RL + Transformer = A General-Purpose Problem Solver

Micah Rentschler
Tennessee Technological University
mrentschler@tntech.edu

Jesse Roberts
Tennessee Technological University
jroberts@tntech.edu

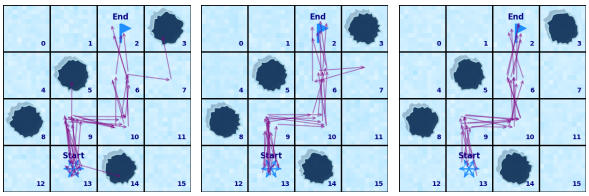
Abstract

What if artificial intelligence could not only solve problems for which it was trained but also *teach itself* to tackle novel tasks? In this paper, we finetune Llama 3.1 using reinforcement learning on the grid-world game Frozen Lake and investigate its ability to solve maps it has *never encountered*—a phenomenon recently termed **In-Context Reinforcement Learning (ICRL)**. Without additional training, the transformer demonstrates the capacity to adapt to both in-distribution and out-of-distribution environment parameterizations. Moreover, it remains effective when trained on data that blends optimal and suboptimal behavior, combines strategies from its context (behavior-stitching), and dynamically adapts to non-stationary environments. These proof-of-concept findings suggest that in-context learning via reinforcement-tuned transformers may form the basis of a promising *general-purpose problem-solver*.

1 Introduction

Imagine a Mars mission in which a robot's appendage suddenly loses functionality. An adaptive agent capable of rapidly learning new behaviors could continue the mission successfully. While reinforcement learning (RL) has excelled in stationary environments (Sutton and Barto, 2018), real-world applications frequently demand quick adaptation to unexpected changes—something traditional RL struggles to achieve efficiently (Tsividis et al., 2017; Duan et al., 2016).

Recent advances leveraging transformer architectures demonstrate remarkable generalization capabilities through in-context learning (ICL), enabling rapid adaptation to novel tasks without retraining (Brown et al., 2020; Vaswani, 2017). Inspired by this, we explore whether transformers finetuned via reinforcement learning can adapt to changing conditions without additional training, a phenomenon we term **In-Context Reinforcement Learning (ICRL)**.



(a) Early inference (b) Mid inference (c) Late inference

Figure 1: ICRL-trained Llama 3.1 learns to solve an unseen Frozen Lake environment. The trajectories in early (a), mid (b), and late (c) interactions show solution refinement. Mistakes in early inference (e.g., falling into holes) disappear with experience in late inference.

In this work, we finetune Llama 3.1 (Dubey et al., 2024) on the grid-world game Frozen Lake (Farama Foundation, 2022) using a reinforcement learning objective (Mnih et al., 2013). We show that our model not only solves previously unseen maps from the same distribution (as seen in Figure 1) but also generalizes to out-of-distribution environments, robustly learns from varying data quality, dynamically adapts to non-stationary conditions, and effectively combines learned behaviors. These capabilities underscore the potential of reinforcement-tuned transformers as versatile, *general-purpose problem solvers* capable of human-like adaptability.

2 Background and Related Work

Recent efforts combining transformers with reinforcement learning (RL) have significantly advanced solving complex sequential decision-making tasks. The Decision Transformer (DT) reformulates RL as a sequence modeling problem by predicting future actions conditioned on past trajectories and desired returns, effectively leveraging transformer architectures for improved performance. However, DT relies heavily on high-quality, diverse training trajectories, limiting its applicabil-

Proceedings of the 1st Workshop for Research on Agent Language Models (REALM 2025), pages 401–410

ity to new or out-of-distribution scenarios (Chen et al., 2021).

To overcome the limitations of DT, Algorithm Distillation (AD) trains transformers to emulate RL algorithms themselves rather than directly modeling trajectory sequences. This allows AD to capture underlying patterns in algorithmic decision-making, enabling sample-efficient generalization. Yet, this approach requires that the problem first be solved with traditional RL, making it dependent on the hyper-parameters and quality of the algorithm it is imitating (Laskin et al., 2022).

The Decision-Pretrained Transformer (DPT) further addresses limitations of DT and AD by training transformers to directly imitate an action oracle, thus learning near-optimal policies without explicitly requiring future reward predictions. Despite its improved trajectory stitching capabilities, DPT's reliance on oracle-provided optimal actions during training restricts its practical application to scenarios where such optimal solutions are readily available (Lee et al., 2024).

Meta-RL approaches utilizing transformers have demonstrated promising capabilities for rapid task adaptation via in-context learning (ICL). These methods efficiently generalize learned behaviors to novel tasks based on minimal contextual experience (Melo, 2022; Bauer et al., 2023). However, their robustness to dynamically changing environments and resilience against imperfections in training data quality have not been extensively explored.

Building upon the foundational works that merge transformers with reinforcement learning, we focus on harnessing the potential of ICRL. While previous studies have established that transformers can be finetuned using reinforcement learning to solve novel problems, our work advances this integration by uncovering and demonstrating several novel advantages of ICRL that have not been previously explored. Specifically, we show:

- **In-Context Behavior Stitching**: ICRL-trained transformers can combine learned skills in novel ways to solve complex tasks. This ability indicates that the models have internalized principles akin to dynamic programming, allowing them to piece together previously acquired knowledge to tackle new challenges effectively.

- **Robustness to Suboptimal Training Data**: We find that ICRL reduces sensitivity to the presence of suboptimal trajectories in the training set. Transformers trained using ICRL can learn effectively even from failed episodes, exhibiting strong generalization abilities despite imperfections in the training data.

- **Adaptation to Non-Stationary Environments**: Our experiments show that ICRL-trained transformers maintain high performance in changing environments by dynamically adjusting to new information. They prioritize recent interactions over outdated data, enabling them to adapt quickly to non-stationary settings and maintain robust performance.

These findings suggest that ICRL offers significant advantages in developing versatile AI systems capable of human-like adaptability.

3 Methodology

To explore the capabilities of **ICRL**, we employ the open-source **large language model (LLM)** called *Llama 3.1 8B Instruct* (Dubey et al., 2024). We finetune this model using the **Deep Q-Network (DQN)** reinforcement learning algorithm (Mnih et al., 2013), which enables the model to learn optimal actions through trial and error.

Our training data are collected from the parametric game *Frozen Lake* (Farama Foundation, 2022), a dynamic environment where the game parameters can be changed between episodes. Rather than focusing on solving a single, specific version of Frozen Lake, our objective is to enhance the model's performance across multiple episodes with varying game configurations. By doing so, we aim to improve the model's ability to generalize and find better solutions over time, thus highlighting the benefits of the ICRL approach.

This section aims to provide a clear understanding of our experiments and results. We begin by explaining the general problem formulation for a **Partially Observable Markov Decision Process (POMDP)**. Then, we review how reinforcement learning is applied to solve a POMDP. Next, we demonstrate how reinforcement learning can be applied to a pre-trained transformer model. Finally, we document our environment setup and data collection procedures.

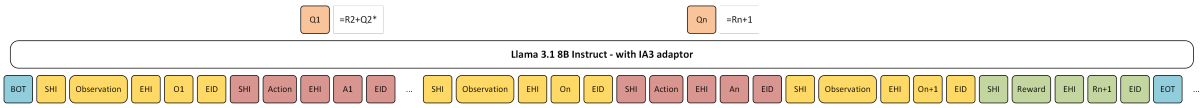


Figure 2: Fine-tuning Llama 3.1 8B Instruct with IA3 Adapters and a reinforcement learning objective. The model is fed sequences of states, actions, and (if nonzero) rewards, with every episode prefixed by the <|begin_of_text|> (BOT) token and terminated by the <|end_of_text|> (EOT) token. Tokens like <|start_header_id|> (SHI), <|end_header_id|> (EHI), and <|eot_id|> (EID) separate the *state*, *action*, and *reward*, mirroring how instruct models delineate *user* and *assistant* roles. The model predicts the Q-value of the current state for every action, updating the Q-values during training using the Bellman backup equation.

3.1 Partially Observable Markov Decision Process

A **Markov Decision Process (MDP)** is a mathematical framework used to model decision-making problems where an agent interacts with a process whose next state depends solely on the previous state and action. In a **Partially Observable Markov Decision Process (POMDP)**, the state is not fully observable. In such settings, the agent does not have direct access to the true state of the environment but must make decisions based on imperfect observations.

Formally, a POMDP is defined by the tuple $(S, A, T, R, \Omega, O, \gamma)$, where:

- S is a finite set of *states* representing all possible configurations of the environment.
- A is a finite set of *actions* available to the agent.
- $T(s' \mid s, a)$ is the *state transition probability*, the probability of transitioning to s' given action a in state s.
- $R(s, a)$ is the *reward function*, the immediate reward received after taking action a in state s.
- Ω is a finite set of agent perceivable *observations*.
- $O(o \mid s', a)$ is the *observation probability*, the probability of observing o after arriving at state s' and taking action a.
- $\gamma \in [0, 1)$ is the *discount factor* used to prioritize immediate rewards over future rewards.

Upon taking an action, the environment transitions to a new state s_{t+1} according to the transition probabilities T. The agent receives a reward r_{t+1} given by the reward function R and observes the next observation o_{t+1} based on the observation probabilities O.

We define the trajectory up to time t as:

$$\tau_t = \{o_0, r_0, a_0, o_1, r_1, a_1, \ldots, a_{t-1}, o_t, r_t\} \quad (1)$$

In practice, the agent does not have access to the true state s at any time. Instead, it maintains a belief about the probability distribution over possible states given the history τ_t.

3.2 Reinforcement Learning

Reinforcement learning involves an agent interacting with an environment to maximize cumulative rewards over time. The agent observes the environment, takes actions, and receives rewards based on those actions.

We define the *action-value function* (or *Q-function*) $Q^\pi(\tau, a)$ as the expected cumulative discounted reward obtained by taking action a given the history τ, and thereafter following a policy π. In deep-RL, the Q-function is normally a parameterized neural network denoted Q_θ or Q_ϕ where θ or ϕ are the network's parameters. Formally, the action-value function is defined as:

$$Q^\pi_\theta(\tau, a) = \mathbb{E}_{a\sim\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,\middle|\, \tau_t = \tau\right] \quad (2)$$

The agent's objective is to find an optimal *policy* π^* that specifies the best action to take based on the history, maximizing the expected cumulative discounted rewards. The optimal action-value function $Q^*(\tau, a)$ corresponds to the maximum expected return achievable from history τ by taking action a and thereafter following the optimal policy:

$$Q^*_\theta(\tau, a) = \max_\pi \; Q^\pi_\theta(\tau, a) \quad (3)$$

The optimal policy can be recovered for the optimal Q-function by taking the action that has the maximum value:

$$\pi^*_\theta(\tau_t) = \operatorname*{argmax}_a \; Q^*_\theta(\tau_t, a) \quad (4)$$

An important property of $Q^*_\theta(\tau, a)$ is that it satisfies a recursive relationship analogous to the *Bellman optimality equation* (Bellman, 1966):

$$Q^*_\theta(\tau_t, a_t) = \mathbb{E}\left[r_{t+1} + \gamma Q^*_\theta(\tau_{t+1}, a_{t+1})\right] \quad (5)$$

Reinforcement learning algorithms aim to estimate $Q^*_\theta(\tau, a)$ by iteratively applying this recursive relationship. A common approach is *value iteration*, where the action-value target y is calculated:

$$y(\tau_t, a_t) = \mathbb{E}\left[r_{t+1} + \gamma Q^*_\phi(\tau_{t+1}, \pi^*_\theta(\tau_{t+1}))\right] \quad (6)$$

The Q-network is trained by minimizing the distance to the target:

$$L = \mathbb{E}\left[\left(y(\tau_t, a_t) - Q_\theta(\tau_t, a_t)\right)^2\right] \quad (7)$$

To facilitate fast convergence, several techniques are typically employed. Gradient flow through the target is stopped so that the current Q-value converges to the target while the target is fixed. However, because the target depends on the Q-network's own predictions from the previous iteration, this creates a moving target scenario. This can be mitigated by keeping a delayed copy of the Q-network (i.e. Q_ϕ) from which we estimate the target and slowly update its parameters to follow the current Q-network (i.e. Q_θ) (van Hasselt et al., 2015). This process is called Polyak averaging

$$\phi_{t+1} = \alpha \times \theta_t + (1 - \alpha) \times \phi_t \quad (8)$$

and is controlled by a constant α.

Thus far, we have described the infinite horizon case. However, many games are episodic, so the action value is the expected sum of rewards until the game terminates, rather than extending to infinity. This is easily incorporated by defining the target function of the last action in a sequence to be equal to the reward alone.

By iteratively updating the Q-network parameters using optimization methods like stochastic gradient descent, the agent learns to approximate the optimal action-value function based on histories. This enables the agent to make informed decisions that maximize cumulative rewards, even in partially observable environments where the true state is not directly accessible.

In traditional reinforcement learning, only the last observation is provided to the network. However, when trying to induce the transformer to learn in-context, we provide the entire history of interactions. Thus, the transformer conditions its output on the whole trajectory τ. Only action tokens contribute to the training loss defined in Equation 7.

3.3 Transformer Network

We selected *Llama 3.1 8B Instruct* because it is a pre-trained transformer that has a demonstrated ability to perform ICL. We use an IA3 (Liu et al., 2022) adapter to decrease the computational load and memory requirements.

To train the network, we use a discount factor $\gamma = 0.9$ and scale the reward by multiplying it by 30. The delayed target adapter's weights are updated using Polyak averaging with a factor of $\alpha = 0.1$ (except when specified). Additionally, we use a learning rate of 1×10^{-2}, warmed from zero over the first ten batches, each consisting of 10 slices of data, with each slice being 4,096 tokens long.

During evaluation, as discussed in Section 4.6, deploying the transformer without forced exploration results in poor performance. Thus, for each evaluation trial, we use an epsilon-greedy-style warmup. In the first twenty episodes, epsilon (which represents the probability of using an action predicted by the transformer) is gradually decreased from 1 to 0, which corresponds to gradually increasing the probability of letting the transformer choose the next action. When the transformer does not get to choose the action, we randomly select it from a uniform distribution. After twenty episodes, we let epsilon remain 0, so the transformer always chooses the next action.

3.4 Environment

Since LLMs process discrete data, our chosen environment must have discrete state and action spaces, along with a broad parametric space to simulate non-stationarity. *Frozen Lake* meets these criteria, making it an ideal setting to showcase the effectiveness of ICRL.

We represent states numerically, corresponding to tile numbers, and actions using the words *up*, *down*, *left*, and *right*. This choice leverages Llama's existing understanding of these terms, enhancing adaptability. Each parameterization of Frozen Lake generates a unique map with randomized starting points, goal points, and holes. The model learns the environment through interaction without direct map visibility, striving to maximize rewards. Players receive a reward of 1.0 upon reaching the goal state, and 0.0 otherwise. Episodes terminate upon reaching the goal, falling into a hole, or after 100 steps.

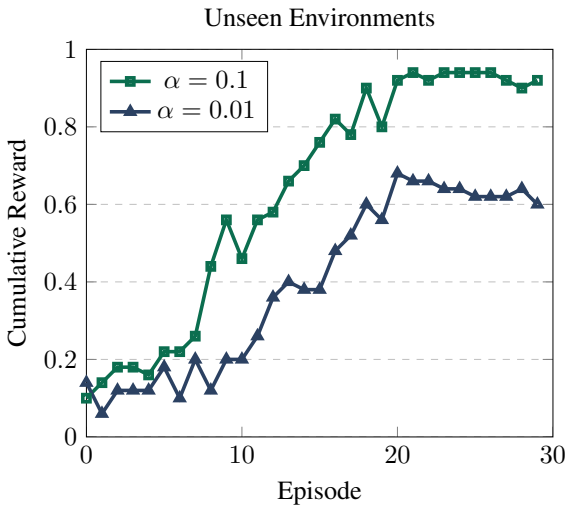


Figure 3: Mean cumulative reward over 50 trials as an ICRL-trained transformer improves its score on *unseen* environments. Maps (i.e. environment parametrization) have never been trained on but are chosen from the same distribution as training examples. Significant improvement can be observed as the agent demonstrates that it has learned to solve unseen maps. Also, $\alpha = 0.1$ significantly outperforms $\alpha = 0.01$.

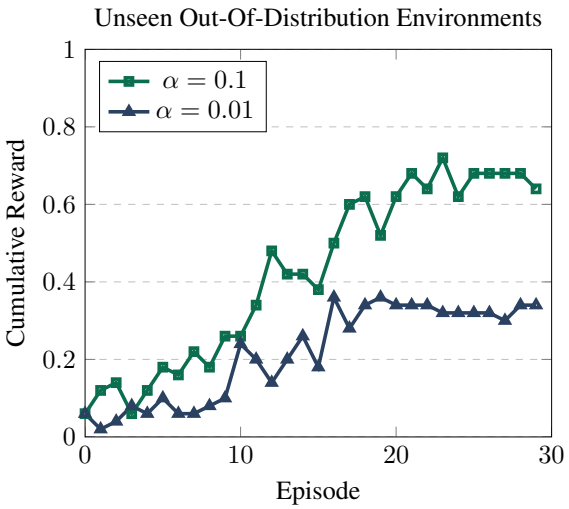


Figure 4: Mean cumulative reward over 50 trials as an ICRL-trained transformer improves its score on unseen and out-of-distribution environments. Generated maps are larger than anything ever seen during training. Improvement can be observed (though not as significant as in the in-distribution case) as the agent demonstrates that it has learned useful behaviors even for environments outside the distribution of its training data.

3.5 Data

Data are generated by training a traditional reinforcement learning algorithm on 250 different parameterizations of our chosen environment and collecting the data. It is extremely important to note that, unlike algorithmic distillation, we randomly mix episodes so that there is no inherent order to the data.

Our data are formatted in conversational form. Unlike the Llama instruct template, instead of the roles *user* and *assistant*, we use the roles of *action*, *observation*, and *reward* as seen in Figure 2.

We concatenate episodes of data together. Every 20 to 40 episodes, the environment parameterization (i.e., map in Frozen Lake) is changed so that the network can practice adapting to non-stationary environments. We call the 20 to 40 episodes with the same parameterization a *set*. Approximately 2 to 3 sets are combined together until 4,096 tokens are reached.

4 Experiments

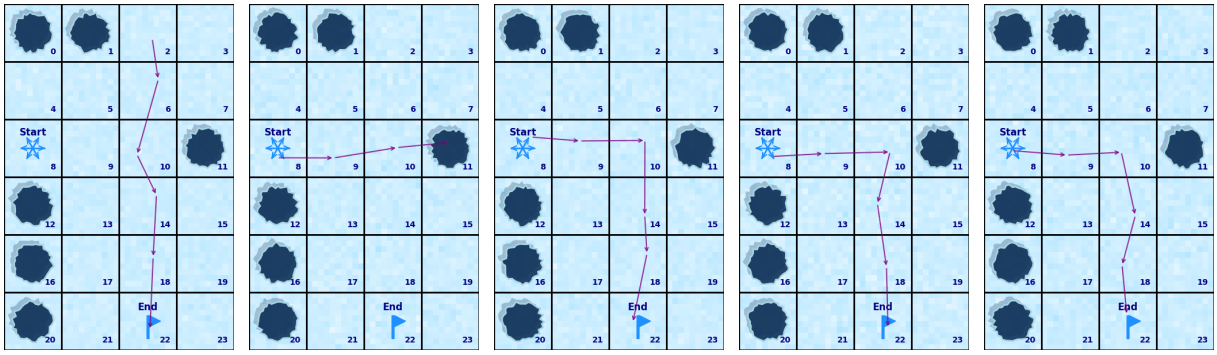
Having finetuned Llama 3.1 with the generated data, in this section, we evaluate its performance across a variety of tasks to demonstrate its capabilities as a general-purpose problem solver. We examine its ability to solve both in-distribution and out-of-distribution examples, its capacity for in-context behavior stitching, its robustness to low-quality training data, and its adaptability to non-stationary environments. We also discuss the challenges associated with exploration in ICRL settings.

4.1 Solving Unseen In-Distribution Examples

To assess the transformer's ability to generalize to unseen but in-distribution examples, we evaluated its performance on new parameterizations of the Frozen Lake environment that were not included in the training set but were generated from the same distribution.

Setup: We generated 50 new Frozen Lake maps with widths and heights ranging from 3 to 5 tiles and randomly placed holes throughout the map, similar to the training data. The agent was not provided with any explicit map information and had to learn the optimal path solely through interaction with the environment. Each evaluation consisted of multiple episodes, allowing the transformer to learn and improve its policy through ICRL. We also examine the Polyak averaging constant's effect by testing both $\alpha = 0.1$ and 0.01.

Results: As shown in Figure 3, the transformer effectively learned to navigate the new environments. During early episodes, the agent fell into holes, only reaching the goal 10% of the time, but as it gained more experience, it achieved a 90%



(a) Example 1 (b) Example 2 (c) Trial 1 (d) Trial 2 (e) Trial 3

Figure 5: Example of how ICRL combines different experiences to generate improved solutions. Subfigures (a) and (b) show two example trajectories provided as context to the ICRL transformer. The inference trials in subfigures (c), (d), and (e) display the paths predicted by the transformer, which leverages information from both examples to develop an optimal solution.

win rate (when $\alpha = 0.1$). This demonstrates that ICRL can successfully generalize to new, unseen maps within the same distribution, improving its performance over time without additional weight updates. We also notice a significant dependence on the Polyak constant, which suggests that allowing the target network to update quickly (i.e. $\alpha = 0.1$) outweighs the benefits of increased stabilization (i.e. $\alpha = 0.01$).

Critical Observation: As we explored specific failure examples, a pattern became clear: in the large majority of cases, the reason that the transformer fails to find the goal is that it fails to explore the whole map. We noticed that the agent almost always avoids holes but would get caught in loops where it traversed the same path repeatedly until it exceeded the number of allowable steps in Frozen Lake. We discuss this more in Section 4.6.

4.2 Solving Out-of-Distribution Examples

To evaluate the model's ability to generalize beyond the training distribution, we tested it on Frozen Lake maps with configurations not encountered during training.

Setup: We created 50 out-of-distribution environments by using larger map sizes (e.g., widths and heights of 6 to 7 tiles). These maps are not only larger; they are also much harder, as a longer sequence of actions must be learned to reach the goal state. As before, the transformer had to learn to navigate these new and more complex environments solely through interaction. We also ablate the Polyak averaging constant by setting α equal to 0.1 and 0.01.

Results: The transformer showed remarkable adaptability to these out-of-distribution environments. While performance was initially lower compared to in-distribution tests, the agent was able to achieve limited success and, most importantly, demonstrated improvement over time (see Figure 4). This indicates that the model can transfer its learning to novel scenarios, suggesting that the meta-learning ability acquired in a restricted domain can generalize to unseen environments. Just as in the in-distribution case, $\alpha = 0.1$ significantly outperforms $\alpha = 0.01$. For the rest of our experiments we choose $\alpha = 0.1$.

4.3 In-Context Behavior Stitching

Humans can compose solutions from individual experiences and acquire expertise in a piecemeal manner (Langley, 2022). This permits significantly more efficient usage of experiential information. One of the key advantages of the ICRL-trained transformer is its ability to likewise combine experiences in novel ways to solve complex tasks, a phenomenon we refer to as **in-context behavior stitching**.

Setup: We generated ten grid-world maps, each containing two paths that cross with only one leading to a goal. The agent needs to walk part of one path and then switch to the other to get a reward, requiring it to combine multiple paths (i.e. skills) to solve the game.

Results: On all ten maps (100% of trials), the transformer reached the goal by splicing together the relevant segments of its prior experiences. Figure 5 shows sample trajectories: the model assembles novel action sequences—never seen together in the same episode—and achieves near-optimal

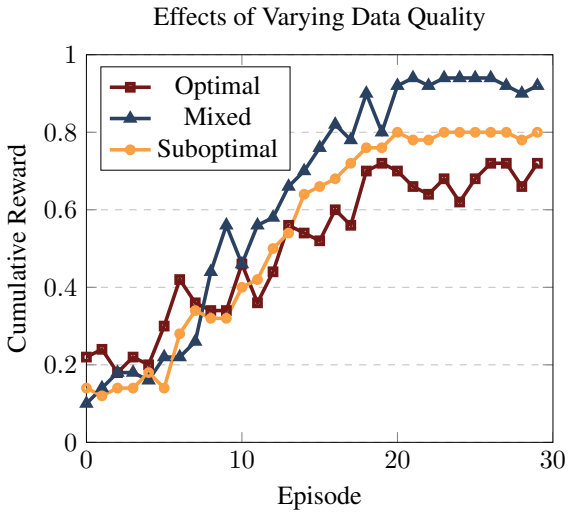


Figure 6: Unlike imitation learning, ICRL is largely impervious to data quality. Here we plot the average return achieved over 50 runs when the network is RL-trained on optimal data, suboptimal data, and a mixture of both. This graph shows that a mixture optimal and suboptimal data actually achieves the highest average cumulative reward.

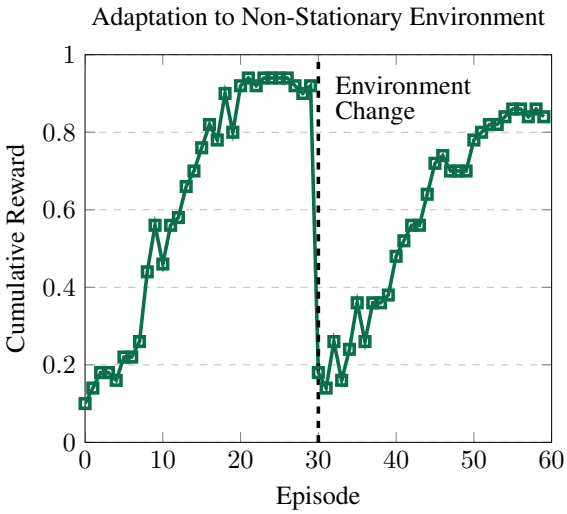


Figure 7: Even when the environment changes, an ICRL-trained transformer can detect and adapt without any explicit signal of the change. The plot shows the average over 50 trials. At episode 30 the environment is changed without any warning to the agent. The agent detects this change and increases its reward through ICRL.

behavior consistent with dynamic programming.

This ability to assemble experiences suggests that ICL acquires expertise in a piece-meal manner, an open question regarding the relationship of ICL to human-like learning (Roberts, 2024).

4.4 Robustness to Suboptimal Training Data

To understand the model's ability to learn from low-quality data with suboptimal actions, we investigated performance when trained on data having varying average cumulative returns.

Setup: We created several training datasets with different levels of reward:

- **Optimal data:** Trajectories from episodes that achieved mostly high rewards. We sampled our data, making it five times more likely to select a successful episode that reached the goal than an unsuccessful episode.
- **Mixed data:** A combination of high-reward and low-reward trajectories. No sampling weight was given to either successful or unsuccessful episodes.
- **Suboptimal data:** Trajectories from episodes that mostly ended with low rewards. We sampled our data, making it five times more likely to select an episode that did not reach the goal than a successful episode.

We then trained separate instances of the transformer on each dataset and evaluated their performance on unseen Frozen Lake environments.

Results: One might expect the transformer's performance to degrade proportionally with the ratio of high-reward to low-reward data as in pure imitation learning (Ghosh et al., 2024) - a form of supervised fine-tuning. However, as shown in Figure 6, our results indicate that varying the quality of the training data had a non-linear impact on the final cumulative reward. Remarkably, the transformer learned effective policies even when trained on suboptimal data, which consisted largely of unsuccessful episodes. In fact, training on optimal data slightly reduced performance compared to mixed data. We hypothesize that ICRL benefits from exposure to a diverse range of experiences, including both successful and unsuccessful trajectories. Our findings demonstrate that the transformer's performance is not proportional to the quality of the training data, allowing it to learn effectively without the need for extensive data curation or filtering.

4.5 Adaptation to Non-Stationary Environments

We tested the transformer's ability to adapt to environments that change over time, reflecting non-stationary conditions that can occur in real-world scenarios.

Setup: We presented the transformer with se-

quences of environments where the map configuration changed after 30 episodes. The agent was allowed to experiment in the new environment, but was not informed of the change. The changes included alterations in the position of holes, the size of the map, and the start and goal locations. This was repeated 50 times, and the average was plotted in Figure 7.

Results: The transformer adapted to the changing environments. It prioritized recent interactions in its decision-making process, effectively disregarding outdated information from previous environments. In Figure 7, the agent's performance drops when the environment changes and then recovers, rising to almost the same level as previously. The agent learned to adjust its policy based on new information! This adaptability is a key advantage of the ICRL approach, enabling operation in non-stationary environments.

4.6 The Challenge of Exploration

Despite the promising performance of the transformer model, we observed challenges related to *exploration*. Specifically, during evaluation, if the transformer is not encouraged to take random actions at the beginning of each new episode, it tends to settle into suboptimal trajectories. Even when we enforce exploration, many failures occur because the model has never seen an example of reaching the goal before.

We believe that part of the problem is related to the distributional shift between offline training and online evaluation (Levine et al., 2020). During offline training, the model is provided with a random mixture of successful and unsuccessful episodes. However, at the start of online evaluation, there is a very high proportion of unsuccessful episodes.

The following solutions may address these challenges:

1. **Online Training**: Train the model in an online manner, allowing it to experience low-reward trajectories initially and adapt over time.

2. **Model-Based Reinforcement Learning (MBRL)**: Train the model to predict tokens in the environment and roll out experiences based on the chosen actions, effectively simulating online learning.

3. **Cross-Episode Reward Function**: Train the network with a reward function where the reward an action receives is based on the expected value of future rewards in *future episodes*. This approach could potentially reward the model for exploration, even if an action does not contribute to attaining the goal in the current episode.

5 Conclusion

In this study, we have demonstrated that fine-tuning Llama 3.1 with reinforcement learning enables it to function as a general-purpose problem solver. Within the Frozen Lake environment, it is capable of adapting to situations it has never encountered before. While in-context reinforcement learning (ICRL) may not always find the perfect solution, it can *enhance performance* in unforeseen scenarios. This progress indicates that agents capable of human-like adaptability and continuous improvement are within reach.

Limitations

This study explores the potential of reinforcement learning (RL)-fine-tuned transformer models, specifically focusing on their adaptability through in-context reinforcement learning (ICRL). While our results demonstrate promising capabilities, several limitations should be noted:

First, our experiments are conducted exclusively within the simplified grid-world environment, Frozen Lake. Although this setup effectively demonstrates the capabilities of ICRL, the complexity and uncertainty inherent to real-world environments may present significantly greater challenges. Therefore, the generalization of these findings to more complex scenarios, such as continuous state-action spaces or high-dimensional observations, remains uncertain.

Second, the exploration challenge identified during our evaluations reveals a critical limitation regarding the initial policy performance when exposed to new or dramatically changed environments. As noted, without enforced exploration strategies, the model tends to settle into suboptimal trajectories. While we proposed several theoretical solutions (e.g., online training, model-based RL), the practical implementation and effectiveness of these methods remain untested within the scope of this paper.

Third, despite demonstrating robustness to low-quality training data within our experimental conditions, our findings may not universally generalize

to all forms of noisy or biased data, particularly in larger or more complex environments.

Finally, computational constraints limited our experiments to relatively small maps and short trajectories. Scaling this approach to significantly larger models or datasets may introduce unforeseen challenges in training stability, computational cost, and inference speed.

Future research addressing these limitations is necessary to further validate the efficacy and generalizability of reinforcement fine-tuned transformers as general-purpose problem solvers.

Ethical Considerations

Extensive analyses have shown that risks to humans escalate as systems become more autonomous; essentially, when users surrender greater control, the potential dangers from the system increase (Mitchell et al., 2025). These risks are further amplified by adaptable agents that, in theory, can learn to solve novel, unforeseen problems autonomously. Consequently, we propose that agents should not be deployed in unconstrained environments at this time. Instead, we recommend that agents be confined within controlled "sandbox" environments where they are unable to affect the outside world, allowing for rigorous testing and validation of their behaviors.

Some argue that, because of these risks, we should cease the development of autonomous agents (AAs) altogether. However, we contend that the advancement of autonomous agents is both inevitable and essential due to the substantial benefits associated with their capabilities. It is in humanity's best interest to pursue the responsible development of autonomous technologies before malevolent actors possibly exploit them. This approach is further justified by the strong likelihood that the most effective defense against harmful AAs will be the deployment of benevolent AAs designed to counteract malicious activities.

References

Jakob Bauer, Kate Baumli, Feryal Behbahani, Avishkar Bhoopchand, Nathalie Bradley-Schmieg, Michael Chang, Natalie Clay, Adrian Collister, Vibhavari Dasagi, Lucy Gonzalez, and 1 others. 2023. Human-timescale adaptation in an open-ended task space. In *International Conference on Machine Learning*, pages 1887–1935. PMLR.

Richard Bellman. 1966. Dynamic programming. *science*, 153(3731):34–37.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. 2021. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097.

Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. 2016. Rl^2: Fast reinforcement learning via slow reinforcement learning. *Preprint*, arXiv:1611.02779.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Farama Foundation. 2022. Gymnasium documentation. `https://gymnasium.farama.org`.

Udita Ghosh, Dripta S Raychaudhuri, Jiachen Li, Konstantinos Karydis, and Amit K Roy-Chowdhury. 2024. Robust offline imitation learning from diverse auxiliary data. *arXiv preprint arXiv:2410.03626*.

Pat Langley. 2022. The computational gauntlet of human-like learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(11):12268–12273.

Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steigerwald, DJ Strouse, Steven Hansen, Angelos Filos, Ethan Brooks, and 1 others. 2022. In-context reinforcement learning with algorithm distillation. *arXiv preprint arXiv:2210.14215*.

Jonathan Lee, Annie Xie, Aldo Pacchiano, Yash Chandak, Chelsea Finn, Ofir Nachum, and Emma Brunskill. 2024. Supervised pretraining can learn in-context reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. 2020. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *Preprint*, arXiv:2005.01643.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Preprint*, arXiv:2205.05638.

Luckeciano C Melo. 2022. Transformers are meta-reinforcement learners. In *international conference on machine learning*, pages 15340–15359. PMLR.

Margaret Mitchell, Avijit Ghosh, Sasha Luccioni, and Giada Pistilli. 2025. Ai agents are here. what now? https://https://huggingface.co/blog/ethics-soc-7. Hugging Face Blog, published on January 13, 2025. Accessed on January 18, 2025.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *Preprint*, arXiv:1312.5602.

Jesse Roberts. 2024. Do large language models learn to human-like learn? In *Proceedings of the AAAI Symposium Series*, volume 3, pages 588–591.

Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

Pedro A Tsividis, Thomas Pouncy, Jaqueline L Xu, Joshua B Tenenbaum, and Samuel J Gershman. 2017. Human learning in atari. In *2017 AAAI spring symposium series*.

Hado van Hasselt, Arthur Guez, and David Silver. 2015. Deep reinforcement learning with double q-learning. *Preprint*, arXiv:1509.06461.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.