Extended Abstract

Motivation. Deep reinforcement learning (RL) has solved tasks from Atari [Mnih et al., 2013] to Dota 2 [Berner et al., 2019], but agents are usually trained and tested on a single fixed configuration, leaving open the central question of whether a policy generalizes or merely memorizes one run. We study this in a Roblox "obby" (a platforming course with ordered checkpoints and gaps that must be jumped), and follow the question all the way to a constructive answer. We ask not only whether a single course-agnostic policy transfers zero-shot, but whether the right training distribution can make it solve genuinely novel combinations of obstacle features it never saw assembled.

Method. We learn a shared actor-critic in two stages. Behavioral cloning (BC) fits a discrete softmax policy to $\sim$10K cleaned human demonstrations; PPO [Schulman et al., 2017] then fine-tunes it on a sparse checkpoint/finish reward with generalized advantage estimation [Schulman et al., 2016], a BC-anchor loss, and logit clamping. The state is local and course-agnostic (24-dim, covering relative platform offsets, ledge raycasts, a waypoint lookahead, and a grounded flag) and actions are world-relative (compass direction + jump bit). Up to 36 NPC agents share the policy and run inference in-engine in Luau. A deployment-time *edge-jump harness* lets the policy pick which jump to take at a ledge. To attack generalization we add **domain randomization** (a random course per episode) and then **procedural course composition**: we auto-generate "mixed" courses that chain random sequences of segment primitives, so the policy must learn feature transitions, not a fixed layout.

Results. Our sole success metric is the *next-platform success rate* (NPSR), the probability of clearing the next transition; we deliberately avoid full-course completion, which compounds many transitions and so mostly reflects course length rather than skill (a course is solved when NPSR $= 1.00$). The arc has three stages. (i) a single-course policy clears its training course (NPSR 0.79) but transfers to none of twelve novel courses (NPSR drops to 0–0.5); it memorizes a course-specific heading. (ii) Training one policy with domain randomization across the twelve courses lifts mean NPSR to ≈ 0.85 and yields strong within-class generalization, but it still solves none of three held-out mixed courses (held-out NPSR $0.03/0.00/0.52$). (iii) Adding six auto-generated procedurally-mixed courses to training (while keeping the three held-out mixes strictly out) raises mean NPSR across 18 courses to 0.78 and drives a held-out novel course to **NPSR** 1.00 **(solved end-to-end)** on the headline run, with the held-out NPSR jumping from $0.03/0.00/0.52$ to $0.67/1.00/0.86$ (a later three-seed run reproduces the transition-skill gains but not that exact end-to-end-solved number). We also find greedy (argmax) deployment dominates stochastic sampling on held-out courses, and that these multi-course runs over-train, so checkpoint selection matters.

Discussion. A course-agnostic architecture is necessary but not sufficient for transfer. Training-set diversity converts latent course-agnosticism into actual generalization, and compositional diversity unlocks novel combinations. The policy generalizes within a definable "obby class" (the span of our segment primitives and their transitions); its remaining wall is course length (a 13–14-platform course needs $\sim$12 consecutive transitions, which even at NPSR 0.86 rarely all succeed in one run).

Conclusion. A small shared policy, trained by BC then PPO with procedural domain randomization, moves from memorizing one course to solving novel compositions of obstacle features — bounded, measurable generalization within a specified obby class.

Parallel Deep-RL Agents for Roblox Obstacle-Course Navigation: From Single-Course Memorization to Generalizing Across Procedurally-Composed Courses

Aidan Whitedeer
Department of Computer Science
Stanford University
awd@stanford.edu

Alex Li
Department of Computer Science
Stanford University
azx0616@stanford.edu

Cheney Sang
Department of Computer Science
Stanford University
cheneys@stanford.edu

Abstract

We train a single deep-RL agent to navigate Roblox obstacle courses ("obbies") via behavioral cloning (BC) then Proximal Policy Optimization (PPO), and use it to chart the path from memorization to generalization. The agent observes a 24-dimensional, course-agnostic state, acts in a 16-action world-relative space, and up to 36 parallel agents share one policy running in-engine in Luau. Our contributions are: (1) an end-to-end Roblox RL system with no per-frame network calls; (2) an edge-jump deployment harness, with the finding that adding this control prior at deploy (NPSR 0.73) beats training with it (NPSR 0.48); (3) an evaluation whose sole success metric is a directly-measured next-platform success rate (NPSR), chosen because full-course completion mostly reflects course length; and (4) a three-stage **generalization study**, the central contribution of this work. A single-course policy transfers to none of twelve novel courses (NPSR collapses from 0.79 to 0–0.5; it memorizes a heading); **domain randomization** across those courses lifts mean NPSR to 0.85 (strong within-class transfer) but still solves none of the held-out mixed courses; and **procedural-mixed training** (auto-generated courses that chain random segment primitives) raises mean NPSR across 18 courses to 0.78 and **drives a held-out novel course to NPSR** 1.00 **(solved end-to-end)** on the headline run, with all three held-out courses showing large NPSR gains; a three-seed run (Section 8.5) reproduces the transition-skill gains but not that exact end-to-end-solved number. We further show greedy deployment dominates sampling on held-out courses and that the multi-course runs over-train. The work gives a quantitative account of how training-set composition, not architecture, drives generalization to novel obstacle combinations.

1 Introduction

Platforming games are a clean testbed for long-horizon, sparse-reward control. Success requires a sequence of precisely-timed jumps, a single misstep ends the episode, and reward arrives only at

widely spaced checkpoints. Roblox "obbies" add a practical twist. The simulator is a full game engine we do not control at the source level, so the agent is driven through a constrained scripting interface, making the engineering (how state is observed, how actions are applied, how weights reach the engine) as much a part of the problem as the learning algorithm.

We organize the project around generalization, and we follow it through three stages rather than stopping at a single negative or positive result. We keep the observation local and course-agnostic so the policy can act from any geometry, use world-relative actions so steering is a learnable map from geometry to direction, and a deployment-time edge-jump harness addresses the dominant failure mode (walking off ledges). With these fixed, we vary only the training distribution, moving from one course to many courses (domain randomization) to many courses plus auto-generated compositions of obstacle primitives. We judge success solely by the next-platform success rate (NPSR), not full-course completion, because completion compounds many independent transitions and so mostly tracks course length. This isolates the role of data diversity in transfer and lets us ask the sharpest version of the question. Can the agent solve a novel combination of features it never saw assembled? Our contributions are:

- A complete BC→PPO system for Roblox with in-engine int8 inference and a local bridge, running up to 36 parallel agents without hitting the HTTP rate limit; a multi-course trainer and a self-contained per-course evaluator share the agent's exact state function.
- An edge-jump harness with selectable modes and the finding that adding a control prior at deploy beats training with it (NPSR 0.73 vs 0.48).
- An evaluation whose sole success metric is the directly-measured next-platform success rate (NPSR), since full-course completion mostly reflects course length rather than skill.
- A procedural course generator that produces six structurally distinct primitives in two orientations (12 courses), plus a **compositional** generator that chains random primitive sequences into "mixed" courses, with a strictly held-out test set of novel mixed courses.
- A three-stage generalization result — single-course memorizes (NPSR $0.79 \to 0$–0.5 on novel courses); domain randomization gives within-class transfer (mean NPSR 0.85); procedural-mixed training expands the boundary to novel combinations (held-out NPSR up to 1.00, a course solved). We also report a greedy-beats-sampling deployment finding and an over-training / checkpoint-selection caveat.

2 Related Work

Deep RL for games. Value-based DQN mastered Atari [Mnih et al., 2013]; large-scale policy-gradient methods reached professional Dota 2 [Berner et al., 2019]; both rely on massive parallelism. We borrow the parallel-agent idea at small scale (up to 36 agents, one engine instance) with a 139K-parameter network. **Policy optimization.** PPO [Schulman et al., 2017] with GAE [Schulman et al., 2016] is our optimizer; we follow common implementation practice [Huang et al., 2022]. **Imitation.** BC dates to ALVINN [Pomerleau, 1989]; its weakness is covariate shift [Ross et al., 2011], so we use BC as an initialization and regularizer. **Exploration and curricula.** RND [Burda et al., 2019] and reverse curricula [Florensa et al., 2017] inform our distributed spawns, a simple backward curriculum [Narvekar et al., 2020]. **Generalization in RL.** Procedural generation [Cobbe et al., 2020] and domain randomization [Tobin et al., 2017] establish that training-set diversity drives transfer. Our work makes this concrete in a controlled setting and pushes one step further. We show that compositional diversity (procedurally-chained primitives) is what unlocks novel feature combinations, not just new instances of a known layout type.

3 Method

3.1 Problem formulation

We model the obby as a discounted MDP and maximize the shared policy's return $J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[\sum_t \gamma^t r_t]$.

State ($\mathbb{R}^{24}$). A pure function of the character, computed identically in the demo logger, the live environment, and the multi-course evaluator. It comprises position (optionally zeroed), a grounded

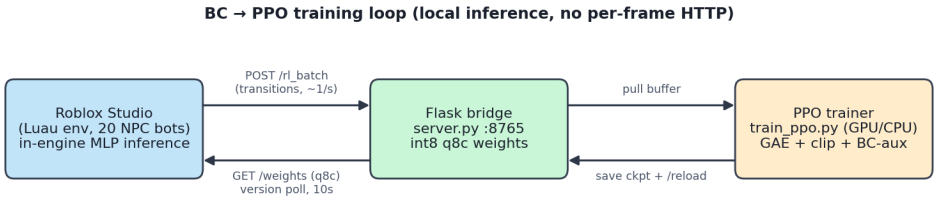

Figure 1: System overview. Parallel agents run the shared actor-critic in-engine and batch transitions ($\sim$1/s) to a local Flask bridge; the PPO trainer pulls the buffer, updates, and hot-reloads int8 "q8c" weights re-fetched on a 10 s version poll. No per-frame HTTP, so Roblox's $\sim$500 req/min cap never binds. For multi-course training, the same loop serves a self-contained evaluator that drives bots on every course and randomizes the course per episode.

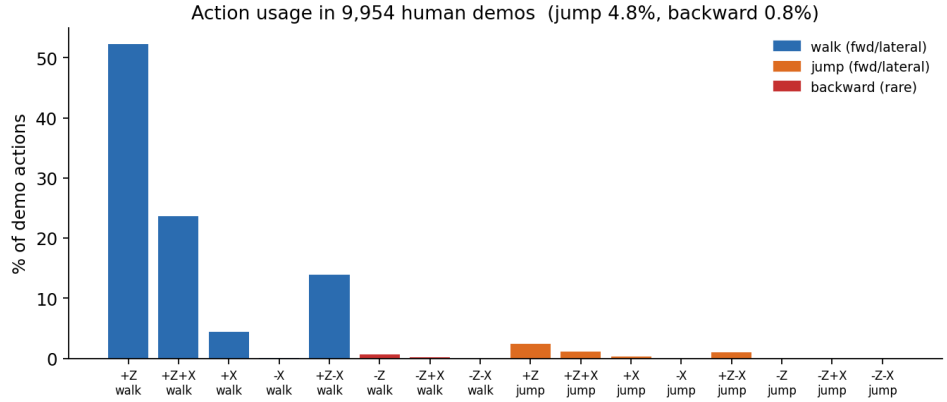

Figure 2: Action usage in the cleaned 16-action demonstration set. Forward / forward-diagonal walks dominate; grounded jumps are $\sim$4.8% and backward directions are negligible, which is exactly why a deployment harness at ledges helps.

flag, relative offsets of the three nearest platforms, a downward ground-probe, the relative offsets of the next two course waypoints, a forward ledge raycast (edge distance and next-platform height across the gap), and two lateral ($\pm 45°$) edge probes. Velocity and conveyor speeds were removed after they induced a momentum-following shortcut. Absolute position is zeroed (`DropAbsPos`) so the policy cannot memorize world coordinates.

Actions ($|\mathcal{A}| = 16$). World-relative: 8 compass directions $\times$ {walk, jump}, with $a \bmod 8$ the direction and $a \geq 8$ the jump bit. With no facing feature, position$\rightarrow$direction is a learnable map. Backward motion is rare in demonstrations (Figure 2, $\sim$0.8%).

Reward. Sparse and course-structured: $+10$ per checkpoint reached in sequence, $+15$ finish, -2 death; dense progress shaping is disabled to avoid rewarding lateral drift. An oversized walk-off penalty (-1000) was found to collapse the policy (PPO normalizes advantages, so a penalty past the reward scale only blows up the critic), so behavior is shaped with the harness, not with reward magnitude. Checkpoints persist within a session to block a die-respawn exploit.

3.2 Stages and architecture

BC. We fit a categorical policy by class-balanced cross-entropy (not L_2, so multimodal labels are not averaged) to cleaned transitions (death-fall frames dropped; mid-air "jumps" relabeled to walks). **PPO.** We fine-tune with the clipped surrogate plus a value loss, an entropy bonus, and a BC-anchor cross-entropy term; actor logits are clamped to ± 15 (unclamped logits ran away to NaN under sparse reward). Inference samples from the softmax during training (on-policy requirement). **Network.** A

shared trunk $24 \to 256 \to 256 \to 256$ with ReLU and linear actor/critic heads ($\approx$139K parameters), served to the engine as per-layer int8 "q8c" chunks.

3.3 Learning objective

The actor is a softmax over clamped logits sharing a trunk with a value head $V_\theta(s)$:

$$\pi_\theta(a \mid s) = \mathrm{softmax}\big(\mathrm{clip}(f_\theta(s), -15, 15)\big)_a. \tag{1}$$

PPO maximizes the clipped surrogate with the importance ratio ρ_t,

$$\mathcal{L}^{\text{CLIP}}(\theta) = \mathbb{E}_t\Big[\min\big(\rho_t(\theta)\,\hat{A}_t,\ \mathrm{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)\,\hat{A}_t\big)\Big], \qquad \rho_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}, \tag{2}$$

where advantages use generalized advantage estimation [Schulman et al., 2016] on the temporal-difference residual δ_t:

$$\hat{A}_t = \sum_{l \geq 0} (\gamma\lambda)^l\, \delta_{t+l}, \qquad \delta_t = r_t + \gamma V_\theta(s_{t+1}) - V_\theta(s_t). \tag{3}$$

The critic regresses to the bootstrapped returns $\hat{V}_t = \hat{A}_t + V_{\theta_{\text{old}}}(s_t)$ under a clipped squared error $\mathcal{L}^V(\theta) = \mathbb{E}_t\big[(V_\theta(s_t) - \hat{V}_t)^2\big]$. We add an (annealed) entropy bonus $\mathcal{H}[\pi_\theta(\cdot \mid s_t)] = -\sum_a \pi_\theta(a \mid s_t) \log \pi_\theta(a \mid s_t)$ and a behavioral-cloning anchor on the demonstrations $\mathcal{D}$,

$$\mathcal{L}^{\text{BC}}(\theta) = -\,\mathbb{E}_{(s,a)\sim\mathcal{D}}\big[\log \pi_\theta(a \mid s)\big], \tag{4}$$

and minimize the combined objective with Adam:

$$\mathcal{L}(\theta) = -\,\mathcal{L}^{\text{CLIP}}(\theta) + c_v\,\mathcal{L}^V(\theta) - c_e\,\mathbb{E}_t\,\mathcal{H}[\pi_\theta(\cdot \mid s_t)] + c_{\text{bc}}\,\mathcal{L}^{\text{BC}}(\theta). \tag{5}$$

The per-step reward is the sparse, course-structured signal

$$r_t = 10 \cdot \mathbb{1}[\text{checkpoint reached in sequence}] + 15 \cdot \mathbb{1}[\text{finish}] - 2 \cdot \mathbb{1}[\text{death}], \tag{6}$$

with coefficients $(c_v, c_e, c_{\text{bc}})$ and the entropy anneal given in Section 4.

3.4 Parallelism and the edge-jump harness

Server-driven agents share the policy; some start on the pad (the honest full-course cohort) and the rest are pinned across platforms (a backward curriculum). The edge-jump harness is a last-resort reflex. When a grounded agent's chosen walk would step off an edge, we convert it to a jump. Selectable modes choose which jump: `forward` (the walk direction), `argmax` (the policy's highest-logit jump), or `sample` (a jump sampled from the policy). The smart modes read the live logits, so the policy aims the jump. A grounded stuck reflex (force a jump after a few seconds of no progress) keeps agents from freezing at a hard ledge during data collection.

3.5 Procedural course generation

We generate courses from six segment primitives (*straight*, *zigzag*, *climb*, *drop*, *shrink* for narrowing pads, and *curve*), each advancing a cursor in $+Z$ with characteristic lateral, vertical, or size changes. Two generators consume these primitives. The first produces six structurally distinct single-primitive layouts (Straight, Staircase, Zigzag, Descending, Spiral, Shrink) in two global orientations (axis-varied / diagonal), giving twelve courses. The second is a compositional generator that chains a random sequence of primitives (4–5 segments of 2–3 platforms each) into a "mixed" course. Both the six procedurally-mixed training courses (`TrainMix1-6`) and the three held-out test courses (`Mix1-3`) are outputs of this same compositional generator, drawn from distinct seeds and world regions; the held-out three are fixed and never loaded during training, so the generalization test probes procedurally-generated novel courses rather than hand-built ones. Every course exposes ordered platforms, checkpoint gates, and a finish trigger so the same reward and metric apply throughout (Figure 3, gallery in Figure 7).

4 Experimental Setup

Hyperparameters. γ=0.99, λ=0.95, clip 0.2, lr 3×10^{-4}, rollout 2048, 4 epochs/update, minibatch 256, BC-anchor 0.1, logit clamp ± 15. Walk speed 13, broad-jump speed $S^{\star}$=18 studs/s, decision rate 40 Hz, absolute position zeroed. Single-course PPO uses entropy $0.02 \rightarrow 0.005$ over 150K steps. Multi-course runs randomize the course per episode; the procedural-mix run resumes from the domain-randomized policy with entropy $0.01 \rightarrow 0.004$ and no critic reset (resetting the critic with the per-platform $+10$ reward on long courses blows up the value loss). Because training runs inside the Roblox engine itself, the in-engine inference and HTTP rate caps (Appendix B) bound experience collection in real time, making each run a long serial engagement; most reported configurations are therefore single training runs, with the procedural-mix run additionally evaluated across three training seeds in Section 8.5. Single-run NPSRs should be read as point estimates (we return to this in the Limitations).

Evaluation. A scoring mode runs agents per course with the harness in a chosen mode and streams each episode's outcome. We count only episodes that started at platform 1 (so mid-course curriculum spawns do not inflate the statistics). Unless noted we report greedy (argmax) deployment at 40 Hz, our best configuration (Section 8.3).

Metric. We report a single success metric, the next-platform success rate (NPSR), and deliberately do not report full-course completion. For an M-platform course, completion requires clearing all $M-1$ transitions in one episode, so it falls off geometrically as completion $\approx$ NPSR^{M-1} and mostly measures course length rather than skill; NPSR measures per-transition skill directly and, unlike completion, does not fall off geometrically with course length. We measure it directly. An episode reaching platform m from the start banks $m-1$ successes and (if it dies before finishing) one failure, and

$$\text{NPSR} = \frac{\sum \text{succ}}{\sum \text{succ} + \sum \text{fail}}. \tag{7}$$

A course is solved when NPSR $= 1.00$ (every transition cleared in every episode). One caveat to keep in mind when reading the numbers. NPSR is survival-weighted, since every episode attempts the first transition but only survivors attempt later ones, so early transitions contribute more samples than late ones. It is therefore not a uniform average over a course's transitions but a sample-weighted one, and a high NPSR paired with a shallow "deepest" platform should be read as "reliably clears the early transitions," not "clears most of the course." We report "deepest" alongside NPSR throughout so the two are never confused.

5 Single-Course Results

5.1 Adding the edge-jump harness at deploy, not at train

Table 1 reports NPSR on the original 7-platform course at $S^{\star}$=18 for seven train/deploy harness combinations. The clean policy reaches NPSR 0.62 unaided; adding the `argmax` harness at deploy raises it to 0.73 (avg. return $20.3 \rightarrow 38.1$), the best robust deploy configuration. Matched "train-with" versions are worse (`argmax`-trained NPSR 0.48): forcing a jump during training overrides the policy's own ledge action, so it never learns to choose the jump and the logits the smart harness reads at deploy become poorly calibrated. Smart-jump (`argmax`/`sample`) beats forced-`forward`, whose fixed $+Z$ jump launches into the course's diagonal gaps. A broad-jump speed sweep (PPO retrained per speed) selects $S^{\star}$=18, and a small number of hard ledges (CP2$\rightarrow$CP3$\rightarrow$CP4) dominate the failures, exactly where the harness intervenes.

6 Generalization, Stage 1: Single-Course Memorization

We took the best single-course policy (clean PPO at $S^{\star}$=18, argmax harness at deploy) and, without any retraining, evaluated it on the twelve novel single-primitive courses (Figure 3). A self-contained Luau evaluator drives its own agents and computes the same 24-dim state relative to each course's geometry; on the original course it reproduces NPSR 0.79, matching the trained environment and validating the port.

Table 1: Original course (7 platforms), $S^{\star}$=18: NPSR by training and deploy harness. "Add-after" (clean policy + deploy harness) dominates "train-with." NPSR here is estimated as $\widehat{\text{NPSR}} = c^{1/(M-1)}$ from the per-config finish rate; it is measured directly in the generalization study.

Train harness	Deploy harness	Avg. return	NPSR
off (clean)	none	20.3	0.62
off (clean)	forward	29.5	0.55
off (clean)	**argmax**	38.1	**0.73**
off (clean)	sample	34.6	0.69
forward	forward	13.7	0.54
argmax	argmax	12.6	0.48
sample	sample	51.5	0.74

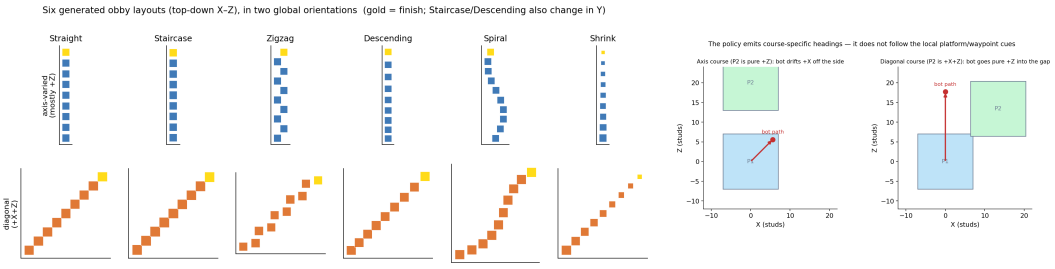

Figure 3: Left: the six single-primitive layouts (top-down X–Z) in two orientations; gold marks the finish. Right (**Stage 1 failure**): the single-course policy emits a memorized heading (drifting $+X$ off a pure-$+Z$ course or going pure $+Z$ into a diagonal gap) instead of following the local cues.

The result is clear. The policy fails to generalize to all twelve novel courses, clearing at most the first gap. NPSR collapses from 0.79 on the training course to 0–0.5 on the novel ones. On pure-$+Z$ courses (Straight, Shrink, Descending) it drifts $+X$ off the platform side (NPSR 0); re-orienting the courses to its trained $+X+Z$ axis does not help (it then goes pure $+Z$ into the gap). Inspection (Figure 3) shows the agent emits a heading tied to the training course, not one read off the local platform/waypoint cues, even though those cues correctly point at the next platform. The architecture permits transfer; a single training course does not induce it. This motivates varying the training distribution.

7 Generalization, Stage 2: Domain Randomization (Within-Class Transfer)

We then trained a single course-agnostic policy with **domain randomization**: each episode assigns each of $\sim$36 parallel agents to a random course among the twelve, with distributed (backward-curriculum) spawns, a strong BC initialization from combined multi-course demonstrations, and the argmax harness. The serving path is critical. The bridge must serve the trainer's live checkpoint (not a frozen BC), or every run reads $\sim$0.

This converts the architecture's latent course-agnosticism into real transfer. The resulting policy (`domain_rand_final`) reaches mean NPSR ≈ 0.85 across the twelve courses, up from 0–0.5 zero-shot, and now follows local cues and clears whole novel layouts of types it trained on. But it remains bounded to those layout types. On three held-out mixed courses that chain several primitives in novel orders, it solves none (held-out NPSR $0.03/0.00/0.52$; Figure 4, grey bars). Domain randomization over single-primitive courses teaches each primitive, but not the transitions between primitives that a mixed course demands.

Table 2: Held-out mixed courses (never trained), greedy deployment at 40 Hz: NPSR of the domain-randomized policy (12-course) vs. the procedural-mixed policy. M=platforms; "deepest" is the furthest platform any agent reached. NPSR rises sharply on all three, and `Mix2` reaches 1.00 (solved).

Course	M	Episodes	domain_rand_final NPSR	domain_rand_mix NPSR	deepest
Mix1	11	186	0.034	**0.667**	3/11
Mix2	11	60	0.000	**1.000**	11/11
Mix3	13	84	0.518	**0.857**	7/13

8 Generalization, Stage 3: Procedural-Mixed Training Expands the Boundary

8.1 Setup

To teach transitions without leaking the test set, we add six procedurally-mixed training courses (`TrainMix1-6`; random primitive sequences, distinct seed and world region) to the training set, giving 18 training courses, and keep the three fixed held-out mixes (`Mix1-3`) strictly out of training (a different course folder never loaded during training). We resume PPO from `domain_rand_final` and randomize across all 18 courses.

Over-training is real. Performance peaks early ($\approx$290K post-resume steps) and then regresses as the entropy bonus hits its floor (aggregate return drifted $17 \to 12$ by 750K steps, with several courses degrading); the value loss stayed bounded (no blow-up), so this is policy over-commitment, not critic divergence. We therefore select the peak checkpoint (`domain_rand_mix`) rather than the final one, a concrete instance of the checkpoint-selection pitfall in long multi-task runs.

8.2 A held-out novel course is solved

Table 2 and Figure 4 give the main result. Across the 18 training courses the policy reaches mean NPSR 0.78, at least matching the 12-course domain-randomized policy on the single-primitive courses, so the added compositional data did not cost single-course skill. The procedurally-mixed training courses reach high NPSR (`TrainMix1/5` at 1.00, `TrainMix6` at 0.95; Figure 5). On the strictly **held-out** set, NPSR rises from $0.03/0.00/0.52$ to $0.67/1.00/0.86$, and `Mix2` reaches **NPSR** 1.00: every transition cleared in all 60 episodes, i.e. the held-out course is solved end-to-end. The policy never saw these three layouts in any form, yet it solves one outright and clears most of the transitions in the others, direct evidence that it learned transferable transition skills rather than memorized layouts. The remaining wall is length: `Mix1` dies at a specific straight$\to$zigzag transition (reaching platform 3 of 11), so its 0.667 reflects reliable clearing of only the first few transitions rather than two-thirds of the course (the survival-weighting noted in Section 4), and `Mix3` (13 platforms) reaches platform 7; clearing all $\sim$12 transitions in one run is unlikely even at NPSR 0.86 ($0.86^{12} \approx 0.16$).

8.3 Greedy deployment beats sampling

On the held-out set, greedy (argmax) action selection dominates stochastic sampling (Figure 6): `Mix2` NPSR is 1.00 greedy, 0.94 at temperature 1.0, and 0.87 at temperature 1.5, and `Mix3` NPSR falls likewise ($0.86/0.82/0.76$). The policy is confident and correct on `Mix2`, so sampling only injects noise that knocks it off; hotter sampling never rescues the truly stuck courses (`Mix1`/`Mix3` are roughly flat), confirming those are genuine capability walls, not exploration artifacts. On a few training courses greedy can deterministically dead-end where sampling occasionally escapes: the clearest case is `TrainMix4` (the 14-platform course at NPSR 0 in Figure 5), where the greedy policy commits to one wrong action at its very first transition and, being deterministic, repeats that exact failure every episode, even though under stochastic sampling the same policy clears about half the course's transitions (NPSR ≈ 0.52 during training); `TrainMix2` behaves similarly. So a NPSR of 0 under greedy does not mean the policy cannot do the course, only that it deterministically locks onto a bad first move there. The optimal inference policy is thus mildly course-dependent, but greedy remains the best single deployment choice.

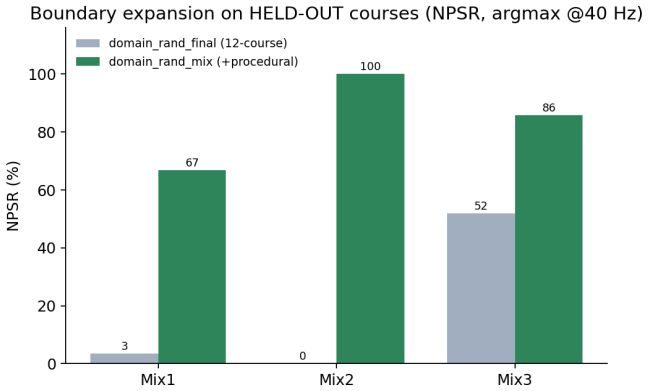

Figure 4: Held-out next-platform success rate, baseline vs. procedural-mixed policy. The boundary expands: NPSR rises on all three held-out courses, and `Mix2` reaches NPSR 1.00 (solved end-to-end).

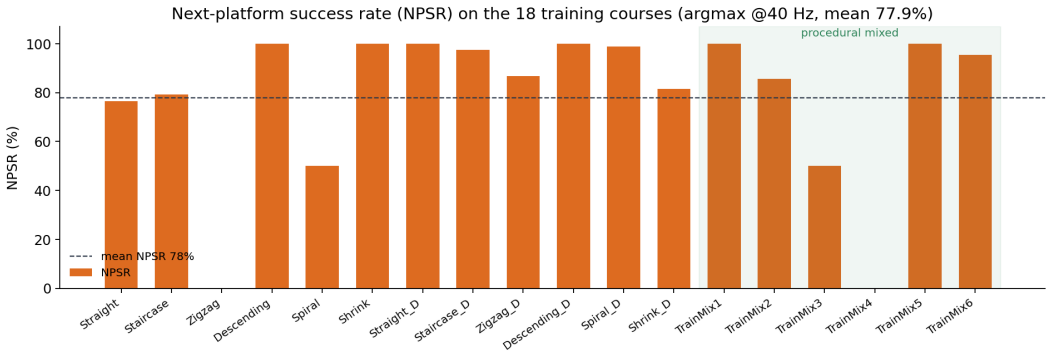

Figure 5: Next-platform success rate across all 18 training courses (greedy, 40 Hz; mean NPSR 0.78). The shaded block is the six procedurally-mixed courses; `TrainMix1/5` reach NPSR 1.00. The two NPSR-zero bars are axis-Zigzag (a genuinely unlearned course) and the 14-platform `TrainMix4` (a greedy dead-end, not an inability; Section 8.3); these and the two longest mixes set the remaining ceiling.

8.4 Extended held-out evaluation across course seeds

Three held-out courses is a small battery, so we re-ran the same compositional generator from six fresh seeds (none equal to a training seed) to produce six additional novel held-out courses (11–15 platforms), and evaluated the procedural-mixed policy on them under the identical greedy deployment, with no retraining and no parameter change. Table 3 gives the result: mean NPSR 0.52 over the six, with four of the six generalizing strongly (NPSR 0.73–0.90) and two collapsing to NPSR 0. The two zeros (Seed42, Seed88) are the same greedy dead-end documented in Section 8.3, where the deterministic policy locks onto a bad first move and repeats it, rather than an inability to do the course. Combined with `Mix1-3`, seven of nine strictly held-out courses show real transition skill (NPSR 0.67–1.00), which is direct evidence that the boundary expansion is not specific to the three originally-chosen test layouts but holds across novel instances of the generator. As elsewhere we report "deepest" alongside NPSR: a high NPSR with a shallow deepest (Seed7 at 0.73 but only platform 5 of 13) means the early transitions are cleared reliably, not most of the course.

8.5 The boundary expansion reproduces across seeds

The natural worry about a single-run headline is that it rode a lucky seed. It did not. Across three random *training* seeds (the learner's RNG, distinct from the course-generator seeds of Section 8.4) of the procedural-mix fine-tuning phase (seeds 0–2, each resuming from the same domain-randomized checkpoint, 200K steps, one fixed selection rule that never sees held-out NPSR), the central result

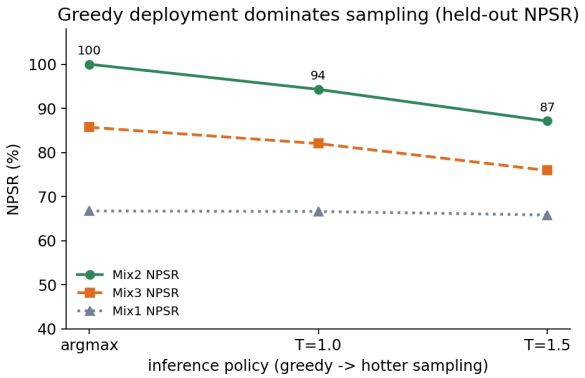

Figure 6: Inference-temperature sweep on the held-out set: greedy (argmax) dominates; held-out NPSR falls monotonically as sampling gets hotter. The stuck courses (`Mix1`) are temperature-invariant.

Table 3: Extended held-out evaluation: the procedural-mixed policy (greedy, 40 Hz) on six additional courses from fresh generator seeds, never trained on. Mean NPSR 0.52; four of six generalize strongly, two are greedy dead-ends (NPSR 0, stuck at platform 1; cf. Section 8.3). "deepest" is the furthest platform any agent reached.

Course (seed)	M	NPSR	deepest
Seed7	13	0.733	5/13
Seed23	13	0.728	12/13
Seed42	15	0.000	1/15
Seed88	13	0.000	1/13
Seed101	13	**0.899**	12/13
Seed256	11	0.785	10/11
mean	—	0.524	—

reproduces (Table 4). *Every* seed improves sharply over the domain-randomized baseline; the transition-skill gain on `Mix1` is tight across seeds (0.87–0.97, mean 0.92), and `Mix3` reproduces directionally (0.50–0.93). So the paper's actual claim, that compositional training buys transferable transition skill on novel held-out courses, is seed-robust, not a one-off.

The one quantity that does not reproduce is the boldest one. At this reduced budget *no* seed re-solved `Mix2` to NPSR 1.00: two seeds reached high NPSR (0.95 and 0.86, deepest platform 10 of 11) but short of the headline run's 1.00, and the third collapsed to NPSR 0, the characterized greedy dead-end of Section 8.3 (which recurred across both of seed 0's independent runs, and which sampled rather than greedy deployment would avoid). We are deliberately careful not to over-attribute that collapse. Because the seed runs use a 200K-step budget with final-checkpoint selection, whereas the headline used peak selection at ≈290K, the study *brackets* the headline configuration rather than replicating it, and seed 0's zero may reflect the weaker checkpoint as much as the dead-end. We chose the fixed final-checkpoint rule on purpose, as the leakage-free, identical-across-seeds choice, and accept the reduced budget as its tradeoff. Averaged over seeds the held-out NPSR is 0.76 (range 0.46–0.95). The honest summary: the qualitative boundary-expansion result reproduces across seeds, while the single end-to-end-solved number did not recur at the reduced budget. We read this as strong support for the contribution together with a sober bound on its most aggressive point estimate, not a clean replication of the headline.

Table 4: Cross-seed held-out NPSR: the procedural-mix fine-tune from three seeds (each resumed from the same domain-randomized checkpoint, 200K steps, final-checkpoint selection), evaluated greedy at 40 Hz. Mean and min–max range across seeds. `Mix1` is robust; the `Mix2`-solved result is seed-sensitive (one seed hits the greedy dead-end of Section 8.3).

Course	seed 0	seed 1	seed 2	mean (min–max)
Mix1	0.872	0.965	0.924	0.920 (0.872–0.965)
Mix2	0.000	0.947	0.857	0.601 (0.000–0.947)
Mix3	0.500	0.925	0.806	0.744 (0.500–0.925)
mean	0.457	0.946	0.862	0.755 (0.457–0.946)

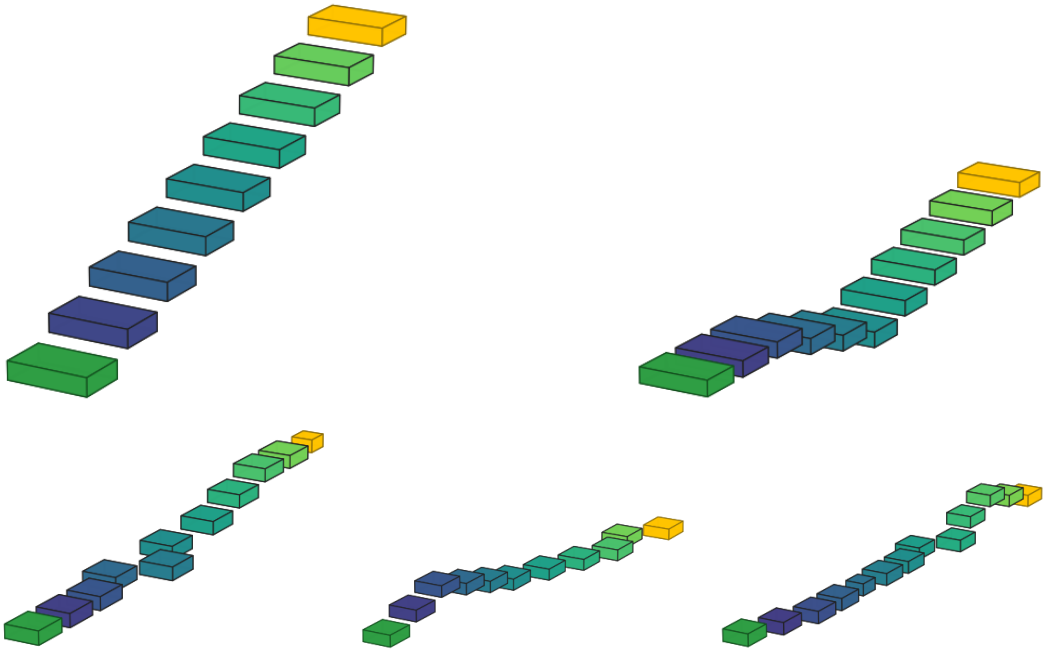

Figure 7: Course gallery, rendered from each course's exact in-place geometry (start in green, gold finish, colour graded along the course). Top: a single-primitive training course (Staircase) and a procedurally-mixed training course (`TrainMix1`). Bottom: the three strictly held-out mixed courses. `Mix2` reaches NPSR 1.00 (solved end-to-end) under the procedural-mixed policy, while `Mix1` (stuck at platform 3) and `Mix3` (13 platforms) remain length/transition-limited.

9 Discussion

Three themes run through the results. First, where a control prior is injected matters. The same edge-jump knowledge helps more added at deploy than trained-with (NPSR 0.73 vs 0.48), and an oversized penalty collapses the policy entirely, so we shape behavior with the harness and reserve reward for the task. Second, and centrally, a course-agnostic architecture is necessary but not sufficient. The identical network and observation that memorize one course generalize across twelve under domain randomization and across novel compositions under procedural-mixed training. The lever is the training distribution, and the finer lever is its compositional structure. Single-primitive diversity teaches primitives, but only chaining them teaches transitions, which is what a novel mixed course actually requires. Third, the generalization is bounded and measurable, and measuring it through NPSR rather than completion is what makes it legible: NPSR cleanly reports transition skill (0.67–1.00 on held-out mixes), whereas completion would length-gate that skill out of view (clearing all 12 transitions of Mix3 has joint probability only $0.86^{12} \approx 0.16$ even at NPSR 0.86). Reporting completion alone would have hidden the boundary expansion behind two stubborn long courses.

Limitations. We flag four, separating present caveats from work we leave to follow-ups. (i) *Seed variance.* The stage-to-stage results are single training runs, and Section 8.5 adds a three-seed run of the procedural-mix fine-tuning phase. The transition-skill gains reproduce (Mix1 NPSR 0.87–0.97 across seeds), but the end-to-end-solved headline did not recur: at the reduced budget no seed re-solved `Mix2` to NPSR 1.00 (two reached ~ 0.9, deepest 10/11, and one hit the greedy dead-end). This is a partial and confounded answer: the seeds vary only the fine-tuning phase (each resumes from the same domain-randomized checkpoint) and use a 200K-step, final-checkpoint budget rather than the headline's peak-at-$\approx$290K, so they bracket the headline configuration rather than replicate it. Each run was a long serial engagement with the in-engine rate caps, so a full-budget, full-pipeline multi-seed sweep was beyond our wall-clock budget and remains the most important follow-up. (ii) *Held-out set size.* The primary held-out test is three mixed courses from the procedural compositional generator (distinct seeds, never trained on), so it probes novel generated layouts rather than hand-built ones, and Section 8.4 extends it to nine courses across six further seeds, of which seven show strong transition skill. "A held-out course is solved end-to-end" still refers specifically to `Mix2`; the other courses improve sharply or generalize strongly but are not all solved, so the solved-outright headline remains one course. Nine is no longer a tiny sample, but it is still a single generator over a fixed primitive library, so the boundary claim is bounded by that generator's support; a much larger battery, or several distinct generators, would tighten it further. (iii) *Metric weighting.* As noted in Section 4, NPSR is survival-weighted and so over-represents early transitions; we mitigate this by always reporting "deepest" alongside it, but it is not a uniform per-transition average and should not be read as a completion fraction. (iv) *Scope and selection.* Multi-task runs over-train, so results depend on checkpoint selection; the longest courses (13–14 platforms) are not solved end-to-end despite high transition skill; greedy is the best single deploy policy but not optimal on every course; and our primitives, while composed randomly, are drawn from a fixed library, so "novel combination" is bounded by that library.

10 Conclusion

A small (139K-parameter) shared policy, trained by BC then PPO across parallel agents, progresses from memorizing a single course (no transfer: NPSR 0–0.5 on novel courses) to generalizing within a layout class under domain randomization (mean NPSR 0.85) to solving novel feature combinations under procedural-mixed training (mean NPSR 0.78 across 18 courses, a held-out course reaching NPSR 1.00 on the headline run; a three-seed run reproduces the transition-skill gains but not that exact number). The path is dictated by measurement at each stage, read through next-platform success. The negative zero-shot result isolates data diversity as the cause, domain randomization confirms it within a class, and compositional generation extends it to novel combinations. The remaining frontier, chaining many transitions over long courses, points to longer or curriculum-ordered compositions and to closing the small greedy-vs-sampling gap per course. Section 8.4 takes a first step in this direction by evaluating six further seeded held-out courses. For future work, we would like to run our algorithm across thousands of procedurally-generated courses to get a far better measure of how generalizable the policy can be, and to find a way to accomplish this within the boundaries of the Roblox engine.

11 Team Contributions

All three authors contributed equally to this project.

- **Aidan Whitedeer:** the procedural course-generation system (single-primitive and compositional "mixed" generators, held-out test set), the edge-jump harness and its selectable modes, the domain-randomized and procedural-mixed PPO training that produced the generalization results, and the report.
- **Alex Li:** state/action design, demo collection and cleaning, BC training and ablations (DropAbsPos), reward design, the NPSR metric definition, and the int8 weight transport and in-engine Luau inference.
- **Cheney Sang:** core environment scripting (episodes, parallel bots, distributed spawns), checkpoint/finish instrumentation, the Flask bridge and the multi-course trainer/evaluator, evaluation tooling, and figures.

Why the breakdown changed from the proposal. Our proposal split the work roughly evenly across a single-course pipeline (environment, BC, PPO), with generalization listed only as a stretch goal. Two mid-project findings forced a re-allocation, and we flag them here as a reflection on the research process. First, the negative zero-shot transfer result (Section 6) redirected the project away from polishing one course and toward a full multi-course generalization study, which became the bulk of the remaining effort. With course generation and multi-course training now the critical path, all three of us reoriented around it. The methodological changes behind this shift were also not anticipated in the proposal and emerged from the experiments. We reduced the action space from 6 facing-relative to 16 world-relative actions, removed velocity/conveyor features, disabled dense progress shaping, and replaced a walk-off penalty (which collapsed the policy) with the edge-jump harness. Together these turn the proposal's open generalization question into a quantitative three-stage answer.

12 Use of AI Tools

In keeping with course policy, we disclose our use of AI assistants during this project. We used them as a coding aid, primarily for debugging, boilerplate, and routine implementation support. All conceptual and design work is our own: the learning algorithm and its objective, the state/action and reward design, the procedural single-primitive and compositional course generators, the edge-jump harness, the in-engine int8 inference and Flask-bridge architecture, and the evaluation methodology (including the NPSR metric and the three-stage experimental design) were conceived, implemented, and validated by the authors. AI served only as a tool to help us realize these designs more quickly, not as a source of the ideas, methods, or results reported here.

References

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations (ICLR)*, 2019.

Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2020.

Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2017.

Shengyi Huang, Rousslan Fernand Julien Dossa, Antonin Raffin, Anssi Kanervisto, and Weixun Wang. The 37 implementation details of proximal policy optimization. *ICLR Blog Track*, 2022.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *NeurIPS Deep Learning Workshop*, 2013.

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research (JMLR)*, 2020.

Dean A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in Neural Information Processing Systems (NeurIPS)*, 1989.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations (ICLR)*, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

A Additional Experiments and Details

Domain-randomization serving fix. The bridge must serve the trainer's live checkpoint (`ppo_latest`) seeded from the initialization, or `/reload` re-serves a frozen BC and every randomized run reads NPSR $\sim$0; this single fix unlocked the within-class transfer result. **No critic reset on resume.** Resuming the procedural-mix run from a trained policy without resetting the critic keeps value targets calibrated; resetting it with the per-platform $+10$ reward on the longer mixed courses (returns up to $\sim$135) blows up the value loss. **Decision rate.** Performance is sensitive to the decision rate. Matching the trained environment's 40 Hz (rather than 30 Hz) is necessary to reproduce its results, so jump timing, not just policy quality, gates performance. **Measurement hygiene.** We count only episodes that started at platform 1; counting curriculum-spawned (mid-course) episodes would inflate the statistics.

B Implementation Details

State parity. The 24-dim state is one function of the character, duplicated byte-for-byte in the demo logger, the live environment, and the multi-course evaluator; a one-dimension drift silently breaks BC train/serve consistency. **Batched HTTP.** Per-frame posting at 20 Hz exceeded the $\sim$500/min HttpService limit and dropped $\sim$40% of data; batching $\sim$1/s is lossless. **Weight transport.** Full-precision JSON ($>$1 MB) exceeds Roblox's HTTP cap; we serialize per-layer int8 ("q8c") chunks, each $<$0.5 MB, dequantized in Luau. **Compute budget.** In-engine inference caps at $\sim$1000 forward passes/s, so agents$\times$decision-rate is held $\lesssim$1000 (e.g. 36 agents at 25 Hz for training, 18 at 40 Hz for evaluation).