AscensionAI Technical Writeup

Version: 0.8.0 **Document Date:** 2026-06-02 **Author:** Justin Chan **Repository:** https://github.com/JustinoChan/AscensionAI

Abstract

AscensionAI is a reinforcement learning system that trains an autonomous agent to play *Slay the Spire* (Ironclad) end-to-end against the live, modded game process. The system uses a structured 717-dimensional observation encoder (expanded from 585 with a 132-dim per-card deck count vector; 19 monster power slots), a 134-action discrete action space with legal-action masking, dense reward shaping, behavior cloning warm-start, and Proximal Policy Optimization (PPO) fine-tuning. Training is parallelized across multiple live game instances feeding a central offline trainer over a checkpoint-tagged rollout protocol, with fixed-seed evaluation and dashboard artifacts used to separate real policy progress from run-to-run variance. As of 2026-05-23, the observation encoder covers all combat-relevant STS1 monster powers and the reward structure includes upgrade incentives, Guardian phase-aware shaping, and tuned elite bonuses. The system integrates with ModTheSpire, BaseMod, CommunicationMod, and a bundled SpireComm Python interface; it runs on Windows under a GUI control panel and, as of 2026-05-30, **headless on a GPU-less Linux cloud VM** for unattended training (see §19). As of 2026-06-02, **deck-building is a learned skill** (Path 2, §20): the observation includes a per-card deck count vector, and card removal and upgrade selection moved from the heuristic to the RL policy with a potential-based deck-quality reward - and the cloud run is fully self-healing (continuous auto-resume + preemption auto-restart). This document describes the architecture, algorithmic choices, implementation, observed throughput, current limitations, and a phased roadmap.

Executive Summary

Aspect	Status
Environment integration	Complete - live STS via CommunicationMod / SpireComm
Observation encoder	Complete - 717-d vector (585-d feature blocks + 132-d per-card deck count vector; 19 monster power slots)
Action space	Complete - 134 discrete actions, legal-action masking
Reward shaping	Complete - dense per-step, terminal, elite win bonus, HP-scaled floor advance, spawner priority, and boss-specific signals
Behavior cloning	Complete - heuristic demos, supervised cross-entropy, resumable checkpointing
PPO fine-tuning	Complete - clipped policy, GAE, entropy/BC auto-tuning, BC anchor loss, target-KL early stop
Parallel rollout collection	Complete - multi-instance workers with checkpoint metadata and stale-rollout rejection
Offline trainer	Complete - batch merge, auto-tune, warm transfer, atomic checkpoint save
GUI control panel	Complete - Windows-native launcher, monitor, archive controls, eval set mode, RAM cap controls

Aspect	Status
Evaluation/reporting	Complete - deterministic seed sets, resumable eval logs, experiment reports, static dashboard
Win-rate convergence	Not yet demonstrated; no recorded full victory in the published fixed-seed evals
Cross-platform support	Windows GUI is primary; headless Linux cloud path added (GCP spot VM, no GPU) - see §19
Cloud deployment	Complete - one-shot installer + headless worker/trainer launcher on a GCP c3-standard-22 spot VM; self-healing (continuous auto-resume + preemption auto-restart), see §19/§20
Learned deck-building	Complete (Path 2, §20) - 717-d per-card deck vector; card removal + upgrade RL-controlled; potential-based deck-quality reward; warm-transferred 585->717

Headline metrics (current state):

- Observation dimensionality: 717 (585 feature blocks + 132-d per-card deck count vector; previously 585, 530)
- Action space size: 134
- Current offline PPO architecture: 717 -> 512 -> 256 -> 256 -> {134 logits + 1 value}, GELU, ~571K parameters
- Monster knowledge base: 66 monsters x 7 behavioral flags x 8-d identity embedding
- Monster power encoding: 19 powers per monster (strength, vulnerable, weakened, artifact, ritual, curlup, thorns, angry, sharphide, modeshift, enrage, curiosity, intangible, invincible, timewarp, beatofdeath, malleable, lifelink, regenerate)
- Training scale: 16,900+ PPO rollout games, 1,856+ update batches
- Training avg floor (last 500): 15.1, elite WR 78%, boss WR 21%
- Best single training run: floor 50 (Act 3)
- Per-step inference: <5 ms on CPU (no GPU required for play/eval)

Table of Contents

---

1. Project Basics

AscensionAI is a reinforcement learning project for training an AI agent to play *Slay the Spire*, currently focused on Ironclad. The project connects to a live *Slay the Spire* process through ModTheSpire, BaseMod, CommunicationMod, and the bundled SpireComm Python interface. The Python side wraps the game in a Gymnasium-style interface with a fixed observation vector, discrete action space, legal-action masking, shaped rewards, and PPO training.

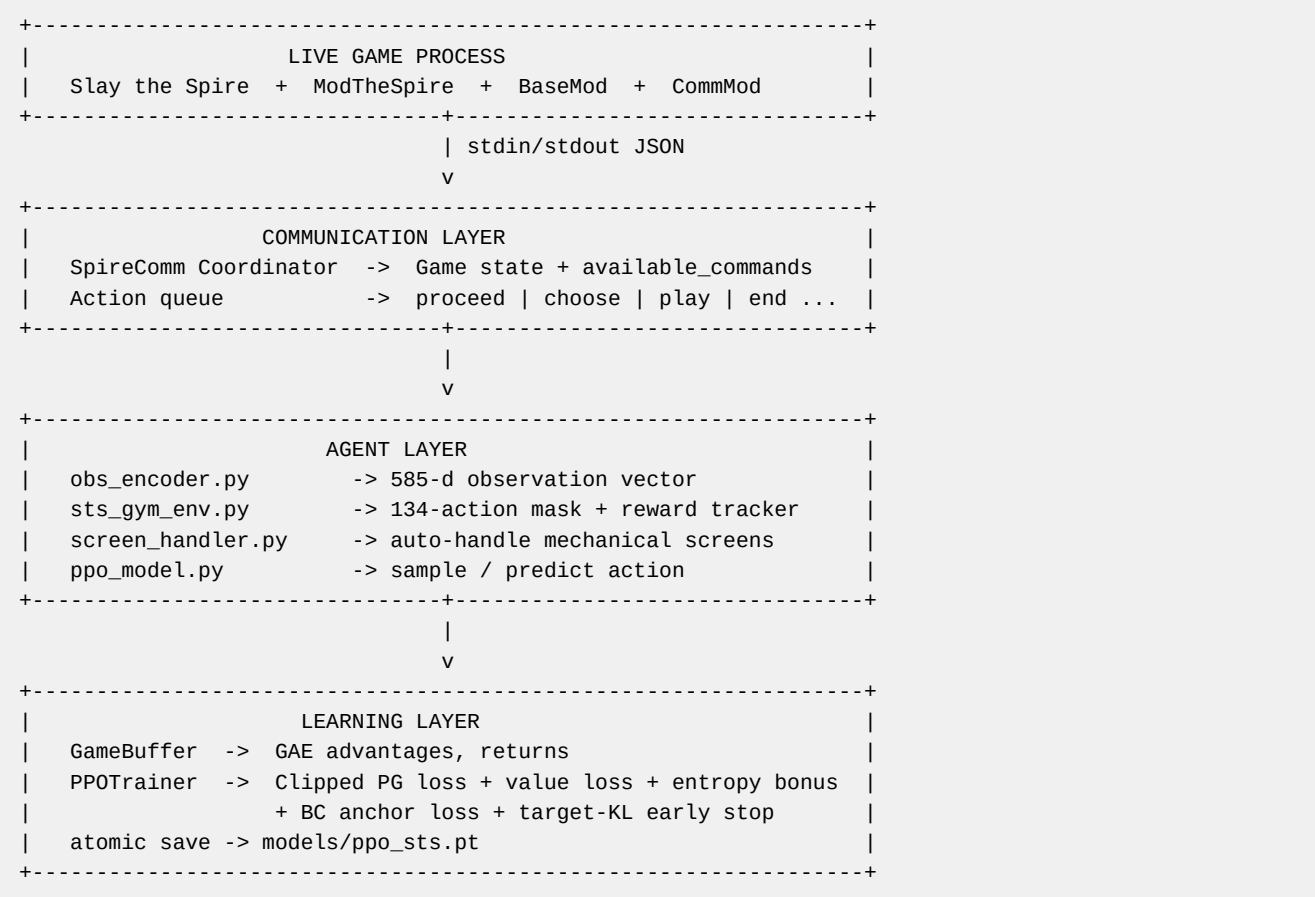
The project is designed around long-running autonomous training rather than one-off scripted play. It supports behavior cloning warm starts, PPO fine-tuning, parallel rollout collection, offline PPO updates, greedy evaluation, passive game logging, training-progress plotting, and a Windows control panel for launching and monitoring multiple game instances.

Configuration	Default
Game	Slay the Spire (Mod the Spire)
Character	Ironclad
Ascension level	0
Environment interface	CommunicationMod via SpireComm
RL framework	Custom PPO implementation in PyTorch
Policy	Actor-critic MLP; current offline PPO default is (512, 256, 256) with GELU, while legacy checkpoints use (256, 256) with Tanh
Warm-start	Behavior cloning from a hand-coded heuristic
Scaling strategy	Multiple rollout workers + central offline trainer
Hardware target	CPU inference/play; offline trainer can run on CPU, GPU, or auto-selected device when available

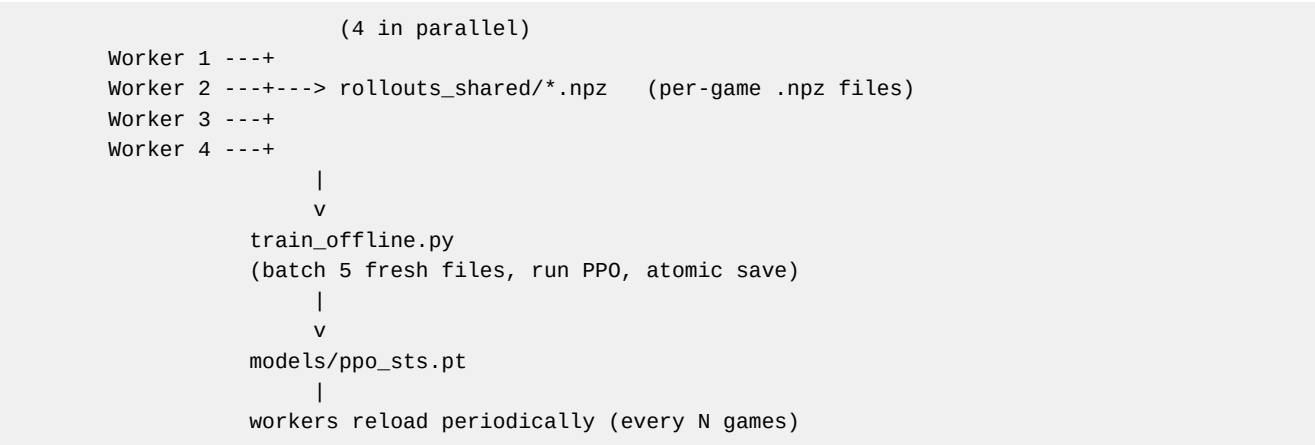
---

2. System Architecture

The project is organized around a four-tier pipeline: live game, communication layer, agent layer, and learning layer.



For parallel training the topology fans out:



Critical design decisions:

Decision	Rationale
CPU-only inference	Network is small (~235K params); GPU saturated by other costs (game speed, IPC).
Per-game .npz rollouts	Each rollout is self-contained, atomic, deletable, and carries checkpoint metadata.
Stale-rollout rejection	Workers can lag the trainer; old rollouts harm PPO objective if importance ratios drift too far.
Heuristic + RL hybrid	RL owns all strategic decisions (combat, card rewards, rest sites, map pathing, boss relics, events); heuristics handle shops and mechanical screens.

Decision	Rationale
Action masking (not penalty)	Illegal actions get probability zero in sampling - no wasted gradient steps on impossible plays.

---

3. Observation Space

Total observation length: **585 floats** (= ~2.3 KB per state). Expanded from 530 on 2026-05-23 by adding 11 STS1-verified monster powers.

3.1 Observation block breakdown

Block	Dim	Description
Player state	15	HP / max HP, energy, block, gold, floor, act, in-combat flags
Screen-type one-hot	14	NONE / MAP / EVENT / CHEST / SHOP / REST / ... / HAND_SELECT / GRID / GAME_OVER
Hand cards (10 slots x 16)	160	Per-card: identity, type, cost, upgrade, exhausts, damage/block, playable
Monsters (5 slots x 30)	150	Per-monster: HP/block/intent + 8-d identity embedding + 3 move-history IDs + 7 behavioral flags
Player powers	20	Strength, Dexterity, Vulnerable, Weak, Frail, Ritual, etc.
Monster powers (5 x 19)	95	Per-monster: 19 power slots covering all combat-relevant STS1 buffs/debuffs
Choice list features	7	Number/type-distribution of currently offered choices
Relics	25	Relic feature bag (presence + key-effect flags)
Potions (5 x 8)	40	Per-slot: id, target type, cost, presence, value flags
Deck profile	20	Card-type distribution, average cost, curse/status counts, upgrade ratio
Map lookahead	39	4 next choices x (one-hot type + 3-floor BFS density) + 3 globals
Total	585	

3.1.1 Monster power slots (19 per monster)

The 19 encoded monster powers cover all combat-relevant STS1 buffs and debuffs:

Index	Power	Key monsters
0	Strength	Many (Cultist, Nob, etc.)
1	Vulnerable	Applied by player
2	Weakened	Applied by player
3	Artifact	Bosses, elites
4	Ritual	Cultist
5	Curl Up	Louse variants
6	Thorns	Spiker, player-applied
7	Angry	Book of Stabbing
8	Sharp Hide	Guardian (defensive mode)
9	Mode Shift	Guardian (phase counter)
10	Enrage	Gremlin Nob
11	Curiosity	Awakened One
12	Intangible	Nemesis
13	Invincible	Corrupt Heart
14	Time Warp	Time Eater
15	Beat of Death	Corrupt Heart
16	Malleable	Writhing Mass
17	Life Link	Darklings
18	Regenerate	Awakened One, burning elites

An unrecognised-powers logger writes unknown power IDs to `logs/unrecognised_powers.log` for coverage auditing.

3.2 Monster knowledge base

The encoder includes a built-in database of all **66 STS1 monsters**. Each monster contributes:

- Identity embedding (8-d): unique deterministic fingerprint per monster ID (cached, L2-normalized).
- Move history (3-d): last move, second-to-last move, and current intent's move ID.
- Behavioral flags (7-d): binary flags for hidden mechanics that would otherwise take thousands of games to discover.

Flag	Meaning	Example monsters
enrages_on_skill	Punishes skill cards	Gremlin Nob
splits_low_hp	Splits at ~50% HP	Slime Boss, Acid Slime L
scales_strength	Stacking strength buff	Cultist, Time Eater
multi_attacker	Multi-hit attack patterns	Hexaghost, Nemesis, Champ
retaliates	Reactive damage on hit	Spiker, Shelled Parasite
escapes	Can flee combat	Mugger, Looter, all gremlins
spawns_minions	Summons more enemies	Gremlin Leader, Reptomancer, Bronze Automaton

The `spawns_minions` flag is the agent's primary signal for the "focus the spawner" strategy. It is paired with a strong reward bonus (see §5).

3.3 Map lookahead

Rather than encoding only the immediate map choice, the encoder runs a **breadth-first traversal of the next 3 floors** from each candidate node, counting reachable rooms by type (Monster, Elite, Rest, Shop, Unknown, Treasure). This lets the policy reason about path *consequences* (e.g., "this path leads to two elites and no rest") rather than only the next room.

---

4. Action Space

Total: **134 discrete actions**, with a legal-action mask computed each step.

Action range	Indices	Count	Description
Targeted card play	0-49	50	hand slot (0-9) x monster slot (0-4)
Untargeted card play	50-59	10	hand slot 0-9
End turn	60	1	
Targeted potion	61-85	25	potion slot (0-4) x monster slot (0-4)
Untargeted potion	86-90	5	potion slot 0-4
Choice selection	91-130	40	choice index 0-39 (events, rewards, map, shops)
Proceed	131	1	
Leave / cancel	132	1	
No-op (request state)	133	1	
Total		134	

The legal mask zeros out actions that the current screen does not support. For example, on a CARD_REWARD screen only choice indices and the skip option (proceed) are legal; on a combat turn only playable hand cards (with valid energy) and end-turn are legal.

Why one fixed action space rather than a per-screen variable space: the policy and value heads stay structurally constant, which makes BC and PPO training simpler and lets a single policy cover combat and non-combat decisions.

---

5. Reward Shaping

Reward shaping is dense to accelerate early learning. All reward sources fire each environment step. Spawner-related shaping is intentionally amplified.

5.1 Reward weights

Source	Magnitude	Notes
Gold gain	+0.01 / gold	
New relic	+1.0	
Max HP gain	+0.10 / HP	Rewards Feed events, relics that raise max HP
Card removed from deck	+0.2 / card	Card removal events
Floor advanced	+0.50 + 0.25 x (HP / max HP)	Hybrid: base progress + HP-scaled bonus
HP loss	?0.08 / HP	Constant penalty to discourage damage taken
Generic enemy damage dealt	+0.015 / HP	Applied to total enemy HP delta
Generic monster killed	+0.75 / kill	
Priority monster damage	+0.03-5.0 / HP	Spawners, boss minions (Donu, TorchHead, BronzeOrb)
Priority monster kill	+1.0-5.0	Weighted by threat level
Rest-site upgrade	+0.30 / upgrade	Counters over-resting at high HP by rewarding smithing
Elite win bonus	+4.0	Awarded on floor advance after elite victory
Boss kill reward	+8.0 + up to +8.0 x HP ratio	Scaled by HP preserved through boss fight
Act advanced (act 2+)	+12.0	Encourages full-act progression
Victory	+60.0	Terminal
Defeat	?25.0 + 0.25 x floor	Floor-prorated to avoid instant runs being equally bad

5.2 Rationale

- HP loss penalty is calibrated so a 10-damage hit costs ?0.8, roughly equal to a generic monster kill (+0.75). This balances offense and defense early in training.
- Priority monster shaping is the critical incentive against "minion farming." Without it, killing easy minions can feel as rewarding as killing the spawner per unit time.
- The elite win bonus (+4.0) makes elite fights clearly positive-EV despite their HP cost (~31 HP average). Training data showed that games with 2+ elites doubled the Act 1 boss win rate, but the reward function made elites reward-neutral. The bonus corrects this.
- The rest-site upgrade reward (+0.30) prevents over-resting at high HP. Without it, the HP loss penalty (0.08/HP) makes topping off immediately rewarding even at 73/80 HP, while upgrade benefits are delayed. The upgrade reward makes smithing competitive with healing.
- The HP-scaled floor advance (0.50 base + 0.25 x HP ratio) gives a dense per-floor gradient: arriving at the boss at 75/80 HP is worth 0.73, arriving at 30/80 is worth 0.59. This teaches HP conservation without punishing survival at low health.
- The act-advance bonus is a structural shaping term: it discourages stalling in act 1 and rewards real progression.

5.3 Shaped vs. terminal balance

For a typical successful 16-floor run with 1 boss kill:

```
Shaped:   ~35-55  (gold, damage dealt, kills, floor progression, elite bonus)
Terminal: boss kill +8 to +16, act advance +12
Total:    ~55-80
```

For a defeat at floor 5:

```
Shaped:   ~5-10
Terminal: -25 + 5*0.25 = -23.75
Total:    ~-15 to -20
```

The shaped portion is meaningful but not so dominant that the terminal signal becomes noise.

---

6. Behavior Cloning

BC is a supervised warm-start phase. A hand-coded heuristic plays full games while the policy network learns to imitate its action distribution.

6.1 Heuristic coverage

The heuristic handles every decision surface:

- Combat: card scoring (block when about to take damage, attack otherwise), spawner targeting (priority over low-HP minions), potion usage thresholds
- Card rewards: type-weighted picks favoring scaling damage and key skills
- Events: routed through pick_event with hand-mapped good/bad choices
- Boss relics, rest sites, shops, map paths
- Mechanical screens (chest open, grid confirm, hand-select fallback)

6.2 Resumable BC checkpointing

Long BC collection (150-200 games) takes 1-3 hours and is fragile to STS crashes. The system saves a per-game checkpoint at `models/ppo_sts_bc_progress.npz` after every completed BC game:

Field	Type
observations	float32 array
actions	int64 array
action_masks	bool array
games_done	int64
total_steps	int64
target_games	int64
saved_at	ISO timestamp

If STS crashes at game 145/150, restarting the same mode resumes from the saved demos. The progress file is automatically removed after successful BC training.

6.3 Supervised training loss

```
L_BC = CrossEntropy(policy(obs), heuristic_action_id)
       restricted to legal actions via action mask
```

Default: 30 epochs, batch size 64, learning rate 1e-3, ~200 BC games producing ~30k-60k labeled transitions.

---

7. PPO Fine-Tuning

PPO fine-tunes the BC policy through online RL. The algorithm follows the standard PPO formulation with several stability additions specific to BC-warm-started agents.

7.1 Loss decomposition

```
L_total = L_PG (clipped) + c_v * L_VF + c_e * H(pi) + c_BC * L_BC_anchor
```

Term	Coefficient	Purpose
L_PG (clipped surrogate)	1.0	Standard PPO policy gradient with clip epsilon = 0.15-0.20
L_VF (value loss)	0.5	MSE between predicted value and discounted return
H(pi) (entropy bonus)	0.05 -> 0.01 (annealed)	Encourages exploration early, exploitation late
L_BC_anchor	0.02	KL-style regularizer pulling toward original BC distribution on demo states

7.2 Stability mechanisms

- Target-KL early stopping: if approximate KL between old and new policy exceeds 0.03, the current update halts. Prevents large policy jumps.
- Entropy annealing: linear decay from 0.05 to 0.01 over the configured PPO game budget (or first 200 games in unlimited mode). Early exploration, late exploitation.
- BC anchor loss: evaluated on a held-out subset of the BC demo set. Anchors the policy near the heuristic distribution and reduces catastrophic forgetting of useful prior behavior.
- Gradient clipping: max norm 0.5.
- GAE returns: gamma = 0.995, lambda = 0.95.
- Per-game PPO batches: updates fire every --games-per-update (default 4) completed games. Larger batches stabilize gradients at the cost of update frequency.

7.3 Target-KL diagnostic example

For a healthy run, approximate KL per epoch should sit roughly in the range:

```
Epoch  ApproxKL     Action
1      0.005-0.015  ok
2      0.010-0.025  ok
3      0.015-0.030  approaching limit
4      >0.030       early-stop kicks in
```

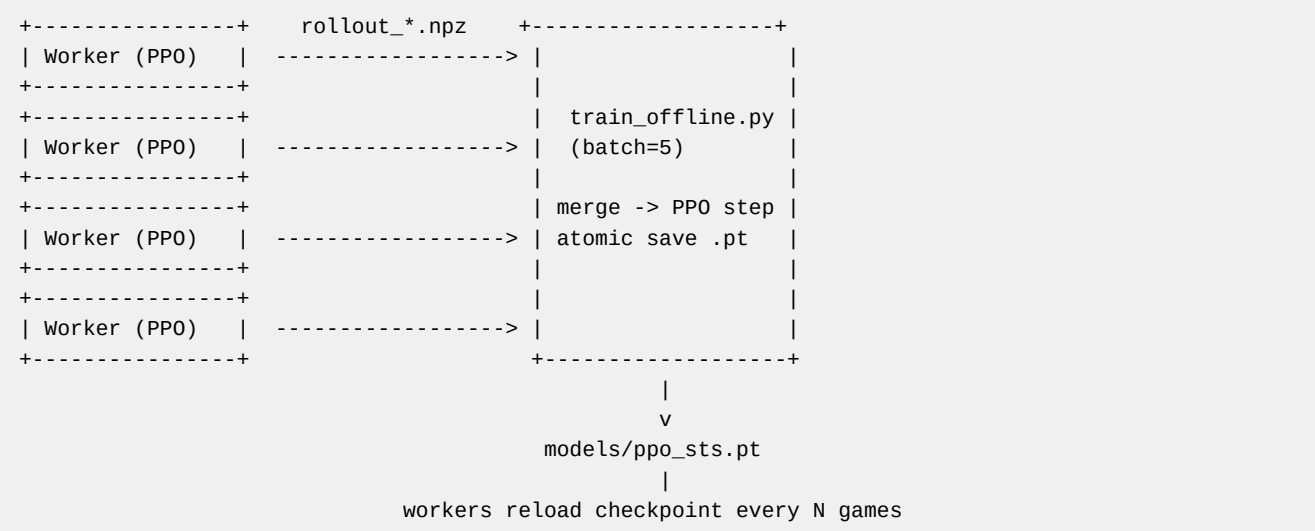
A consistently early-stopping run indicates the learning rate or entropy is too high for the current data distribution.

---

8. Parallel Training Architecture

Live-game training is bottlenecked by single-instance simulation speed. The parallel architecture decouples data collection from policy updates.

8.1 Components



8.2 Rollout file contents

Each `.npz` file represents one completed game:

Field	Type	Description
obs	float32 [T, 585]	Observation history
actions	int64 [T]	Sampled action ids
rewards	float32 [T]	Per-step shaped + terminal
dones	bool [T]	Episode termination flags
masks	bool [T, 134]	Legal-action masks at decision time
log_probs	float32 [T]	Old log-probs for importance sampling

Field	Type	Description
values	float32 [T]	Old value estimates
meta.checkpoint_id	int64	Update count of the checkpoint that produced this rollout
meta.game_outcome	dict	Final floor, victory flag, hp, act

8.3 Stale rollout handling

The trainer rejects rollouts that lag too far behind the current checkpoint:

```
if (current_update_count - rollout.checkpoint_id) > MAX_LAG (default: 10):
    reject -> "stale" counter incremented
```

Workers reload the model checkpoint periodically (default: every 5 games) to keep their on-policy assumption approximately valid.

8.4 Throughput multiplier (estimated)

Workers	Games / hour (with Super Fast Mode)	Wall-clock cost
1	~30-60	baseline
2	~60-110	low marginal CPU cost
4	~110-200	comfortable on 8-core / 16GB
6	~140-250	requires 16+ GB RAM, watch for STS instability
8+	diminishing	mod thread contention dominates

---

9. Performance and Throughput Estimates

Live-game RL fundamentally throttles on game-simulation speed. Numbers below are observed/estimated on an AMD Ryzen 7 9800X3D, 32 GB RAM, with Super Fast Mode at 200% speed.

9.1 Time per game

9.2 Estimated total training cost

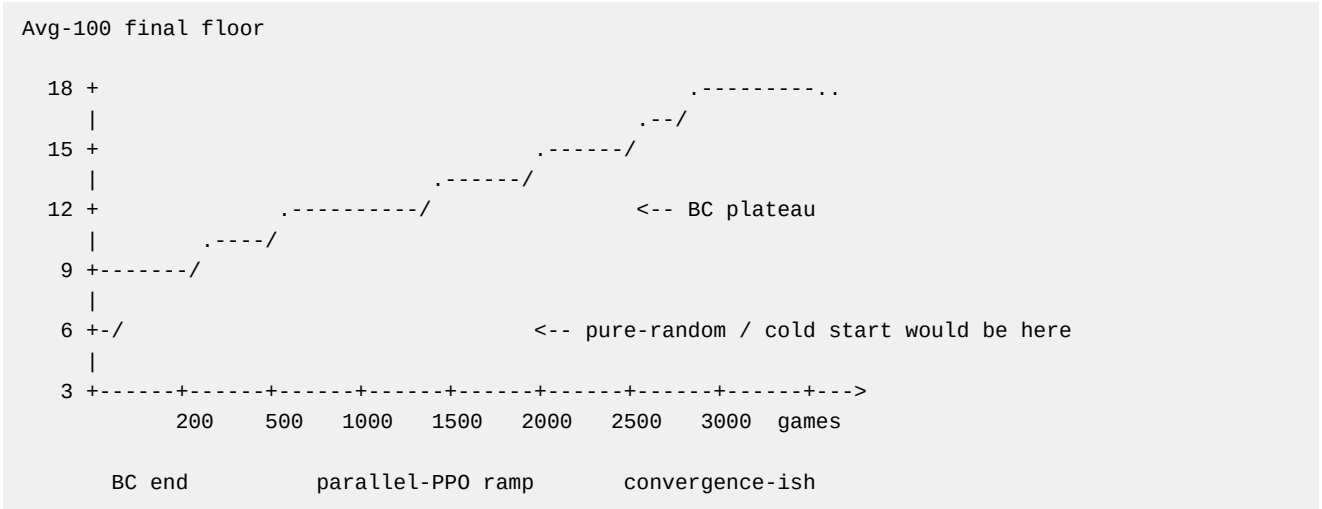
Phase	Games	Time per game	Total wall-clock
BC collection	200	~60 s	~3 hours (1 instance)
BC supervised training	-	-	~2 minutes (CPU)
Initial PPO sanity	50	~75 s	~1 hour (1 instance)
Parallel PPO main run	1500-3000	~75 s	8-24 hours (4 workers)
Greedy evaluation	200	~50 s	~3 hours (1 instance)

9.3 Memory footprint

Item	Approximate size
Policy/value network parameters	~235K floats ~ 1 MB
Per-state observation	2.1 KB
One full game (200 steps)	~1.5 MB rollout .npz
Active GameBuffer (4 games)	~6-10 MB
1000 BC demos	~3-5 MB compressed

9.4 Expected learning curve (illustrative)

This is a rough sketch of the *shape* a successful run typically has on shaped reward, not actual numbers from this codebase:



What this implies in practice:

- BC alone tends to stall in act 1 / early act 2 (~floor 9-13 average).
- The first ~500 PPO games typically don't move avg-100 much because of buffer warm-up and entropy still being high.
- Real gains usually start after ~1000 cumulative PPO games and after entropy has annealed.

These are *expected* numbers based on similar STS RL work and are not yet validated by a long converged run on this codebase. Treat them as targets, not measurements.

10. Code Organization

10.1 Top-level

File	Responsibility
AscensionAI.pyw	Windows control panel: launches STS instances, writes CommunicationMod commands, tails logs, detects worker crashes, provides Stop Now / Finish && Stop.
launch_workers.ps1	PowerShell command-line launcher (BC, PPO, eval, logger, parallel modes).
requirements.txt	numpy, torch, gymnasium, psutil, matplotlib.

10.2 Scripts

File	Responsibility
obs_encoder.py	585-d observation construction, monster knowledge base (19 power slots), map lookahead.
sts_gym_env.py	Action space, action masking, flat-id ? SpireComm action conversion, RewardTracker.
ppo_model.py	GameBuffer (GAE), PPOTrainer (clipped policy, value loss, entropy, BC anchor, target-KL early stop, atomic checkpoint save/load).
behavior_clone.py	Heuristic policy + supervised BC training driver. Includes resumable progress checkpointing.
train_bc_ppo.py	End-to-end BC -> PPO pipeline in a single session.
train_ppo.py	Single-instance PPO training.
rollout_worker.py	Worker for parallel rollout collection.
train_offline.py	Offline trainer that consumes rollouts and updates the shared model.
screen_handler.py	Shared mechanical screen handling for non-decision surfaces.
eval_model.py	Greedy (sampling-free) evaluation harness with fixed-seed support.
game_logger.py	Passive game-state recorder for debugging and trace analysis.
fight_tracker.py	Per-fight elite/boss outcome tracking. Handles the in-combat-on-death edge case.
game_data.py	Static card / relic / potion knowledge tables.
make_eval_seeds.py	Deterministic seed list generator for fair comparisons.
analyze_training_rewards.py	Reads training_stats.csv; reports reward-vs-outcome correlations.
analyze_trace.py	Analyzes passive-logger traces for debugging.
plot_training.py	Matplotlib training-curve plot generator.

10.3 External

Path	Description
external/spirecomm/	Bundled SpireComm library; speaks the CommunicationMod stdin/stdout protocol.

---

11. Reliability and Safety

Long-running modded *Slay the Spire* sessions are fragile. Concrete reliability mechanisms:

Mechanism	Where	What it prevents
Atomic model save (tmp + os.replace)	ppo_model.py, train_offline.py	Half-written .pt files on crash
Atomic rollout write	rollout_worker.py	Half-written .npz files
Per-game BC progress checkpoint	behavior_clone.py, train_bc_ppo.py	Lost demos on STS crash mid-collection
Worker heartbeat files	rollout_worker.py	Silent worker hangs
Worker crash detection + relaunch	AscensionAI.pyw	Stalled multi-instance runs
Orphan Java/Python sweep	AscensionAI.pyw	Leftover processes after Stop Now / Finish && Stop
Stuck-state detection	train_bc_ppo.py, train_ppo.py	Infinite loops on rare game states; dumps to bug_debug.log
HAND_SELECT fallback	behavior_clone.py, screen_handler.py	Headbutt/Warcry-style infinite loops when choice_list is empty
on_error smart recovery	All training scripts	"Invalid command: proceed" loops; auto-falls-back to choose 0
CommunicationMod error logging	All scripts	Silent skipping of failed commands
Stale-rollout rejection	train_offline.py	Off-policy drift from lagging workers

---

12. Current Limitations and Known Issues

12.1 Scope and characters

- Ironclad only. Card-stat tables and heuristic logic are Ironclad-specific. Other characters would need new card stats and heuristic policies.
- Default Ascension 0. Higher ascensions add elite/normal modifiers and would need additional reward tuning and BC heuristic adjustments.

12.2 Throughput

- Live-game bottleneck. Even at maximum Fast Mode + Super Fast Mode 200%, one game costs ~30-90 seconds. There is no headless *simulator* integration - the cloud path (§19) still runs the real game, just without a display.
- Single-machine ceiling. ~6-8 concurrent instances is realistic on a 16-core machine; beyond that, mod thread contention and CPU saturation dominate (each instance is ~2 vCPU). Cloud spot VMs raise the ceiling per node but the per-instance cost is unchanged. Multi-machine pooling exists but is manual.

12.3 Algorithmic / training risk

- No proven win-rate convergence yet. The full pipeline has now produced multi-thousand-game PPO checkpoints and 200-game fixed-seed evaluations, but no published eval has recorded a full victory. The best documented PPO eval reached floor 42, and the main bottleneck remains Act 1 boss conversion around floor 16.
- Reward shaping bias risk. Dense shaping accelerates learning but can encode biases (e.g., over-prioritizing damage dealt vs. healthy block usage). The reward-correlation analyzer exists but tuning is ongoing.
- PPO can forget BC. With high entropy / learning rate / KL movement, PPO can drift away from valuable BC priors. The BC anchor loss and auto-tuned BC coefficient mitigate this but do not eliminate the exploration-valley risk.
- Rollout staleness. Workers cache the model and lag the trainer by N games. Beyond ~10 updates lag the importance ratios are stale; rollouts get rejected, throughput drops.

12.4 Coverage gaps

- Shop logic is fully heuristic. Shop rooms are entered once per floor (a deliberate hack to prevent loops). Real budget-aware shopping should eventually be RL-driven.
- Grid screens are heuristic-handled. Match-and-keep, transform, and special grids have specialized handlers but no learned policy. These have low strategic value.
- Events with disabled options can be brittle. Most edge cases are handled; new mod-introduced events may not be.

12.5 Platform

- Windows GUI is primary. The control panel, PowerShell launcher, process cleanup, and Steam path detection assume Windows. A headless Linux path now exists for the core training loop (§19), but the GUI itself is not ported; macOS is untested.
- Path handling. Several scripts use Windows path separators in defaults; the vm/ scripts handle the Linux equivalents but other tooling switching shells requires care.

12.6 Evaluation

- Eval game count is the bottleneck for confidence. A 20-game greedy eval has ~22% standard error on win rate; meaningful comparisons need 100-200 games on the same fixed seed list.
- No inter-checkpoint regression detection. If a PPO update degrades performance, it is not flagged automatically; only the rolling-average plot reveals it.

12.7 Known infrequent edge cases

- HAND_SELECT screens for in-combat card selection (Headbutt, Warcry) can have empty choice_list while scr.cards is populated. Patched, but rare new variants of this pattern may surface.
- Some events (Match and Keep, Falling) produce non-standard grid states; handled via specialized pickers but new modded variants may still hang.
- ModTheSpire's launch dialog can occasionally need a manual click; handled by a fallback click in the control panel.

---

13. Roadmap and Future Directions

The roadmap is organized into three horizons. Items are listed in rough priority order within each.

13.1 Short-term (next 1-3 months)

1. Run the first full converged training to ground all the estimates above. Target: 200 BC games + 3000 PPO games at 4 workers. Goal: verify avg-100 floor trends upward and victory rate exceeds heuristic baseline.
2. Controlled greedy evaluation pipeline. Run eval_model.py with seeds/eval_200.txt after every 250-500 PPO games. Track win rate, avg floor, elite win rate, boss win rate as time series.
3. Reward-correlation regression check. Use analyze_training_rewards.py to verify shaped reward correlates positively with actual outcomes (final floor, victory). If correlation is weak, retune weights.
4. Checkpoint versioning. Replace single-file ppo_sts.pt with named checkpoints (ppo_sts_g{N}.pt) plus a current.pt symlink. Allow rollback if PPO regresses.
5. Run manifest. Persist hyperparameters, git commit, BC count, PPO count, worker count, entropy schedule, and final-eval results to runs/run_YYYYMMDD/manifest.json.

13.2 Medium-term (3-6 months)

1. Move shop decisions into RL. Build a structured shop observation block (cards on offer with prices, relics on offer, gold available) and let the policy decide buy/skip/remove.
2. Move grid screens into RL. Match-and-keep, transform, and similar grids are good candidates for a learned policy, since the heuristic is already weak there.
3. Headless or accelerated simulator. If a reliable open-source STS simulator becomes available, integrate it as a parallel path. A headless simulator would unlock 10-100x throughput.
4. Better experiment tracking. Integrate with TensorBoard or a lightweight equivalent for live training-curve dashboards.
5. Saved policy-state snapshots for offline inspection. Instead of needing a live eval to inspect top actions, persist a curated set of representative game states and run the policy on them.
6. More aggressive PPO ablations. Sweep learning rate, entropy schedule, BC anchor coefficient, target KL, and games-per-update on a fixed-seed eval set.

13.3 Long-term (6+ months)

1. Additional characters. Silent -> Defect -> Watcher. Each needs new card/relic stats and BC heuristics; the obs encoder schema needs character-specific blocks.
2. Higher ascensions. Ascension 1-20 progression with curriculum-style training.
3. Multi-machine training pool. Replace the manual zip-folder collaboration workflow with a proper rollout server (HTTP or shared filesystem with auth).
4. Cross-platform support. Linux/macOS port for GUI, launcher, and process management.
5. More expressive policy. Replace the 2-layer MLP with a transformer or attention-pooled architecture over the variable-length sub-vectors (hand cards, monsters, choices). Current MLP is fine but capacity-limited.
6. Reward learning. Replace hand-tuned shaping weights with a learned reward model from human play traces or self-play comparisons.

13.4 Risk register

Risk	Likelihood	Impact	Mitigation
BC heuristic ceiling lower than thought	Medium	Medium	Run controlled BC-only baseline eval; compare to PPO checkpoints
PPO catastrophic forgetting	Medium	High	BC anchor loss + target-KL early stop already in place; extend with KL-to-BC penalty if needed
Live-game throughput plateau	High	High	Dependent on headless-simulator availability (medium-term
Reward shaping bias dominates	Medium	High	Reward-correlation analyzer; long-run validation against pure terminal-only signal
Worker fragility on long runs	Medium	Medium	Heartbeat + relaunch already present; add per-worker memory/CPU watchdog

14. Recommended Workflows

14.1 Fresh-restart workflow

For a clean restart from no useful model:

1. Archive prior artifacts: models/, logs/*.csv, logs/*.log, rollouts_shared/*.npz.
2. Run BC with 150-200 games (Mode: BC -> PPO (End-to-End) in the GUI). Resumable checkpointing protects long runs.
3. (Optional) BC -> PPO sanity: add 0-50 PPO games after BC to validate the pipeline before scaling.

4. Switch to Parallel Workers with 4 workers for the main PPO budget (target ~3000 games).
5. Greedy eval every 250-500 PPO games on seeds/eval_200.txt.
6. Plot trends: python scripts/plot_training.py --save logs/training_plot.png. Use rolling avg-100 as the primary signal.

14.2 Hyperparameters: when to change what

Symptom	Likely cause	Adjustment
Avg floor flat for >500 PPO games	Entropy too high, exploration is masking gains	Drop --ent-end from 0.01 -> 0.005
Approximate KL hits target every update	LR too high or entropy too high	Halve --ppo-lr or drop --ent-start
BC anchor loss climbs steadily	Policy drifting from BC distribution	Increase --bc-anchor-coef from 0.02 -> 0.05
Many stale-rollout rejections	Workers reloading too rarely	Decrease worker reload interval
Few wins despite high shaped reward	Reward shaping is misaligned	Run analyze_training_rewards.py; retune low-correlation weights
Worker crashes / hangs	STS / mod instability	Reduce concurrent workers; enable verbose logging

---

15. Metrics Reference

Metric	Source	What it measures	Healthy range
Avg-100 final floor	training_stats.csv	Rolling progression depth	Trending upward over 1000+ games
Lifetime avg final floor	training_stats.csv	Long-run baseline	Slowly climbing
Greedy eval avg floor	eval_stats.csv	Sampling-free progression	Should match or exceed avg-100
Greedy eval win rate	eval_stats.csv	Final outcome rate	Higher than heuristic baseline
Elite / boss win rate	fight_stats.csv	Per-fight outcome	Climbing as policy learns
PPO updates	training_stats.csv	Total update count	Monotonic
Trainer-consumed transitions	training_stats.csv	Total samples used	Monotonic
Policy entropy	training_stats.csv	Sampling diversity	0.05 -> 0.01 over annealing
Value loss	training_stats.csv	Critic fitting quality	Decreasing then plateau
Approximate KL	training_stats.csv	Old -> new policy distance	<0.03 (target-KL)
Clip fraction	training_stats.csv	% steps hitting PPO clip	0.1-0.3
Explained variance	training_stats.csv	Critic predictive power	Climbing toward 1.0
BC anchor loss	training_stats.csv	Distance from BC distribution	Stable, not climbing
Stale-rollout rejection count	train_offline_debug.log	Worker-trainer drift	Low (<10% of files)
Reward / final-floor correlation	analyze_training_rewards.py	Shaping validity	>0.4 strongly preferred
Stuck-state dumps	bug_debug.log	Edge-case frequency	Low / decreasing

For *Slay the Spire*, **avg-25 is too noisy** - one early death or one deep run swings it heavily. Avg-100 is the recommended primary signal.

---

16. Hyperparameter Reference

16.1 BC

Parameter	Default	Notes
--bc-games	50 (suggested 150-200)	Number of heuristic demonstration games
--bc-epochs	30	Supervised training epochs
--bc-lr	1e-3	BC learning rate
--batch-size	64	BC and PPO minibatch size

16.2 PPO

Parameter	Default	Notes
--ppo-games	200	Online RL game budget per session
--ppo-lr	1e-4	PPO learning rate
--gamma	0.995	Discount factor (long-run shaping)
--gae-lambda	0.95	GAE smoothing
--clip	0.15	PPO clip range epsilon
--ent-start	0.05	Initial entropy coefficient
--ent-end	0.01	Final entropy coefficient (annealed)
--target-kl	0.03	Per-update KL early-stop threshold
--bc-anchor-coef	0.02	BC distribution regularizer
--n-epochs	4	PPO epochs per update
--games-per-update	4	PPO updates fire every N completed games
--max-grad-norm	0.5	Gradient clipping

16.3 Network

Parameter	Legacy default	Current offline PPO default
Hidden layers	(256, 256)	(512, 256, 256)
Activation	Tanh	GELU
Optimizer	Adam	Adam
Total parameters	~236,000	~504,000
Migration path	direct checkpoint load	--warm-transfer from compatible older checkpoints

The `PPOTrainer` class still accepts arbitrary `net_arch` and `activation` values. The May 21 offline-training path defaults to the larger GELU architecture because the 12k-game diagnosis showed the 256x256 MLP plateauing around 15 average floor with explained variance roughly flat near 0.78. Warm transfer copies

compatible weight blocks from the old checkpoint and partially initializes widened layers so the policy does not restart from scratch.

---

17. Glossary

Term	Meaning
BC	Behavior cloning - supervised imitation of a heuristic.
PPO	Proximal Policy Optimization - clipped policy-gradient RL algorithm.
GAE	Generalized Advantage Estimation - variance-reduced advantage estimator.
Spawner	Enemy that summons additional minions during combat (Gremlin Leader, Reptomancer, Bronze Automaton).
CommunicationMod	STS mod that exposes game state and accepts actions over stdin/stdout.
SpireComm	Python library that speaks the CommunicationMod protocol.
Action mask	Per-step boolean vector marking which of the 134 actions are legal.
Stale rollout	A rollout produced by a checkpoint too far behind the current trainer state.
BC anchor loss	A regularization term that pulls the PPO policy toward the original BC distribution on demo states.
Target-KL	A threshold on approximate KL divergence used to early-stop PPO updates that move the policy too far.
Avg-100	Rolling average over the last 100 games - the recommended primary learning-progress signal.

---

18. May 21, 2026 Current-System Addendum

This addendum captures the major work after the original May 7 technical writeup. The important shift is that the project moved from "pipeline exists and needs a first long run" to "long-run PPO evidence exists, the Act 1 boss wall is measured, and the next architecture iteration is implemented."

18.1 Commit range covered

The original 0.4 writeup landed at `71964ad` and the PDF layout fix landed at `fc243ed`. Since then, the repository added 33 commits through `5d9edb1` on `main`. The changes fall into these themes:

Theme	Representative commits	Why it mattered
PPO/BC stability	ed17d2b, 82f257e, e8fa0bf, 2842221, 2d9bd02, db01b9d, e4c5dd0	Tracked BC baselines, tuned entropy and BC coefficients, added auto-tuning, normalized stats, and made the GUI expose the controls needed to keep a BC-warm-started PPO run from drifting blindly.
Behavior cloning strength	d6d4bb7, af06ee9, 2670c91	Improved BC throughput and resume behavior, preserved BC anchors, and fixed event-choice handling so the supervised warm start receives cleaner decisions.
Public reporting	6ed45f1, e62483a, c31e02b, 45b6557, 44a5bde, b62c31c, 4fa198e, 4c252a0	Added the static site, dashboard, demo assets, experiment index, and May 18 result snapshot so training claims are inspectable outside raw
Evaluation hardening	e92bb0b, d198ad8, 1df9765, 9010ee1, a69febf, 5c16406	Published long-run evals, made fixed-seed eval resumable/clean, separated eval artifacts into Eval/, and added GUI controls for bounded worker/eval modes.
Long-run process reliability	5433f95, ae4ab51, 60cd5ba	Added game targets, model archiving, restart-every cycling, relaunch-count fixes, and per-instance JVM heap limits to reduce memory growth and stale rollouts.
Policy quality and capacity	b9f244e, f8f93b7, 5d9edb1, ac34dc1	Overhauled heuristic card/event/relic logic, added boss reward shaping and faster BC-anchor decay, upgraded to the (512, 256, 256) GELU model via warm transfer, then added elite win bonus (+3.0) and HP-scaled floor advance to improve Act 1 boss readiness.

18.2 Current training diagnosis

The 5,146-game PPO checkpoint was the strongest published pre-shaping result: 15.44 average floor, 4.03 average reward, best floor 42, 79.9% elite win rate, and 31.1% boss conversion on the 200-game fixed-seed eval. That nearly matched the heuristic average floor of 15.78 but still trailed the heuristic on boss conversion.

The 12,088-game result after boss reward shaping and accelerated BC-anchor decay regressed to 14.83 average floor and 21.7% boss conversion. This is documented as an exploration valley rather than a simple infrastructure failure: the policy loosened from heuristic imitation while the new boss signals had not yet converted into better boss-specific behavior. The key measured bottleneck is still floor 16, where roughly half of deaths occur in the 12k eval.

The 256x256 network appeared to plateau: it reached roughly 15 average floor early and explained variance stayed around 0.78 for thousands of games. The May 21 network upgrade responds to that diagnosis by doubling representational capacity to roughly 504K parameters and switching to GELU while preserving as

much learned behavior as possible through warm transfer.

18.3 Code-level changes to remember

- scripts/ppo_model.py now supports configurable activations and warm_load(), including partial tensor transfer and identity initialization for newly inserted compatible layers.
- scripts/train_offline.py now defaults to --net-arch 512,256,256, supports --activation gelu, --warm-transfer, auto device selection, and richer auto-tune behavior for learning rate, entropy coefficient, and BC coefficient.
- scripts/sts_gym_env.py contains boss-specific reward shaping for Guardian, Hexaghost, Slime Boss, Bronze Automaton, Champ, and Donu/Deca-style priority targets, with boss kill rewards scaled by remaining HP. Includes elite win bonus (+4.0), HP-scaled floor advance (0.50 base + 0.25 x HP ratio), rest-site upgrade reward (+0.30), and Guardian offensive-mode damage bonus (+0.03).
- scripts/screen_handler.py and scripts/behavior_clone.py have newer event, card-tier, boss-relic, grid, and rest-site handling so disabled choices, Coffee Dripper/Fusion Hammer constraints, Match and Keep, and boss relic screens do not poison training data or freeze workers.
- AscensionAI.pyw now includes archive controls, eval-set orchestration, auto-tune and entropy display, bounded worker games, restart-every cycling, and per-instance JVM heap caps.
- docs/dashboard/, docs/experiments/, and docs/index.html are no longer decorative; they are the public evidence layer for comparing BC, PPO, heuristic, and fixed-seed results.

18.4 Near-term operating guidance

1. Treat the current larger GELU network as a transition experiment, not as proof of convergence.
2. Re-evaluate around 15k games using the same fixed seed file before changing reward weights again.
3. Watch boss conversion, floor-16 deaths, normalized entropy, explained variance, and stale-rollout counts together; no single metric is sufficient.
4. Keep auto-tune enabled unless normalized entropy collapses below the healthy band or KL/clip behavior shows unstable updates.
5. Preserve checkpoints and dashboard snapshots before any major reward or architecture experiment so regressions remain explainable.

---

19. May 30, 2026 Headless Cloud Deployment Addendum

This addendum documents moving the training stack off the desktop and onto a **headless Linux cloud VM**, so unattended multi-hour runs no longer occupy a workstation. The architecture is unchanged - the same parallel workers, file-queue rollouts, and offline PPO trainer described in §8 - but the operational substrate is new: a GPU-less server with no display running a GUI-bound, mod-loaded desktop game many times concurrently.

19.1 Deployment target and tooling

Item	Value
Provider / instance	GCP c3-standard-22 spot (22 vCPU, 88 GB RAM, Intel Sapphire Rapids, no GPU)
OS	Ubuntu 22.04 LTS
Provisioning	--provisioning-model=SPOT --instance-termination-action=STOP (preemption stops,
Installer	vm/install.sh - one-shot, idempotent: system packages, Java 8 pin, Python venv + CPU torch, dir layout, validation
Launcher	vm/run_training.sh --workers 8 --hours 12 - N workers + offline trainer
Eval	vm/run_eval.sh - headless fixed-seed evaluation (single or multi-instance)
Sync	vm/sync.ps1 / vm/sync.sh - push code/model/seeds, pull model/logs (gcloud or rsync)

A reviewer goes from a blank Ubuntu image to a running 8-worker job in three steps: push the repo's `vm/` + `scripts/` + `external/`, run `bash vm/install.sh`, copy the Steam game files in, then `./vm/run_training.sh`.

19.2 Headless engineering challenges

Each of the following was an observed failure during bring-up; the fix is encoded in the `vm/` scripts.

#	Challenge	Root cause	Fix
1	No OpenGL context	No GPU and no X display on the server	Per-worker Xvfb virtual display + software GL (LIBGL_ALWAYS_SOFTWARE=1, -Dorg.lwjgl.opengl.Display.a
2	~100x slowdown with multiple instances	OpenGL rendering serializes across windows sharing one X server	One Xvfb display per worker (:99+id)
3	Intermittent SIGSEGV at startup	LWJGL native-library extraction races when JVMs share /tmp	Per-worker -Djava.io.tmpdir=/tmp/sts_worker_<id>
4	Mods silently fail to load	ModTheSpire/BaseMod assume the Java 8 module model; Java 17+ blocks the reflective access they rely	Install and pin Java 8 via update-alternatives
5	Startup crash on audio init	the headless image lacks libopenal.so.1 that LWJGL dlopen's	Install libopenal1, set -Dorg.lwjgl.openal.libname=/usr/lib/x86_64-linux-gnu/libopenal.so.1
6	Worker killed ~10 s after launch	CommunicationMod's READY handshake has a 10 s timeout; the worker's import torch alone exceeds	

it

Write ready\n to the real stdout before any heavy import, then load torch

#	Challenge	Root cause	Fix
7	Silent worker death after ~35 games	JVM heap (-Xmx512m) OOMs over a long session; the worker just disappears, dropping throughput	-Xmx2048m plus --restart-every 25 for a fresh heap
8	All workers share one config	CommunicationMod ignores XDG_CONFIG_HOME and always reads ~/.config/ModTheSpire/	Default each worker --id to its PID so logs/rollouts don't collide
9	Spot preemption mid-run	GCP reclaims the instance	STOP termination preserves the disk; a 30-minute monitor restarts training after the next boot

Challenge #6 is worth emphasizing: the READY timeout is a hard 10 seconds, and a cold CPython process importing torch + numpy can take longer than that, so the obvious "set everything up, then signal ready" ordering fails 100% of the time on a fresh VM. The worker now redirects its own stdout, emits `ready` on the real handle first, and only then performs heavy initialization.

19.3 Throughput and sizing

Under software GL there is no effective frame limiter, so each STS instance is CPU-bound at roughly **2 vCPU for the game loop plus ~0.7 vCPU** for the Python policy/IO bridge. On 22 vCPUs this makes **8 workers** the practical sweet spot: the machine runs near saturation (load ~ core count) without thrashing, sustaining **~90+ games/hour** once heap-stable. Pushing to 10-12 workers oversubscribes the cores and slows every instance, netting no gain. An earlier silent-OOM bug (challenge #7) had masked this, capping the same hardware at ~55 games/hour until heap and restart cadence were fixed.

Machine choice matters more than core count: an `e2-standard-16` (older Broadwell, shared cores) ran the same workload ~2-3x slower per core than the `c3` (dedicated Sapphire Rapids), so the compute-optimized class is worth the marginal spot premium.

19.4 Cost and recovery

Spot pricing is ~$0.19/hr for the `c3-standard-22`, so a 12-hour unattended run costs a few dollars. Converting an existing standard VM to spot is not an in-place operation - it requires detaching the boot disk, deleting the instance, and recreating it as spot with the same disk reattached (`set-disk-auto-delete --no-auto-delete` -> `delete` -> `create --disk=...`). Because `--instance-termination-action=STOP` preserves the disk, the model checkpoint and `training_stats.csv` survive a preemption, and training resumes from the last atomic checkpoint after a `start` + relaunch.

19.5 Code-level changes to remember

- scripts/rollout_worker.py now signals READY before heavy imports and defaults --id to the process PID for multi-worker isolation on the shared CommunicationMod config.
- vm/install.sh is the canonical headless installer (it replaces an earlier vm/setup.sh that incorrectly installed Java 17).
- vm/run_training.sh encodes the per-worker Xvfb display, per-worker JVM tmpdir, 2 GB heap, software-GL env, OpenAL path, and 25-game restart cadence.

- vm/run_eval.sh uses an 8 GB heap so a full 200-game eval does not OOM mid-run.

---

20. June 2, 2026 Learned Deck-Building Addendum

This addendum documents **Path 2**: making deck construction a *learned* skill. Diagnosis of the long Act-1-boss / Act-2 wall pointed at the deck itself - the agent could win Act 1 by brute force but never built a deck that scales, and it died in Act 2. Crucially, the policy *could not* learn deck-building for two structural reasons, independent of reward tuning:

1. It couldn't observe its deck. The encoder only summarized the deck as a 20-d aggregate profile (type ratios, counts, a few key-card flags) - not the actual cards. You can't learn "cut a Strike, keep Demon Form" from aggregates.
2. It didn't control deck-building. Card removal (purge grids) and card upgrades (smith grids) were chosen by a heuristic (screen_handler.pick_grid_card / _pick_grid_upgrade), not the RL policy.

So reward shaping alone was the wrong lever - the gating constraints were **observation and control**.

20.1 Observation: per-card deck count vector (585 -> 717)

Appended a 132-dim count vector over the full Ironclad-encounterable card pool (`game_data.CARD_ID_LIST`: starters, commons/uncommons/rares, colorless, statuses, curses), normalized by 5. Now the policy sees exactly which and how many of each card it holds, not just a profile. The block is appended **last** so warm-transfer keeps the existing 585 features aligned and zero-initializes the new inputs. New obs size: **717**; first-layer params grow ~67.6K (~571K total).

20.2 Control: removal & upgrade moved into the RL policy

`auto_handle_screen` now returns `None` (delegates to the policy) for purge and upgrade grids instead of calling the heuristic, so those become real training decisions. The environment already supported it - `compute_action_mask` masks the grid choices legal and `flat_action_to_spire_action` maps a CHOOSE action to `ChooseAction(choice_index)` - plus a defensive mask fallback when `choice_list` is empty. The BC path keeps the heuristic (`heuristic_all`) so demos still teach sane removals.

20.3 Reward: potential-based deck-quality shaping

The flat per-removal (+0.2) and per-upgrade (+0.30) rewards were a liability once the RL controlled those screens - flat rewards would pay it to purge *good* cards. Replaced with potential-based shaping:

```
Phi(deck) = mean over cards of [ quality(card) * (1 + 0.3 * upgraded) ]
reward    += 1.0 * (Phi_now - Phi_prev)
```

Using the existing per-card `quality` score (?4..+9). Because it's the *mean*, removing a below-average card raises Phi (rewarded) and removing an above-average card lowers it (penalized) - quality-weighted, context-dependent, and symmetric (potential-based shaping doesn't distort the optimal policy). Worked magnitudes: remove a curse +0.31, remove a Strike +0.13, remove Demon Form ?0.27, draft junk ?0.02, upgrade Demon Form +0.13, upgrade a Strike ~+0.015. No deck-size floor - the game already bounds removals.

20.4 Migration, BC anchor, and learning rate

- Warm transfer 585 -> 717 on the main machine (behavior preserved: old model on x equals new model on [x, zeros] to float precision), checkpoint backed up; the trained model continues rather than restarting.
- Light BC anchor: collected fresh 717-d heuristic demos (vm/collect_bc.sh, behavior_clone --collect-only, no net training) into bc_demos_shared/; the offline trainer loads them at a small bc_coef to keep the warm-transferred policy near sane removals while PPO + Phi refine it. Fixed a deadlock where a checkpoint bc_coef=0 caused set_bc_reference to silently discard the anchor.
- --override-lr: the zero-init deck-vector inputs learn slowly at the converged lr (2.37e-5), which --auto-tune always inherited from the checkpoint. Added an override (mirroring --override-ent-coef) to re-raise lr (1e-4, floored at 6e-5) so the new inputs/behaviors learn fast enough to actually test the hypothesis.

20.5 Hands-off self-healing operation

The cloud run is now continuous and self-recovering with no external session (it only stops when `logs/.autorun` is removed): a per-worker watchdog in `run_training.sh` (kill+relaunch a JVM whose log goes silent), a VM-side cron (`vm/monitor.sh`, `.autorun`-driven, 10-min heartbeat to `logs/monitor_heartbeat.log`) that relaunches training after any death/clean-end/reboot, and a Cloud Scheduler job that restarts the VM after spot preemption. A real preemption exposed two bugs since fixed: the BC-anchor `coef=0` deadlock (above) and a cron `pgrep -c || echo 0` that produced `"0\n0"` and disabled the relaunch test.

20.6 Early signal

The deck-vector inputs are integrating: their first-layer weight magnitude relative to the original inputs climbed from ~0.14 to ~0.35 over the first ~1.5 days. Behavior (avg floor ~13, Act 2 reach ~15-17%) has not yet broken from the pre-Path-2 baseline - expected, since the new inputs are still a junior partner and behavior change lags weight growth. A fresh fixed-seed eval will be published once the deck vector matures; if behavior still doesn't move at high weight strength, that will indicate deck-building alone is not the bottleneck.

Document History

Version	Date	Notes
0.1	2026-04-15	Initial writeup
0.2	2026-04-26	Added monster knowledge base, parallel architecture
0.3	2026-05-06	Added BC progress checkpointing, refined sections
0.4	2026-05-07	Full restructure with tables, estimates, ASCII charts, expanded limitations and roadmap
0.5	2026-05-21	Updated current status after 33 post-writeup commits, 12k PPO diagnosis, boss reward shaping, auto-tune maturation, dashboard/eval hardening, and 512/256/256 GELU

warm-transfer upgrade

Version	Date	Notes
0.5.1	2026-05-22	Updated reward weights table with current values, added elite win bonus and HP-scaled floor advance documentation, corrected heuristic vs RL decision boundary
0.6.0	2026-05-23	Expanded observation encoder from 530-d to 585-d with 19 monster power slots (11 new STS1-verified powers). Added REST_UPGRADE_REWARD (+0.30), bumped ELITE_WIN_BONUS to +4.0, added GUARDIAN_OPEN_DMG_BONUS (+0.03). Fixed warm_load() to zero-initialize extra capacity. Added --warm-resume/--net-arch/--activation to train_ppo.py. 16,900+ training games, 1,859 PPO updates.
0.7.0	2026-05-30	Added §19 Headless Cloud Deployment addendum: GCP c3-standard-22 spot VM, one-shot vm/install.sh installer, headless worker/trainer launcher. Documented the nine headless bring-up challenges (per-worker Xvfb + software GL, per-worker JVM tmpdir, Java 8 pin, OpenAL path, CommunicationMod READY-before-import, 2 GB heap + restart cadence, PID worker IDs, spot preemption recovery) and CPU-bound throughput/sizing (~90+ games/hr on
0.8.0	2026-06-02	Added §20 Learned Deck-Building addendum (Path 2): observation 585->717 with a 132-d per-card deck count vector; card removal + upgrade moved from heuristic to RL policy; potential-based deck-quality reward replacing the flat per-removal/per-upgrade rewards; warm transfer 585->717; light BC anchor (vm/collect_bc.sh) + --override-lr; self-healing cloud ops (.autorun cron + heartbeat, Cloud Scheduler preemption auto-restart, per-worker watchdog). Updated abstract, executive summary, and headline metrics to 717-d / ~571K params.