. I also developed LLMs that scaled test compute to discover knowledge beyond training data [F], culminating in the best <2B math-reasoning model e3-1.7B [35] with 10k+ downloads (Aug–Oct '25). My algorithms for TTA are taught in ML courses (CMU 10-703, 10-605) and my works have won multiple spotlights at ICML, ICLR conferences and orals, best-paper awards at top ICML, ICLR and NeurIPS workshops in 2025.

Looking forward, building on my work, I aim to develop unified analysis, practical workflows and algorithmic tools that make test-time adaptation a principled, reliable, and scalable capability. Despite the initial success of TTA in applications (*e.g.*, AlphaGo and "thinking" models like o3), we are just scratching the surface of methods for effective TTA. My ultimate goal is to develop *training-sample* efficient algorithms that can learn *test-compute* efficient models, *i.e.*, models that can persist and keep trying safely and robustly until they solve hard problems that demand open-ended exploration in applications such as drug discovery and materials design (Section 2).

1 Past Research

Early promise of TTA. Early in my Ph.D., I worked on the robustness of static predictors to train/test mismatch and found a hard limit: elaborate algorithms (*e.g.*, DRO [17], contrastive learning on web data [3, 32]) remained *provably* susceptible to *spurious* features: correlations that hold during training but not at test-time (*e.g.*, background may correlate with the bird in the training images [11]). I showed that test-time adaptation can breach this limit and unlearn spurious features *memorized* during training, if we can design domain-specific test-time procedures (*e.g.*, prompt VLMs to label the spurious feature [32], linear probe over trained features at test-time [13], or self-train on randomly augmented inputs to attenuate spurious features [3]). Motivated by my work [13, 31, 3, 4, 1, 17], I pushed to make TTA a generally applicable, scalable and performant capability. Before that, a fundamental question: what data provides the best signal to train models that can adapt at test-time?

1.1 What Kind of Training Data Enables Learning to Adapt at Test-Time?

For learning models that can adapt at test-time, the "go-to" approach is to collect training data with intermediate steps that teach models how to meaningfully adapt (*i.e.*, they describe the sequence of test-time operations that can make the final prediction of the model more accurate); but existing datasets provide only the final label, and not the operations that led to them. To fill this gap, I used model-generated ***synthetic data***. I realized the promise of synthetic data in my work [31] on how loss-maximizing perturbations can generate synthetic data that can make it easy to unlearn spurious features. I also extended this idea to reduce data memorization in LLMs [23, 6].

For TTA of LLMs, in [16] I used the instruction-following capabilities of the modern-day LLMs and prompted them to collect synthetic data that outline the test-time operations (*e.g.*, steps that lead to the correct answer on math problems). If a synthetic trace ends in a correct final answer, we call it *positive*; otherwise *negative*. The obvious way of training solely on positive synthetic data surprisingly doesn't scale: because only the final answer is verified, and intermediate operations can be spurious (incorrect, repetitive, or derailing). Inspired by advantage-weighted RL [9] and my work [7] on counterfactual data augmentation to combat spurious correlations, I scaled positive and negative synthetic data via *branched rollouts*, *i.e.*, rolling out the LLM from intermediate steps (Fig. 2) to identify and prune spurious steps by solving the *credit assignment* problem. This improved performance by $8\times$. Applying my insights on synthetic data to larger models, Google later showed that scaling test-time compute can outperform scaling pre-training model size [12].

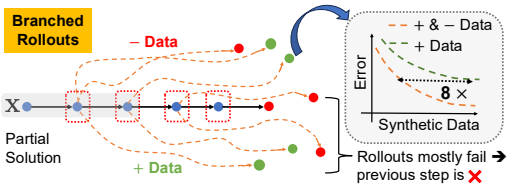

Figure 2: Scaling branched synthetic data → $8\times$ gains.

1.2 Which Data-Driven Algorithms Enable Test-Time Adaptation?

TTA demands models that implement a sequence of useful operations targeted at learning more about the test input. Consequently, a ***key insight*** that I based my research on is: learning to do TTA is nothing but "learning to learn" or ***meta-learning***. In my work [30], I found that while TTA afforded by popular meta-learning algorithms at the time [2] improved robustness to distribution shifts, these algorithms often yielded models that memorized training data and directly predicted labels without any real TTA. One reason for this was restricting TTA to fixed adaptation procedures (*e.g.*, running gradient descent on a few labeled examples similar to the test input).

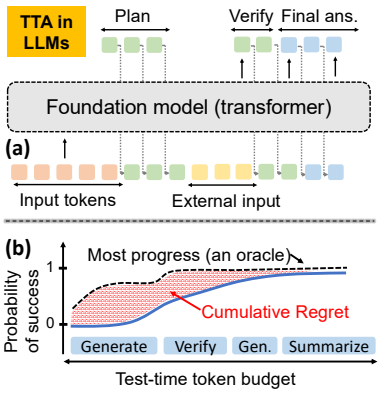

Figure 3: (a) LLMs can scale test-time compute to implement in-context search. (b) To be effective the LLM should reduce cumulative regret with more tokens.

Meta-RL: A principled framework for TTA algorithms. To develop models that can adapt more freely and without any preset biases on the adaptation procedure, I moved towards foundation models (LLMs/VLMs): (i) in theory their auto-regressive generative nature enables them to adapt more expressively (including running gradient descent [22] on in-context examples); (ii) I showed they can be fine-tuned to reliably invoke useful test-time skills such as self-verification [33] and planning [10]. My blog [E] was the first to show that LLMs, if trained properly can perform test-time search by simply sampling more tokens, *i.e.*, repeatedly plan, generate responses, and verify them, until a final answer can be produced with greater confidence (see Fig. 3a), and that learning such LLMs requires solving a meta-RL problem [27]. Equipped with a more principled framing of learning TTA through meta-RL, in [27] I showed that we can learn LLMs that optimally use test-time compute by minimizing a common metric in sequential decision making: the cumulative regret over the sampled tokens, measured against an oracle that makes maximal progress with every new token (Fig. 3b). Next, I used these insights to develop scalable (with more training data/compute) and practical algorithms for TTA.

Scalable algorithms for TTA. In [33], I proved a ***theoretical separation*** between two classes of algorithms to train LLMs for TTA: (i) *verification-free* or *SFT* that trains models by maximizing likelihood on expert traces; and (ii) *verification-based* or *RL* that trains the models on self-generated traces scored against a verifier learned on suboptimal data. I showed that RL is more sample-efficient than SFT and the performance gap between them grows as we increase the test-time compute (token) budget available to the trained LLM (Fig. 4), implying that RL is better suited to learn LLMs that can benefit from larger test-time budgets (also shown by industry labs that train models to use over 100k tokens [18]). But, the true promise of TTA is in improving performance as we sample far more tokens than seen during training, *i.e.*, as we *extrapolate* test-time compute (*e.g.*, an LLM that can search over solutions by simply sampling for longer [E]).

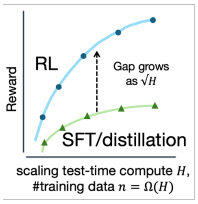

Figure 4: Gap b/w RL and SFT grows.

LLMs that can search at test-time. In [35] I presented key design principles for RL to teach LLMs scalable test-time operations like search [D]. Naïve outcome-reward RL on a base model only sharpens the distribution of the pre-trained model on correct solutions it can already sample, but not discover new solutions for any unsolved problems [F] (Fig. 5a). To fix this, first, I showed that it is essential for the base LLM to present *asymmetric skills* (*e.g.*, LLMs that can verify responses more accurately than generate the correct one), which can then be chained (*e.g.*, verify→generate→revise) to discover the right answer. In the absence of these, in [10] I showed that we can also build new skills into the pre-trained LLM like generating useful abstractions (plans, hints). Now that the model can explore over the chaining of skills, as opposed to raw tokens, in [35] I proved that negative gradients in RL can amplify the chaining of these skills in diverse ways, and we can further structure this exploration via a *curriculum* over token budgets and task sets. Together, this resulted in ***e3-1.7B: the best <2B math-reasoning model*** that can extrapolate compute better than much larger 7B/32B models trained at industry scale (Fig. 5b). In [34, 27], I also showed that dense rewards in RL are key to improving exploration on hard problems failed by the base model.

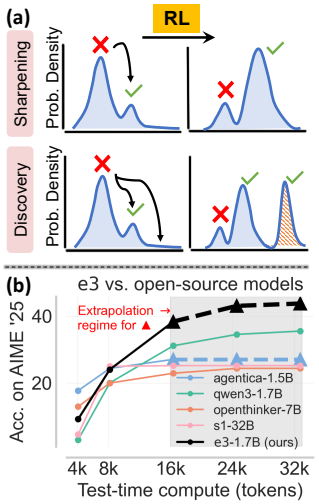

Figure 5: (a) Naïve RL only sharpens the distribution over known solutions. My work [35] gets RL to discover new solutions, yielding the best <2B model (b).

TTA that is efficient and information-seeking. Ideally we want TTA models to spend test compute efficiently (*e.g.*, LLMs don't oversample tokens on easy problems) and that for every unit of compute spent (*e.g.*, a reasoning step) they make *progress*, *i.e.*, improve the likelihood of future success. We also want them to be information-seeking and query the test environment whenever they can't make progress. My work on *process advantage verifiers* [34] formalized automated, step-level rewards via a class of process rewards that measure *progress* under a separate prover policy. This improved RL's rate of convergence by 6×, trained LLM's test compute efficiency by 10×, and was the ***first result to show the benefit of dense rewards*** for RL training of LLMs. In [21], we explored a new axis of scaling: test-time interactions (TTI). Here, an LLM is trained to not just sample more tokens to solve a problem, but also query the environment to collect informative feedback (*e.g.*, navigate and explore web pages on a site before answering a user query). Extending ideas in [35] to the TTI setting, we built a state-of-the-art open-source web agent [21].

2 Future Research Agenda

My work on test-time adaptation (TTA) provides a principled framework for practical learning algorithms, and tools for analyzing models that can improve performance with reasoning, search, and test-time interaction. But several core questions remain before TTA becomes fully dependable and a "first-class" capability in modern-day AI systems. To close these gaps, I will lead a research program organized into the following synergistic thrusts:

Best-practices and easy-to-deploy TTA. Predictable scaling laws have powered progress in training static predictors (*e.g.*, laws that predict optimal batch size and model size for LLM pre-training). To make TTA equally dependable, we need analogous laws that inform us about best and stable hyper-parameter configurations for nuanced algorithms like dense-reward RL. This means diagnosing and fixing failure modes such as RL-induced diversity collapse [28] and turning them into actionable design principles. Building on my work [30, 23, 3] [F], I will develop algorithms with practitioner-oriented workflows (*e.g.,* default configs, safety checks, diagnostics). Ultimately, success is real-world usability: methods that run out-of-the-box, and tune predictably.

Open-ended exploration, lifelong TTA & continual learning. In open-ended domains like materials science and drug discovery, exploration gets extremely challenging for TTA models since: (i) the learning signal from costly verifiers like wet-lab tests is very weak and labeled data is too limited to learn reliable reward models; (ii) with access to limited guidance, models need to *continually learn* about new test domains through interactions

and search that runs for days (*e.g.*, over a large space of crystal and protein compositions); and (iii) open-ended domains are less present in internet data, requiring us to leverage synthetic data, and build autonomous models with information-seeking capabilities. I will make TTA models applicable in open-ended settings by building on my works on synthetic data [16, 31, 6, 23], dense reward models [34, 27], information-seeking LLMs [21], structured exploration in RL [33, 35, 10], and guided exploration for scaling LLM RL on hard problems [G].

Pre-training for TTA. Today's recipe for *adaptive* models is somewhat paradoxical: we just post-train models pre-trained to be good *static* predictors (*e.g.*, post-train LLMs with RL) and hope they magically adapt at test time. This two-stage approach breaks down when the pre-trained model lacks useful test-time skills [35, 10] (like self-verification) for post-training to amplify. Going beyond human data imitation, I will make TTA *intrinsic* by studying data, model architectures, and learning objectives to pre-train models from scratch on internet data with the goal of explicitly teaching them *how to adapt* to unseen test domains. Grounded in my meta-RL view of TTA [E] [27], I will build on my works on scaling test-time compute [33, 34], and synthetic data [31, 16].

Advancing real-world applications. I will iterate on applications in lockstep with algorithms. I already witnessed impactful results when I used ideas from my early research on TTA [3, 32, 13] (with fixed test-time adaptation procedures like linear probing) to improve the utility of privacy-preserving algorithms [15, 26]. Similarly, in the case of foundation models (that can adapt more broadly [35]), I will build on my work in LLM math reasoning [34, 10, 16] and web agents [21]. Concretely, I will extend it to harder domains such as drug discovery and protein design where exploration is combinatorial and verification is costly/noisy. TTA is broadly applicable across ***sequential decision-making systems*** that power robots, nuclear reactors, and power grids. Traditionally, these are solved via RL that yields domain-specific specialists and demands large-scale online/offline data collection. By using TTA's connection to meta-RL [27, 33] [E], I intend to train generalist models on internet data that can solve sequential decision-making problems with minimal data collected from the test environment.

[1] A. Chen, Y. Lee, **A. Setlur**, S. Levine, and C. Finn. When to take shortcuts for subpopulation shifts. In *Arxiv*, 2023.
[2] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.
[3] S. Garg* and **A. Setlur*** , A. Raghunathan et al. Complementary benefits of contrastive learning and self-training. *NeurIPS*, 2023.
[4] G. Ghosal* and **A. Setlur*** et al. Contextual reliability: When different features matter in different contexts. In *ICML*, 2023.
[5] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *NeurIPS*, 2014.
[6] K. Kang, **A. Setlur**, C. Tomlin, and S. Levine. Deep neural networks tend to extrapolate predictably. In *ICLR*, 2024.
[7] D. Kaushik, **A. Setlur**, E. Hovy, and Z. C. Lipton. Explaining the efficacy of counterfactually augmented data. In *ICLR*, 2021.
[8] P. W. Koh, S. Sagawa, and et al. Wilds: A benchmark of in-the-wild distribution shifts. In *ICML*, 2021.
[9] X. B. Peng, A. Kumar, G. Zhang, and S. Levine. AWR: Simple and Scalable Off-Policy Reinforcement Learning. In *ICLR*, 2020.
[10] Y. Qu* and **A. Setlur** , A. Kumar et al. RLAD: Training LLMs to discover abstractions for reasoning problems. In *Arxiv*, 2025.
[11] S. Sagawa, P. W. Koh, T. Hashimoto, and P. Liang. Distributionally robust neural networks. In *ICLR*, 2019.
[12] C. Snell, K. Xu, and A. Kumar. Scaling test-time compute optimally can be more effective than scaling parameters. In *ICLR*, 2025.
[13] A. Chen, **A. Setlur**, C. Finn et al. Project & probe: Domain adaptation by interpolating orthogonal features. In *ICLR*, 2024.
[14] A. Kulkarni, L. Dery, **A. Setlur**, G. Neubig et al. Multitask learning can improve worst-group outcomes. In *TMLR*, 2024.
[15] **A. Setlur**, V. Feldman, K. Talwar. In *NeurIPS*, 2024.
[16] **A. Setlur**, V. Smith, A. Kumar et al. RL on incorrect synthetic data scales the efficiency of llm reasoning by 8x. In *NeurIPS*, 2024.
[17] **A. Setlur**, V. Smith, S. Levine et al. Bitrate-constrained dro: Beyond worst case robustness to group shifts. In *ICLR*, 2023.
[18] D. Guo, D. Yang , H. Zhang et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 2025.
[19] D. Silver et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 2016.
[20] J. Jumper, R. Evans, A. Pritzel, T. Green et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, 2021.
[21] J. Shen*, H. Bai*, **A. Setlur** et al. Thinking vs. doing: Agents that reason by scaling test-time interaction. In *NeurIPS*, 2025.
[22] J. Von Oswald, M. Vladymyrov et al. Transformers learn in-context by gradient descent. In *ICML*, 2023.
[23] K. Kang, **A. Setlur**, A. Kumar et al. What do learning dynamics reveal about generalization in LLM reasoning? In *ICML*, 2025.
[24] L. Yuan, W. Li et al. Free process rewards without process labels. In *ICML*, 2025.
[25] M. Tan, M. Merrill, T. Hartvigsen et al. Are language models actually useful for time series forecasting? *NeurIPS*, 2024.
[26] P. Thaker, **A. Setlur**, Z. S. Wu, V. Smith. On the benefits of public representations for private transfer learning. In *NeurIPS*, 2024.
[27] Y. Qu*, M. Yang*, **A. Setlur**, A. Kumar et al. Optimizing test-time compute via meta reinforcement fine-tuning. In *ICML*, 2025.
[28] Y. Yue et al. Does RL really incentivize reasoning capacity in LLMs beyond the base model? In *NeurIPS*, 2025.
[29] Z. Jia, A. Rakhlin. Do we need to verify step by step? process supervision from a theoretical perspective. In *ICML*, 2025.
[30] **A. Setlur**, O. Li, and V. Smith. Two sides of meta-learning evaluation: In vs. out of distribution. *NeurIPS*, 2021.
[31] **A. Setlur**, B. Eysenbach, V. Smith, and S. Levine. Adversarial unlearning: Reducing confidence adversarially. In *NeurIPS*, 2022.
[32] **A. Setlur**, V. Smith, and S. Levine. Prompting is double-edged: On worst-group robustness of foundation models. In *ICML*, 2024.
[33] **A. Setlur**, N. Rajaraman, S. Levine, and A. Kumar. Scaling test-time compute without RL is suboptimal. In *ICML*, 2025.
[34] **A. Setlur**, Aviral Kumar et al. Rewarding progress: Scaling automated process verifiers for LLM reasoning. In *ICLR*, 2025.
[35] **A. Setlur**, Aviral Kumar et al. e3: Learning to explore enables extrapolation of test-time compute for LLMs. In *Arxiv*, 2025.