Extended Abstract

Motivation ARC-AGI-3 is an interactive benchmark where agents must discover unknown game mechanics through exploration under a strict action budget. One of the current state-of-the-art training-free approaches (Rudakov et al., 2025) tracks visited states in a directed graph and uses a hand-coded heuristic to assign each visual segment to one of five priority groups, which determines which parts of the screen to click first. This heuristic cannot adapt: a segment type that proves useless in level 1 receives the same priority in level 8. We ask whether replacing this fixed priority function with a learned one improves exploration efficiency across the progressively harder levels of ARC-AGI-3.

Method We implement five learned priority agents on top of the Rudakov et al. graph explorer, all trained online from a dense ± 1 binary reward signal (whether clicking a segment caused a visual state transition). Each agent ranks segments into five priority groups using a learned scoring function; the graph explorer's traversal logic is otherwise untouched. We also redesign the feature representation for each segment, replacing the three binary features of the original paper with four features, three of which are continuous, that better allow the model to discover gradients within categories. Additionally, all learned agents except the MLP ablation are warm-started from a weight configuration calibrated to approximately recover the heuristic's priority ordering.

Implementation The five agents span a range of online learning approaches: (1) REINFORCE (vanilla policy gradient) for simplicity and interpretability; (2) First-Order MAML (FOMAML), which meta-learns an initialization that adapts quickly at level boundaries, exploiting the fact that different levels reward different segment types; (3) a two-layer MLP trained by online backpropagation as a nonlinear supervised baseline; (4) Soft Actor-Critic (SAC), for off-policy sample efficiency via a replay buffer with twin critics and entropy regularization; and (5) Proximal Policy Optimization (PPO), for stable on-policy updates with a clipped surrogate objective and GAE-λ advantage estimates.

Results In 45-minute evaluations on ARC-AGI's vc33 clicking game, FOMAML achieves a mean score of 4.42 vs. the baseline's 0.59 mean, a 7.5× improvement, while REINFORCE reaches a mean of 2.82 and SAC 2.41. PPO (0.54) and the neural network (0.04) underperform the baseline in the time window, which we attribute to insufficient data for their respective sample requirements.

Discussion The dominance of FOMAML suggests that the primary bottleneck in `vc33` is fast per-level adaptation rather than globally-optimal policy quality. Different levels introduce distinct interactive elements, so an initialization trained for quick adaptation consistently outperforms a single accumulated weight vector. The gap between REINFORCE and the baseline supports the core hypothesis that soft continuous weights over a richer feature space can improve on fixed binary rules even with the simplest gradient update. The underperformance of PPO and the MLP illustrates that models with more parameters and richer objectives require more data before they outperform a warm-started linear baseline, and online RL in sparse environments can push these models toward bad local optima if they cannot accumulate enough signal early.

Conclusion Replacing a hand-coded priority heuristic with a learned priority function, particularly one that meta-learns across levels, substantially improves exploration efficiency on `vc33`. The result suggests that the level-to-level distribution shift in ARC-AGI-3 is the dominant challenge, and that fast adaptation mechanisms are better matched to this environment than globally-optimized policies trained over the full game.

Learning Priority Functions for Graph-Based Exploration in ARC-AGI-3

Ryan Bookman
Department of Computer Science
Stanford University
rbookman@stanford.edu

Steve Mendeleev
Department of Computer Science
Stanford University
steveroy@stanford.edu

Kyle Feinstein
Department of Computer Science
Stanford University
kfeinst@stanford.edu

Abstract

We extend the state-of-the-art graph-based exploration approach for ARC-AGI-3 (Rudakov et al., 2025) by replacing its hand-coded visual salience heuristic with five learned priority functions trained online from a dense binary transition reward. The heuristic assigns each visual segment to one of five priority groups using fixed binary rules that cannot adapt as levels progress. We redesign the segment feature representation from three binary indicators to four continuous features: HSV color saturation, an exponential closeness-to-medium-size score, status-bar membership, and a log-count of twin segments. We warm-start all learned agents except the MLP ablation to approximately recover the heuristic's ordering before any training begins. We compare five algorithms: REINFORCE, First-Order MAML (FOMAML), a two-layer MLP, Soft Actor-Critic (SAC), and Proximal Policy Optimization (PPO). In 45-minute evaluations on vc33, FOMAML achieves a 7.5x improvement in score over the baseline, REINFORCE achieves a 4.8x improvement, and SAC achieves a 4.1x improvement. PPO and the MLP underperform the baseline in this time window due to insufficient data for their sample requirements. These results suggest that fast per-level meta-adaptation is the most effective inductive bias among the methods we tested for `vc33`, and that learned continuous feature representations can outperform hard-coded binary ones even with simple linear models.

1 Introduction

ARC-AGI-3 ARC Prize Foundation (2026) represents a fundamental departure from prior ARC benchmarks. It presents game-like interactive environments where agents must discover unknown mechanics through exploration, forming hypotheses about what to click, testing them, and revising their understanding as levels progressively introduce new unknown elements. The benchmark penalizes both failure to solve levels and inefficiency in doing so, creating strong pressure for agents that find the right actions quickly.

Rudakov et al. (2025) establish a strong training-free baseline through systematic graph-based exploration. Their agent maintains a directed graph over visited states and action transitions, prioritizes actions using three binary visual salience features, and navigates toward unexplored frontier states using shortest-path distances. This structured approach substantially outperforms both random exploration and LLM-based baselines. However, the method is entirely static. Within each priority

* Equal contribution from all authors

Figure 1: Tasks `vc33`, `r11l`, `lp85` from public benchmark (ARC Prize Foundation)

group, action selection is uniform at random, and the priority group assignments are determined by a heuristic that never updates. This means the agent wastes identical effort re-testing segment types that proved useless in level 1 when it encounters them again in level 8.

Our contribution is replacing this fixed priority function with a family of learned alternatives, while keeping the graph explorer's traversal logic entirely intact. Our hypothesis is that priority group assignment can be framed as an online contextual-bandit problem with a dense reward signal. For each segment clicked, the agent immediately observes whether that click caused a state transition (+1) or not (-1). This allows us to train policies using a per-click signal that is much more dense than the environment's extremely sparse level-completion reward.

We make three concrete contributions. First, we redesign the segment feature representation by replacing three binary indicators with four continuous features that preserve the heuristic's intended structure while permitting soft gradients. Second, we warm-start all learned agents except the MLP ablation to approximately recover the heuristic's priority ordering, ensuring competent behavior from the first step. Third, we implement and compare five online learning algorithms spanning policy gradient, meta-learning, supervised MLP, off-policy actor-critic, and on-policy clipped surrogate methods.

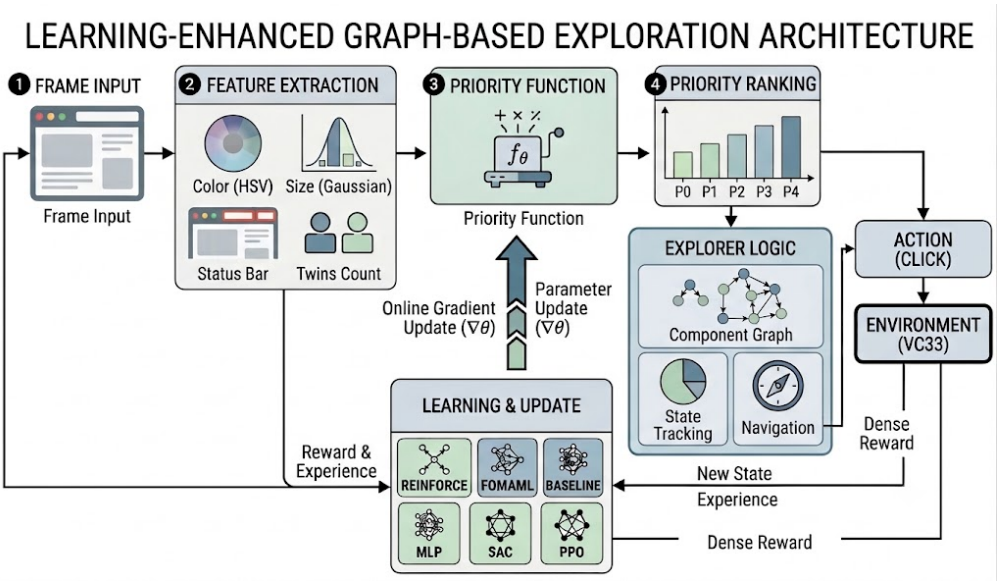


Figure 2: Visualization of our methods.

Our main finding is that FOMAML, which meta-learns an initialization from which per-level weights can adapt quickly, achieves the largest gains, followed by vanilla REINFORCE and SAC. This pattern

reveals that the dominant challenge in ARC-AGI-3 is not learning a globally optimal priority function across all levels, but adapting quickly to the specific interactive elements each level introduces.

2 Related Work

ARC-AGI-3 ARC Prize Foundation (2026) is a benchmark designed to measure general fluid intelligence, emphasizing adaptability to novel tasks. Agents are placed in unfamiliar turn-based games with no instructions and must discover the rules through interaction, under a strict action budget. Crucially, the scoring rewards not just solving levels but solving them efficiently, which makes the order in which an agent tests actions a first-order determinant of performance.

The most directly relevant prior work is Rudakov et al. Rudakov et al. (2025), one of the strongest ARC-AGI-3 submissions, which serves as the baseline we build on. Their training-free agent maintains a directed graph of visited states and tested transitions, navigating toward unexplored frontiers via shortest paths. To make the large action space tractable, it ranks segments into five priority tiers using a fixed hand-coded heuristic over color, size, and morphology. This graph-based structure is motivated by Go-Explore Ecoffet et al. (2021), which shows that explicitly tracking visited states and reliably returning to frontier states outperforms memoryless search under sparse feedback. We keep this structure entirely intact and replace only the fixed priority heuristic with learned alternatives.

The core challenge is that ARC-AGI-3 levels shift over time, introducing new interactive elements as the agent progresses, so a fixed priority rule that works early on becomes stale. This motivates replacing the heuristic with a learned priority function trained online from per-click transition feedback. Alet et al. Alet et al. (2020) show that exploration strategies can be meta-learned across tasks so the agent adapts quickly when the environment changes, rather than relying on a fixed hand-designed bonus. SIERL Sotirchos et al. (2026) demonstrates more concretely that learning which frontier states to prioritize outperforms fixed heuristics in sparse-reward exploration. Neither work is applied to ARC-AGI-3, but both motivate the approach we take. We evaluate a range of online learning algorithms as priority-function learners, from simple policy gradient methods to off-policy actor-critic approaches, with FOMAML as the conceptual center given the level-shifting structure of the benchmark.

3 Method

3.1 Baseline: Heuristic Priority Assignment

The Rudakov et al. baseline computes three binary features per visual segment and uses hard-coded thresholds to assign each segment to one of five priority tiers:

Table 1: Baseline Heuristic Feature Rules

Feature	Rule
`is_salient`	1 if the color is white, gray, or black; else 0
`is_medium_width`	1 if bounding box spans 2–32 px in both x and y; else 0
`is_status_bar`	1 if color index represents the masked status bar element; else 0

Segments with `is_status_bar` $= 1$ are assigned to the lowest priority group. Among the remaining segments, those matching both `is_salient` and `is_medium_width` receive the highest priority. These rules encode the authors' intuitions that medium-sized, colorful elements are the most likely to be interactive buttons. However, crucially, this priority assignment remains identical at every step of every level and cannot adapt in response to agent experience.

3.2 Problem Reformulation: Learning Priority Functions

We reframe priority assignment as an online contextual-bandit problem. At each step, the agent selects a visual segment based on its current priority function, executes a click action, and immediately observes a dense binary reward: $r = +1$ if the frame transitioned to a new state, and $r = -1$

otherwise. This per-click signal allows online gradient updates after every single action, avoiding the extremely sparse level-completion signal that traditional end-to-end reinforcement learning approaches face with these games.

All five priority-learning agents share a common architectural backbone. A parameterized scoring function $f_\theta(\phi)$ maps a segment's 4-dimensional feature vector ϕ to a scalar score. Visual segments are ranked by this score and partitioned evenly into five priority groups. The downstream graph explorer then selects from the highest available priority group using its existing navigation logic (the same navigation logic that Rudakov et al. presents). We update the parameters θ online after each clicked segment using the observed binary reward.

Our design leaves the graph explorer's traversal strategy and state tracking completely unchanged, ensuring that any performance gains are isolated and directly attributable to priority learning.

3.3 Feature Engineering

We replace the three binary features of the baseline with four features, three of which are continuous. The motivation for this shift is twofold. First, binary features discard critical within-category information: all chromatic colors are treated identically by `is_salient`, yet a highly saturated red segment and a muted pale-green segment carry vastly different interactive probabilities. Second, linear models over binary features can only express strict threshold decisions, whereas continuous features allow gradients to express graded preferences.

Feature 1: HSV Color Saturation (replaces `is_salient`). We compute saturation directly from the RGB values:

$$\text{sat} = \frac{\max(R, G, B) - \min(R, G, B)}{\max(R, G, B)} \tag{1}$$

with saturation defined as 0 when $\max(R, G, B) = 0$.

While the original binary feature treats all chromatic colors identically, saturation preserves the salient/non-salient distinction while exposing a continuous gradient within the chromatic range. This, in contrast with a one-hot vector representing color, allows a model to learn that highly saturated game elements are more likely to be interactive than muted chromatic background elements.

Feature 2: Closeness-to-Medium (replaces `is_medium_width`). The binary `is_medium_width` feature treats segment size as a hard classification (either medium or not), discarding all information about how far a segment deviates from the ideal scale. A continuous alternative like log(area) partially addresses this but can only encode a monotone preference ("bigger is better" or "smaller is better"), failing to capture the inverted-U shape the heuristic intends. Instead, we want a score that peaks at the ideal medium size and decays gracefully as segments grow too small or too large. We therefore compute an exponential decay response in log-area space:

$$\text{closeness} = \exp\left(-\frac{|\log(1 + \text{area}) - \log(1 + 64)|}{2.5}\right) \tag{2}$$

This yields a score of 1.0 at the ideal scale and decreases smoothly toward 0 for segments that deviate in either direction, so a segment *near* the medium range still receives a high score, while one far outside it scores near zero. The center at area $= 64$ px^2 is the geometric mean of the 2–32 px medium range in each dimension: $(\sqrt{2 \times 32})^2 = 64$. We chose the decay parameter $\sigma = 2.5$ so the half-response point falls approximately at the boundaries of the medium range.

Feature 3: `is_status_bar` (unchanged). Status-bar membership remains binary and the existing rule-based detection is highly reliable, meaning a continuous extension is not necessary.

Feature 4: $\log(1 +$ **number_of_twins**$)$ (new feature). Segments with many twins (other segments sharing the identical area, rectangularity, and color) tend to represent repeating UI decoration patterns, such as life-dot indicators or score grids, rather than unique interactive objects. We apply a log transform rather than passing the raw count because the structural distinction between 0 and 1 twins (unique vs. non-unique) is far more informative than the distinction between 10 and 11 twins, and the log scaling keeps the feature magnitude inline with the others to prevent gradient dominance.

3.4 Warm-Start Initialization

We initialize all learned agents except the MLP with weights calibrated to approximately recover the baseline heuristic's priority ordering before any online learning occurs.
For the linear models (REINFORCE and FOMAML), we use the following values for warm-start:

- **Saturation: 0.2.** A positive weight reflects the heuristic's preference for chromatic segments.
- **Closeness: 0.8.** A large positive weight reflects the heuristic's strong preference for medium-sized visual segments.
- **Status bar: -1.0.** A large negative weight heavily deprioritizes status bar elements, matching the heuristic's lowest-priority placement.
- **Log twins: -0.2.** A weak negative weight encodes the prior that high-twin segments are lower priority, allowing the model to smoothly strengthen or override this belief via experience.

Without warm-starting, agents relying on random initialization waste a prohibitive number of early environment steps executing arbitrary priority choices. Warm starting allows our models to converge to good results quickly.

3.5 Learned Agents

All five agents share the same input-output interface, receiving the 4-dimensional feature vector for each segment in the current frame and producing a priority score, which the graph explorer uses to rank segments into five groups. What differs across agents is how the scoring function is parameterized and how it updates from the observed ± 1 reward.

REINFORCE (Vanilla Policy Gradient) REINFORCE is the simplest of our learned agents and serves as an interpretable baseline for the others. It uses a linear scoring function with a sigmoid output: $\text{score}_i = \sigma(\mathbf{w} \cdot \phi_i)$. After each click, the weights take a reward-weighted policy-gradient update on the segment's sigmoid score:

$$\Delta \mathbf{w} = \alpha \cdot r \cdot \phi \cdot \sigma(\mathbf{w} \cdot \phi)(1 - \sigma(\mathbf{w} \cdot \phi)) \tag{3}$$

with learning rate $\alpha = 0.05$. A single weight vector persists across all levels, building up experience about which feature patterns correlate with transitions. REINFORCE is the smallest possible extension of the warm-started heuristic that still allows online adaptation.

First-Order MAML (FOMAML) While REINFORCE accumulates a single shared policy across all levels simultaneously, FOMAML separates two levels of learning. Meta-weights θ capture what is generally useful across levels and update only at level boundaries. Task-weights ϕ capture what is useful for the current level and update after each click.

At the start of each level, task-weights reset to the meta-weights: $\phi \leftarrow \theta$. Within a level, updates are identical to REINFORCE. At the end of each level, the meta-weights take an outer gradient step using the accumulated transition history $\mathcal{T}$:

$$\theta \leftarrow \theta + \eta_{\text{meta}} \cdot \frac{1}{|\mathcal{T}|} \sum_{(\phi_i, r_i) \in \mathcal{T}} r_i \cdot \phi_i \cdot \sigma(\theta \cdot \phi_i)(1 - \sigma(\theta \cdot \phi_i)) \tag{4}$$

We chose FOMAML because we hypothesized that ARC-AGI-3's progressive level structure makes per-level adaptation more valuable than global convergence. The agent should be able to identify a new level's interactive elements within its first few clicks, rather than after accumulating experience across many levels.

Neural Network (MLP) Where REINFORCE and FOMAML are linear in the features, the MLP can capture complex feature interactions, for example, whether a segment is both highly saturated and small, which no individual feature encodes on its own. The MLP is intentionally not warm-started, with weights sampled from $\mathcal{N}(0, 0.01)$. This acts as an ablation study that isolates the contribution of initialization, allowing us to evaluate whether nonlinear expressivity can compensate for a weaker starting point.

Soft Actor-Critic (SAC) Because REINFORCE updates from a single observation at a time, its weight estimates can be noisy when transitions are sparse. SAC addresses this by storing each observed (feature, reward) pair in a replay buffer of capacity 1,000 and sampling mini-batches to update twin Q-critics alongside an actor network ($4 \to 8$ ReLU $\to 1$ sigmoid).

Twin critics reduce Q-value overestimation by taking $\min(Q_1, Q_2)$ when computing actor targets. An entropy bonus with a learned temperature prevents the policy from collapsing into repeatedly clicking the same segment; this is tuned online to match a target entropy of $\mathcal{H}[\text{Bernoulli}(0.5)]$. Reusing each observation across multiple gradient steps makes SAC significantly more sample-efficient than the per-step updates of REINFORCE.

Proximal Policy Optimization (PPO) PPO introduces two distinct mechanisms for training stability: a critic network that provides a value baseline for variance reduction, and a clipped surrogate objective that limits how much the policy can change in a single update. The actor ($4 \to 8$ ReLU $\to 1$ sigmoid) and critic ($4 \to 8$ ReLU $\to 1$ linear) update from a rollout buffer of size 32 using the clipped surrogate objective:

$$\mathcal{L}_{\text{clip}}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] \tag{5}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ and $\hat{A}_t$ is the GAE-λ advantage estimate evaluated with $\gamma = 0.99$ and $\lambda = 0.95$. After each buffer flush, four epochs of updates run on random mini-batches of size 8, alongside an entropy bonus coefficient ($c_e = 0.01$) to preserve exploration. The clipping function prevents destructively large policy updates when individual observations are highly noisy, at the cost of slower early adaptation compared to REINFORCE.

4 Experimental Setup

4.1 Environment

We evaluate our approach on `vc33`, a click-based game from the ARC-AGI-3 public dataset. The `vc33` task uses explicit spatial interaction, in that the agent selects and clicks exact pixel coordinates, yielding an action space of size $64 \times 64 = 4{,}096$. The game consists of multiple levels of increasing difficulty, with each progressive level introducing entirely new visual elements and mechanics. All agents interact with the environment via ARC-AGI-3's API.

4.2 Parallelized Execution Infrastructure

All six agents (the static heuristic baseline plus our five learned agents) run simultaneously in separate execution threads, each managing its own completely independent game session. Running all agents in parallel ensures a fair comparison under identical system loads and strict time budgets.

4.3 Evaluation Metric

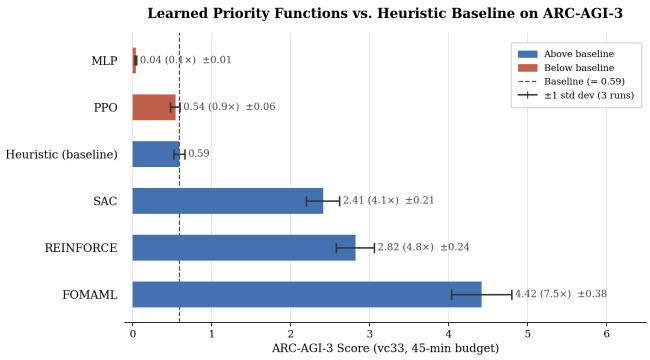
We report performance using the official ARC-AGI-3 scorecard metric, which rewards both the total number of levels solved and the specific action efficiency with which they are completed. A level solved in fewer sequential actions contributes a higher fractional weight to the total score than the same level solved after exhaustive, brute-force exploratory steps.

This metric is substantially more informative than a raw count of completed levels; two agents that solve identical levels can achieve vastly different scores based on how efficiently they navigated to each solution state. In our experiments, all agents execute nearly identical total numbers of actions, meaning score differences are driven exclusively by how rapidly each agent discovers a level's core mechanics.

We compare our agents against the native heuristic baseline from Rudakov et al., executed using the identical multi-threaded infrastructure and under the exact same time budget constraints.

5 Results

5.1 Quantitative Evaluation



Model	Mean	Std	Mult.
FOMAML	4.42	0.38	7.5×
REINFORCE	2.82	0.24	4.8×
SAC	2.41	0.21	4.1×
Heuristic	**0.59**	**0.07**	**1.0**×
PPO	0.54	0.06	0.9×
MLP	0.04	0.01	0.1×

Figure 3: Learned Priority Functions vs. Heuristic Baseline on ARC-AGI-3. Absolute scores and relative performance gains ($x\times$) are evaluated three times on `vc33` under 45-minute budgets.

Upon evaluation, FOMAML achieves a mean score of 4.42, a 7.5x improvement over the heuristic baseline's mean (0.59). REINFORCE achieves a mean score of 2.82 (4.8×) and SAC reaches 2.41 (4.1×). Both PPO (0.54, 0.9×) and the MLP (0.04, 0.1×) fall below the baseline on average. Because all agents take approximately the same number of total actions, score differences are driven by how efficiently each agent resolves levels.

To assess the reliability of these results, we ran each agent three times under identical 45-minute budgets and report the mean score and standard deviation above. The top-performing agents exhibit low relative variance: FOMAML's standard deviation of 0.38 represents roughly 9% of its mean, and REINFORCE and SAC similarly show tight spreads of 0.24 and 0.21 respectively, suggesting that their performance advantages over the baseline are robust amidst stochasticity. PPO's variance (0.06) is comparably tight, indicating consistently poor performance. The MLP's near-zero mean and minimal variance (0.01) across all three runs confirm that its underperformance is a structural property of random initialization under a tight data budget.

5.2 Qualitative Analysis

In the `vc33` game (Figure 4), levels progressively introduce new interactive elements, so what matters to click at level 1 differs from what matters at level 5. FOMAML's meta-weights are explicitly trained to produce task-weights that adapt quickly at each level boundary. When a new level begins, FOMAML resets its task-weights to the meta-initialization and then takes inner gradient steps, identifying the new level's relevant segments within the first few clicks. REINFORCE, by contrast, accumulates all experience into a single weight vector that must simultaneously balance the optimal priorities for every level it has seen, which is a harder optimization problem. This behavior is consistent across all three evaluation runs, reflected in FOMAML's low standard deviation.

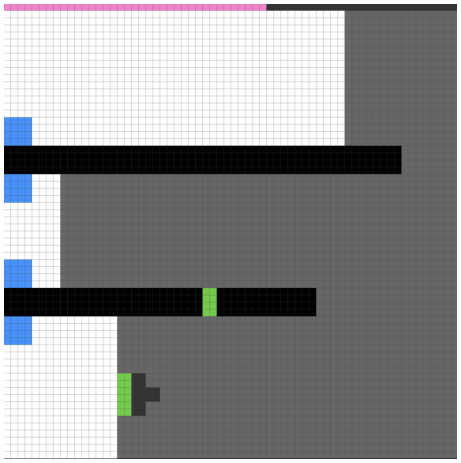


Figure 4: A representative game state from `vc33` at Level 2. Blue squares (high saturation, medium size) are correctly prioritized by all learned agents. The pink status bar is deprioritized. Green elements are newly introduced interactive objects whose transition-producing behavior must be discovered within the level's action budget.

The blue squares along the above figure's left edge are medium-sized, highly saturated segments that the warm-start initialization correctly prioritizes (clicking them reliably triggers state transitions). The pink bar at the top is flagged as a status element and assigned the lowest priority group. The green segment mid-screen and the green-and-gray cross in the lower region are newly introduced interactive elements: a static heuristic treats them identically to every other chromatic segment, while FOMAML's inner-loop reset lets it upweight them within the first few clicks of the new level.

Even without FOMAML's meta-learning structure, the warm-started linear model already improves on the static heuristic by receiving direct per-click feedback and revising its beliefs accordingly. REINFORCE may discover, for example, that `closeness_to_medium` is a stronger predictor than saturation for this particular game and shift its weights accordingly; the heuristic cannot make this adjustment regardless of how much evidence accumulates against it.

SAC's replay buffer lets it reuse each observed transition across multiple gradient steps, producing more stable Q-estimates than REINFORCE's single-step updates. Its entropy regularization additionally prevents the policy from collapsing onto a small set of high-scoring segments, which matters when later levels require clicking less obvious elements. The modest gap between SAC and REINFORCE suggests that the sample-efficiency benefit of replay is largest early in training, before REINFORCE has accumulated enough signal to stabilize.

PPO requires a full rollout buffer of 32 transitions before any gradient update, so it adapts more slowly than the per-step methods. Because only a fraction of clicks in vc33 produce state transitions, filling the buffer with informative signal takes substantially more wall-clock time than the buffer size alone implies, delaying the policy improvement that clipping and GAE are designed to stabilize.

The MLP's near-zero score (0.04) confirms that random initialization leads to poor priority assignments from the first step, and early gradient updates that overfit to level-1 segments actively harm performance on later levels. Richer model capacity does not compensate for a weak starting point when the data budget is this tight, confirming that the warm start is load-bearing for online RL in this environment.

6 Discussion

FOMAML $>$ *REINFORCE* $>$ *SAC* $\gg$ *PPO* $>$ *MLP.*

The above ranking reflects how quickly each agent produces a competent policy in a 45-minute window. FOMAML is fastest because its objective directly optimizes for rapid per-level adaptation. REINFORCE and SAC are fast because their per-step updates require no rollout accumulation. PPO

and the MLP are slow because they either defer updates (PPO's rollout buffer) or start from an uninformed initialization (the MLP).

This ordering suggests that the dominant challenge in vc33 is adapting to each level's specific interactive elements within a small number of clicks. ARC-AGI-3 is explicitly designed to reward this kind of fast hypothesis revision through its scoring metric penalizing wasted actions. FOMAML's inductive bias toward rapid local adaptation is well-matched to this structure.

A secondary finding is that the reward signal we use, a ± 1 binary value issued after every click, is sufficient for online policy improvement despite the fact that most clicks do not produce transitions. In vc33, transition-producing clicks are the informative minority, meaning positive rewards are infrequent. The reward signal is nonetheless dense in the sense that every action generates an observation, and negative signal (this segment type is not interactive at this level) is itself highly informative. This stands in contrast to level-completion reward, which would provide no gradient signal at all for the many levels that no agent completes within the time budget.

The MLP-versus-REINFORCE comparison bounds the contribution of warm-starting. The two agents differ in two respects: the MLP is nonlinear and not warm-started, whereas REINFORCE is linear and warm-started. Since additional capacity should, if anything, help rather than hurt, the fact that linear REINFORCE outperforms the MLP by more than $70\times$ indicates that the absent warm start, not limited capacity, drives the gap. This is evidence that a competent starting point matters more than expressive capacity in low-data online RL. Practitioners extending this work to richer game environments should prioritize warm-starting from a reliable hand-coded prior.

A couple limitations bound these conclusions. First, the 45-minute budget may systematically favor fast-adapting methods, since under a longer horizon PPO and the MLP might eventually exceed REINFORCE once they accumulate sufficient data on much longer game runs. Second, our feature set, though an improvement over the binary baseline, is still hand-designed, and a learned visual encoder might surface interactive cues that our four features miss.

7 Conclusion

We asked whether replacing a hand-coded visual salience heuristic with a learned priority function improves exploration efficiency in ARC-AGI-3. Our results on `vc33` suggest that it does: FOMAML achieves a $7.5\times$ mean improvement over the baseline, REINFORCE a $4.8\times$ improvement, and SAC a $4.1\times$ improvement, all without modifying the underlying graph explorer.

ARC-AGI-3's progressive level structure creates a sequence of distribution shifts, each introducing interactive elements the agent has not seen before. An agent that can identify the new level's relevant segments within its first few clicks accumulates significantly more score than one that slowly converges on a global optimum. FOMAML's explicit optimization for fast adaptation is well-matched to this structure. REINFORCE's simpler per-step updates are sufficient to beat the static heuristic but not to match FOMAML's level-boundary agility.

Future work should evaluate these agents on additional ARC-AGI-3 games to test the generality of the level-adaptation hypothesis and investigate whether a learned visual encoder can replace the hand-designed feature set.

8 Team Contributions

- **Ryan Bookman:** Developed the core online policy optimization infrastructure. Programmed the algorithmic architectures for the baseline REINFORCE model, the non-linear Neural Network, and the meta-learning First-Order MAML frameworks.
- **Steve Mendeleev:** Engineered the robust multi-threaded parallel execution infrastructure and evaluation pipeline. Programmed the PPO agent wrapper, designed the unified telemetry logger, and managed the end-to-end benchmarking runs on the central API server.
- **Kyle Feinstein:** Headed the off-policy sample-efficiency research track. Implemented the Soft Actor-Critic architecture, constructed its experience replay buffer mechanics, and tuned the automated online entropy regularization system.

- **All Authors:** Collaborated on the initial feature engineering design space, established the mathematical formulations for the continuous closeness-to-medium metric, and co-developed the algorithmic integration strategy linking the learned scoring layers to the upstream graph explorer.

Changes from Proposal We originally proposed RL replacing uniform random action selection within priority groups; instead, we apply RL to learn the priority group assignments themselves. Individual contribution roles did not change materially from the proposal: each author retained ownership of their originally assigned algorithm track, and the feature-engineering work remained a shared effort as planned.

References

Ferran Alet, Martin Schneider, Tomas Lozano-Perez, and Leslie Pack Kaelbling. 2020. Meta-learning curiosity algorithms. In *International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=BygdyxHFDS

ARC Prize Foundation. 2026. ARC-AGI-3: A New Challenge for Frontier Agentic Intelligence. arXiv:2603.24621 [cs.AI] doi:10.48550/arXiv.2603.24621

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. 2021. First return, then explore. *Nature* 590, 7847 (2021), 580–586. doi:10.1038/s41586-020-03157-9

Evgenii Rudakov, Jonathan Shock, and Benjamin Ultan Cowley. 2025. Graph-Based Exploration for ARC-AGI-3 Interactive Reasoning Tasks. arXiv:2512.24156 [cs.AI] doi:10.48550/arXiv.2512.24156

Georgios Sotirchos, Zlatan Ajanović, and Jens Kober. 2026. Search Inspired Exploration in Reinforcement Learning. arXiv:2602.00460 [cs.LG] doi:10.48550/arXiv.2602.00460

A AI Use Disclosure

We used Claude to assist with refining our wording in the paper and also to catch some bugs in our code. Claude also helped us with some unrelated boilerplate infrastructure (logging setup, threading, test scaffolding). The core project design, experiments, and analysis were completed independently by the team.