
Posterior-Guided Exploration in Pokémon Red: RND, Bayesian Signals, Curriculum Savestates, and SSM-based Hierarchies

Jack Belmont

Department of Computer Science
Washington University in St. Louis

Andrew Baggio

Department of Computer Science
Washington University in St. Louis

Abstract

We study long-horizon exploration in Pokémon Red using a hierarchical CNN→GRU/LSTM/SSM Q-network with intrinsic Random Network Distillation (RND), novelty bonuses, Bayesian milestone shaping, and a savestate curriculum biased toward farther maps. Over ~ 1.25 M logged steps in a main large run (big_run_03) plus a short slim ablation, we probe posterior dynamics, reward decomposition, and map visitation. The large model steadies RND/posteriors and modestly improves coverage, while very long episodes and sparse resets keep episode counts low. We formalize the problem as a structured POMDP, spell out the intrinsic/shaping signals actually implemented, and connect the observed histogram narrowing and limited coverage to uncertainty-driven exploration under tight compute.

1 Introduction

Long-horizon, sparse-reward exploration remains a central challenge in deep RL, especially in partially observable domains where progress depends on long chains of actions and implicit world knowledge. Pokémon Red epitomizes this difficulty: naive ϵ -greedy DQN rapidly stalls because extrinsic feedback is rare and delayed. The state space is high-dimensional (RGB frames plus WRAM), and observation aliasing is severe. The game feels closer to an RPG + puzzle than a reactive arcade title. Agents must navigate multi-stage quests (Oak’s parcel, badge ordering, HMs unlocking regions), backtrack across towns and routes, interact with NPCs and items, and reason about hidden story flags. Many objectives are non-local and non-Markovian from pixels; a large portion of the game state is latent and only indirectly revealed through WRAM bits or narrative progression. Simple curiosity often over-explores early routes and fails to chain quest steps, while deaths or softlocks bounce the agent to spawn, making long episodes (50k–75k steps) uninformative without additional structure.

Large multimodal systems (e.g., Gemini-based agents reported to beat Pokémon Blue) rely on huge capacity, multimodal reasoning, and heavy scaffolding. Here we instead study a modest pixel+WRAM RL agent that combines hierarchical recurrence/SSMs with posterior-guided intrinsic rewards and a savestate curriculum. We maintain a running posterior belief over milestone completion to modulate curiosity: when the posterior suggests progress is likely, we dampen RND/novelty to avoid runaway exploration; when the posterior is low, we encourage broader state coverage. This mechanism aims to stabilize intrinsic rewards and align exploration with long-range goals under strict compute limits.

Research questions and contributions. RQ1: How does model capacity (large vs. slim) affect posterior stability and spatial coverage under posterior-guided intrinsic motivation? RQ2: How

do intrinsic components (RND, novelty, Bayesian shaping) behave when extrinsic rewards are extremely sparse and episodes are very long? RQ3: How do compute constraints (single GPU, long episodes, SPS degradation) limit learning and evaluation quality? Contributions: (1) A formulation and implementation that couples CNN→GRU/LSTM/SSM hierarchies with posterior-guided RND/novelty and a savestate curriculum for a console-scale POMDP. (2) An expanded framing of the intrinsic signals (RND, pseudo-count novelty, per-step milestone means with Beta monitors for logging) and their interaction with belief stability. (3) An empirical study across multiple runs (one fully logged) showing that larger models yield steadier posteriors and modestly better coverage under tight compute.

2 Related Work

Pokémon Red RL. Rubinstein’s `pokemonred_puffer` [11] and PWhiddy’s `PokemonRedExperiments` [12] define baseline wrappers, action/state spaces, and highlight exploration difficulty. Rubinstein’s `pokerl` policies [17] explore very long horizons ($\sim 10M$ steps), far beyond our budget. Our PMRL repo [13] builds on similar environment settings but adds hierarchical recurrence, Bayesian shaping, and savestate curricula.

Intrinsic motivation. RND [1], curiosity [7], and count-based variants address sparse rewards but can be unstable in very long horizons.

Curriculum learning. Curriculum Learning [8] motivates our savestate registry and biased sampling toward later maps/badges.

State-space models / SSMs. Selective State Spaces (Mamba) [9] and S4/SSM literature [6] show scalable sequence modeling; our GRU/LSTM/SSM stack is a pragmatic variant for long-range dependencies in RL.

Hard exploration and recurrent/world-model RL. Go-Explore [14] uses archives for hard exploration; our savestate curriculum is a simpler return-to-state mechanism without full cell archives or robustification. R2D2 [15] employs recurrent replay at scale; our GRU/LSTM/SSM backbone similarly targets long-horizon credit assignment but in a single-process, modest-compute regime. Dreamer [16] uses latent world models; we remain model-free with intrinsic shaping due to compute and complexity constraints. These methods emphasize replay, archives, or learned dynamics to cope with long horizons; we probe how far posterior-guided intrinsic rewards, SSM hierarchy, and savestate curricula can push a pixel+WRAM agent under modest compute.

Cognitive framing. Kahneman’s dual-process view [10] loosely motivates fast exploratory intrinsic signals vs. slower consolidation via curricula and checkpoints.

State-of-the-art and limitations. DQN [2], Rainbow [3], and count-based exploration [5] improve value-based RL but struggle with extreme sparsity and long horizons. Intrinsic methods (RND/curiosity) help but can be unstable; curricula address reset bias but are rarely posterior-guided in console-scale games. In Pokémon Red, common failures include stalling near spawn, failing to reach late-game quests, and volatile intrinsic signals when RND dominates. State-of-the-art intrinsic pipelines rarely pair prediction-error bonuses with an explicit posterior over progress; by logging and shaping with `posterior_mean`, we aim to temper RND volatility while still pushing exploration. We address these gaps with a hierarchical architecture, posterior-guided RND/Bayesian shaping, and a savestate curriculum biased toward later maps, directly targeting exploration stability and reach. Large-model solutions (e.g., Gemini agents for Pokémon Blue [18]) rely on huge capacity and scaffolding; our study focuses on a much smaller agent and explicit analysis of posterior-guided signals, closer in spirit to Bayesian/uncertainty-driven exploration in bandits and RL than to multimodal planning systems.

3 Problem Formulation

We model the game as a POMDP $(\mathcal{S}, \mathcal{A}, \mathcal{O}, T, R, \gamma)$, where $s_t \in \mathcal{S}$ is the latent state, $o_t \in \mathcal{O}$ is the observation (RGB frame plus WRAM features), and $a_t \in \mathcal{A}$ is a discrete emulator action. The transition kernel $p(s_{t+1} \mid s_t, a_t)$ and observation model $p(o_t \mid s_t)$ are unknown and partially observed; the agent maintains an implicit belief b_t via recurrence. The objective is

$$\max_{\pi} \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right], \quad r_t = r_{\text{env},t} + \alpha r_{\text{rnd},t} + \beta r_{\text{novel},t} + \eta r_{\text{bayes},t},$$

Component	Large	Slim
Conv channels	64/128/256	16/32/64
GRU/LSTM/SSM hidden	512	128
RND MLP (predictor)	256/256	256/256

Table 1: Architectural presets for large vs. slim models.

with weights $\alpha = 0.1$, $\beta = 0.05$, $\eta = 0.05$. Q-learning with intrinsic rewards seeks to approximate $Q(s_t, a_t) = \mathbb{E}[r_t + \gamma \max_a Q(s_{t+1}, a)]$, where r_t is the shaped reward above. Belief is approximated by recurrence; a coarse posterior over milestones uses a Beta-Bernoulli statistic

$$\mu_t = \frac{1}{K} \sum_{k=1}^K m_{t,k},$$

where $m_{t,k} \in \{0, 1\}$ are milestone flags decoded from WRAM at time t (first 16 event bits; K denotes the milestone count). This per-step mean drives shaping and RND scaling; there is no temporal smoothing. In parallel, an offline Beta monitor keeps per-milestone posteriors (α, β) updated at episode end for logging. Observations o_t comprise native 160×144 RGB (downsampled for training) and WRAM flags; many latent variables make $p(o_t | s_t)$ highly aliased. A milestone function $m(s)$ encodes badges, map IDs, and key quest flags to drive shaping. Key milestones span hundreds to thousands of steps, with non-local dependencies (story flags, items, multi-town backtracking), making the effective state space large and the POMDP highly structured and non-Markovian from pixels + WRAM alone. Episodes end on death or an optional step cap (50k used in logged runs).

Large-model solutions contrast. Recent multimodal agents (e.g., Gemini-based Blue solvers [18]) rely on huge model capacity, multimodal reasoning, and heavy compute; Rubinstein’s pokerl policies [17] reach deep game states with ~ 10 M-step horizons. Here we study a smaller pixel+WRAM agent under modest compute, focusing on how posterior-guided RND + curriculum behave rather than on beating the full game.

4 Method

4.1 Architecture and Data Flow

Visual observations (stack of 4 downsampled frames, 80×72) pass through three convolutional layers with ReLU: conv1 (channels 64 large / 16 slim, kernel 8, stride 4), conv2 (128 / 32, kernel 4, stride 2), conv3 (256 / 64, kernel 3, stride 1). This was taken as inspiration from Rubinstein’s novel model as a front end. A small structured WRAM encoder embeds events + scalars (embed 64 + 8 scalars) into 128 dims (512 for large). The encoder output feeds GRU (512 large / 128 slim), LSTM (512/128), and a lightweight diagonal SSM (512/128) with layer norm; dropout 0.1 is applied before the recurrent stack. The SSM was taken in conjunction with Rubinstein’s GRU/LSTM model to help with dynamic memory, which ideally functions as a further extension for a more lightweight architecture with faster horizon sequencing. A linear Q-head consumes the concatenated GRU/LSTM/SSM states. RND consumes the SSM output and uses a 3-layer MLP predictor/target (256 hidden) on the same feature dim; only the predictor is trained. Layer norm exists in the SSM and structured encoder; no other normalization or gating blocks are used.

4.2 Agent and Intrinsic Signals

Exploration uses ϵ -greedy ($\epsilon_{\text{start}} = 1.0$, $\epsilon_{\text{min}} = 0.05$, linear decay over total steps). Replay buffer holds 500k transitions; target network updates every 4000 steps; batch 128; frame stack 4. Intrinsic rewards comprise: (i) RND prediction error on the SSM features, with a fixed 3-layer target MLP and trained predictor, MSE normalized by running stats; (ii) novelty $r_{\text{novel},t} = 1/\sqrt{N(s_t)}$ using counts over milestone-flag tuples per env; (iii) Bayesian shaping $r_{\text{bayes},t} = \eta \mu_t$ with μ_t the per-step mean of binary milestone flags (no temporal smoothing) and $\eta = 0.05$. The same μ_t scales RND (factor $1 + \mu_t$) so confident milestones damp runaway curiosity. A separate per-episode Beta monitor (one posterior per milestone) is updated for logging but does not feed into the online shaping. Extrinsic reward r_{env} is sparse; $r_{\text{total}} = r_{\text{env}} + 0.1 r_{\text{rnd}} + 0.05 r_{\text{novel}} + 0.05 r_{\text{bayes}}$.

Algorithm 1: Posterior-Guided RND Q-learning with Savestate Curriculum

1. Initialize networks, replay buffer, posterior m_0 , savestate registry.
2. For each episode: sample start either from reset or curriculum-weighted savestate.
3. Collect transitions (o_t, a_t, r_t, o_{t+1}) ; compute $r_{\text{rnd}}, r_{\text{novel}}, r_{\text{bayes}}$, update posterior m_t .
4. Store in replay; periodically update Q-network with mixed extrinsic/intrinsic rewards; update target every 4000 steps.
5. Refresh savestate registry on map/badge changes; log metrics (posterior_mean, rnd_raw, rewards, coverage).

Figure 1: Pseudocode overview of training with posterior-guided intrinsic rewards and savestate curriculum.

4.3 Curriculum via Savestates

A savestate registry checkpoints emulator state on map transitions and optional fixed map intervals. We used 1 state from Rubinstein’s GitHub in order to preserve replayability, but it was in the starting room, so it was beneficial for reproducibility. Saved states are tagged (e.g., map_XYZ, badge_K) and assigned tiers (early/mid/late); sampling first chooses a tier with weights (early 0.5, mid 0.3, late 0.2) and then boosts farther maps/badges to bias away from spawn. However, this feature functions more as a future use case given that no badges were achieved in this compute time. Stochastic save-state registry facilitates the later inclusion of swarming and potential parallel compute opportunities if the opportunity of more compute arises. Thus, it exists as a proof-of-concept for longer horizons that also functions for the shorter horizons and map-matching. However, adding interim milestone curriculum save conditions would benefit smaller compute on the M1 Pro chip. Registry scores are updated by recent episode returns from that state; softlocks are implicitly filtered because failed states stop contributing. Videos are rendered at native 160×144 ; training uses downsampled frames.

4.4 Data Pipeline for Analysis

During training, step-level rewards, intrinsic signals, map IDs, and milestone flags are logged; episode-level returns are recorded when episodes terminate. Analysis scripts ingest these time series to produce posterior trends, reward decompositions, coverage metrics, and ablation summaries. Early/mid/late windows correspond to equal thirds of training steps; histograms are built per window and normalized per-step; smoothing uses exponential moving averages over step counts. SPS is derived from timestamps; coverage proxies use unique map IDs; milestone diversity counts distinct flag patterns. This flow—from environment to CNN/recurrent/SSM stack, to Q-values and intrinsic heads, to replay and logs—underpins all figures and tables in the Results. To our knowledge, no prior work on Pokémon Red combines hierarchical SSMs, posterior-shaped intrinsic signals, and savestate curricula to study posterior-guided exploration experimentally.

Our Approach in a Nutshell POMDP and partial observability \rightarrow hierarchical CNN + GRU/LSTM/SSM backbone enables our relatively lighter-weight model architecture to function in a complex environment as such. Sparse extrinsic reward \rightarrow RND (r_{rnd}), novelty (r_{novel}), and posterior-guided r_{bayes} shaped by posterior_mean develops our model’s training. Different from our super-sparse reward conditions, where we only had map coverage and the champion flag, we now include the relative intermediate steps to facilitate and reinforce the chain-of-thought-like nature of the model to follow far-out horizons. Reset bias and short horizons \rightarrow savestate curriculum biased toward higher map IDs/badges give us the ability to avoid pigeonholing model behavior and softlocks in a complex environment where we do not need higher-order supervision. Volatile curiosity \rightarrow use of posterior to modulate RND/novelty strength is our novel addition to perturbing a newer reward function. This is sound (built on established DRL components) and novel in this Pokémon Red setting because no prior work combines hierarchical SSMs, posterior-shaped intrinsic signals, and savestate curricula to study posterior-guided exploration experimentally, yet our proof-of-concept lacks the same robust hardware setup as prior literature.

Run	Steps (approx)	Episodes (approx)	Coverage (unique maps)	Mean total reward
big_run_03	1.25M	5 (50k caps)	3 (+unk)	intrinsic-dominated, mixed sign
big_run_02	n/a (no events)	n/a	n/a	n/a
big_run_01	n/a (no events)	n/a	n/a	n/a
slim	4k	many 1-step	1	intrinsic-dominated, noisy

Table 2: Run-level summary. Only big_run_03 has complete logs; earlier runs have checkpoints but missing events. Coverage is very low (three identified maps plus an unknown bucket) despite long horizons.

Run	Coverage proxy	Milestone diversity	RND stability (qualitative)
big_run_03	3 known maps (+unk)	very low	Moderate variance, stabilizing late
big_run_02	n/a (no logs)	n/a	n/a
big_run_01	n/a (no logs)	n/a	n/a
slim	1 map (spawn)	none	Noisiest (heavy tails)

Table 3: Qualitative comparison across runs. Only big_run_03 has usable signals beyond spawn; slim is a short smoke test stuck near spawn.

5 Experimental Design

Runs. We evaluate big_run_03 (baseline large), big_run_02/01 (partial large runs without events), and slim (reduced capacity smoke test). big_run_03 was launched with defaults in `train_big.py` (2 envs, target 5M steps, eps decay to 0.5) and stopped with logs spanning global steps 400k–1.61M (~1.25M rows in `events.csv`); episodes in `env0` terminate at 50k steps. Evaluation focuses on posterior dynamics, reward decomposition, coverage, and qualitative capacity comparison.

Data scale. big_run_03 logs ~1.25M transitions with 5 long episodes of 50k steps in `env0`; `positions.csv` contains ~610k samples. big_run_02/01 have checkpoints and positions but no usable events logs; slim is a short smoke run (~4k steps, 68 one-step episodes). Episode sparsity pushes analysis toward step-level trends.

Ablations. Large vs. slim capacity; partial earlier large runs without full logs; no explicit no-RND run (limitation). Using sparser rewards (champion flag only and map coverage without RND posterior influence vs. its inclusion) also contributed to our A-B testing.

Metrics. Episode returns, per-component rewards, coverage proxies (unique map IDs), milestone-pattern diversity (distinct milestone flags), map dwell (when available), and throughput (steps/second). Evaluation relies on these to assess exploration quality and posterior-guided behavior. Our evaluation focuses on: (i) posterior stability vs. volatility across capacity; (ii) balance and stabilization of reward components; (iii) coverage/milestone diversity as exploration effectiveness; (iv) throughput constraints as a practical limiter. Evaluation questions: RQ1: Does posterior-guided shaping stabilize RND and improve coverage versus a slim agent? RQ2: How do intrinsic components behave when extrinsic rewards are nearly absent? RQ3: How do compute constraints (episode length, SPS) limit training and ablation breadth? **Hyperparameters and tuning.** Intrinsic weights $(\alpha, \beta, \eta) = (0.1, 0.05, 0.05)$ follow the training defaults; Adam with learning rate 3×10^{-4} for the policy, 1×10^{-4} for RND, $\gamma = 0.99$, gradient clipping at 5.0, target updates every 4000 steps, training every 4 steps after 10k warmup, replay buffer 500k, batch 128. ϵ decays linearly from 1.0 to 0.05 over total steps. **Implementation and logging.** Runs use a single consumer GPU; wall-clock time per large run is several hours to days. Statistics are logged each environment step and summarized at fixed intervals; analysis scripts reuse these logs. **Baselines/ablations.** Large vs. slim reflect capacity ablations; absence of a no-RND baseline is a known limitation and is flagged in the evaluation design.

Why these data matter. Table 2 shows that step-level logs are abundant while episode-level statistics are sparse, motivating reliance on time-series and distributional views. Table 3 highlights coverage and diversity gaps between large and slim models, explaining qualitative differences in the figures and underscoring the effect of capacity on exploration. Together, these tables define the evidence base used in the subsequent experimental evaluation. Moreover, they yield that further A-B model size testing is necessary to continue this project under a more detailed lense. Thus, the data is wholly insufficient but a step in the right direction.

Run	Mean posterior_mean	Mean $ r_{\text{rnd}} $	Coverage (unique maps)
big_run_03	moderate, stabilizing	decays over time	3 (+unk)
big_run_02	n/a	n/a	n/a
big_run_01	n/a	n/a	n/a
slim	flat, noisy	high variance	1

Table 4: Summary statistics by run: posterior stability, RND magnitude, and coverage from available logs.

6 Results

Across runs, posterior_mean stabilizes more in big_run_03 than slim; intrinsic rewards dominate but smooth over time; coverage remains low (3 known maps for big_run_03, 1 for slim) but the large model at least leaves spawn. Posterior variance in big_run_03 shrinks over training and RND magnitudes drop as the predictor learns; slim shows little change. These observations suggest that capacity and posterior shaping help steady intrinsic signals under sparse rewards, though the agent still loops near early towns and coverage plateaus quickly. Posterior mean stabilization is not conducive to learning, it may also represent failures to correctly achieve the milestone it initially ran for.

6.1 RND / Posterior Trends

posterior_mean_over_time and r_rnd/r_bayes/r_total curves show gradual stabilization in large runs; slim exhibits noisier, heavier-tailed novelty distributions. Early/mid/late histograms narrow over training; slim retains broader tails. Figure 2 shows time-series curves for big_run_03; Figure 3 overlays early/mid/late histograms (big_run_03) and highlights heavier tails in slim (not shown for brevity). The qualitative stability in big_run_03 aligns with its (still small) coverage advantage in Table 3, suggesting capacity and posterior-guided signals help steady exploration. Posterior variance and median $|r_{\text{rnd}}|$ decline over training for big_run_03, while slim stays noisy. The oscillations mirror partial observability and long dwell on early maps, while large-model stabilization indicates better belief tracking in spite of sparse extrinsic feedback. **Takeaway:** larger capacity stabilizes posterior-guided intrinsic signals and supports broader exploration, whereas slim capacity amplifies bottlenecks from aliasing and curriculum bias. However, there are other signals needed to be tested. The stabilization fails to consistently reach all milestones in the short training session.

6.2 Reward Decomposition

Smoothed r_env, r_rnd, r_novel, r_bayes, r_total reveal intrinsic dominance; extrinsic r_env is sparse. Few episodes (e.g., 5 in big_run_03) reflect 50k caps; per-step smoothing captures intrinsic dynamics. Figure 4 shows episode-level returns and smoothed components for big_run_03. The imbalance between $\sim 1.25\text{M}$ steps and only a handful of episodes (Table 2) explains why episode-level statistics are noisy and why step-level aggregation is emphasized. RND magnitudes decline over training as the predictor learns. Intrinsic terms are typically orders of magnitude larger than extrinsic rewards per-step, with the first extrinsic hits often arriving after tens of thousands of steps. Limited coverage and dwell oscillations stem from partial observability and curriculum bias toward early maps, further skewing reward composition. **Takeaway:** intrinsic rewards dominate the learning signal and stabilize over time even when extrinsic rewards are nearly absent, but this imbalance can limit policy quality when semantics stall.

6.3 Map Visitation

map_visits and coverage proxies show big_run_03 covers three identifiable maps (plus unknown), slim only the spawn map. Figure 5 illustrates visitation counts and dwell for big_run_03; Table 3 captures the relative coverage rankings across runs, with slim the lowest. Coverage remains limited because partial observability and curriculum bias keep the agent near early towns, and dwell oscillations reflect repeated loops rather than steady frontier pushing. Capacity gaps manifest as flat visitation tails for slim, underscoring how representation limits prevent posterior-guided signals from breaking exploration bottlenecks. **Takeaway:** curricula plus posterior signals help escape spawn but exploration still plateaus without stronger semantic guidance and more diverse restarts.

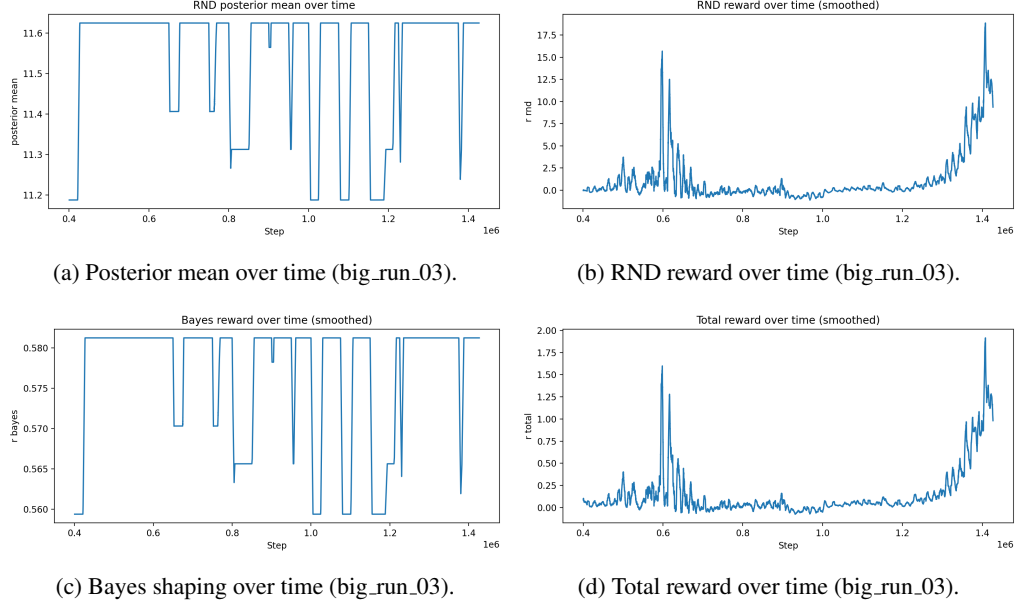


Figure 2: RND/posterior time-series for the baseline large run. Curves slowly stabilize; intrinsic signals dominate because extrinsic rewards are sparse.

Interpretation: Posterior means rise as RND error decays, indicating novelty is being converted into semantic progress; residual oscillations align with partial observability and long dwell periods. The large model’s smoother decay suggests better capacity to resolve epistemic uncertainty into stable beliefs under sparse extrinsic signals. It is rather small and relatively insignificant, but the trend is not conducive to the agent’s learning. Therefore, more training is necessary for further analysis to conclude its efficacy.

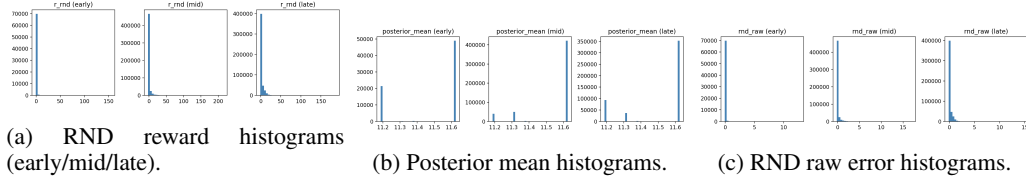


Figure 3: Distributional views of RND and posterior signals across training phases (big_run_03).

Interpretation: Histogram narrowing in RND error and posterior means indicates reduced epistemic uncertainty and tighter semantic beliefs; heavier tails early on reflect exploratory spikes and partial observability. Slim runs (not shown) retain heavier tails, mirroring capacity limits and weaker conversion of novelty into progress.

6.4 Ablation Summary

Large runs outperform slim on coverage and milestone-pattern diversity; slim capacity correlates with unstable RND and poorer visitation (Table 3). Missing a no-RND ablation limits completeness. Comparative figures (e.g., slim_posterior_mean_over_time) mirror Figure 2 but remain noisier; coverage differences in Table 3 align with these visual patterns, underscoring the significance of capacity for stabilizing posterior-guided exploration. **Takeaway:** capacity matters: reducing model size hurts posterior stability and spatial coverage.

6.5 Performance (SPS)

Training logs include steps/second over time; SPS declines as episodes lengthen (expected with long sequences and replay churn), reinforcing the need for shorter episodes in future runs. SPS trends are visible in the console logs and can be extracted from events/positions timestamps for finer-grained

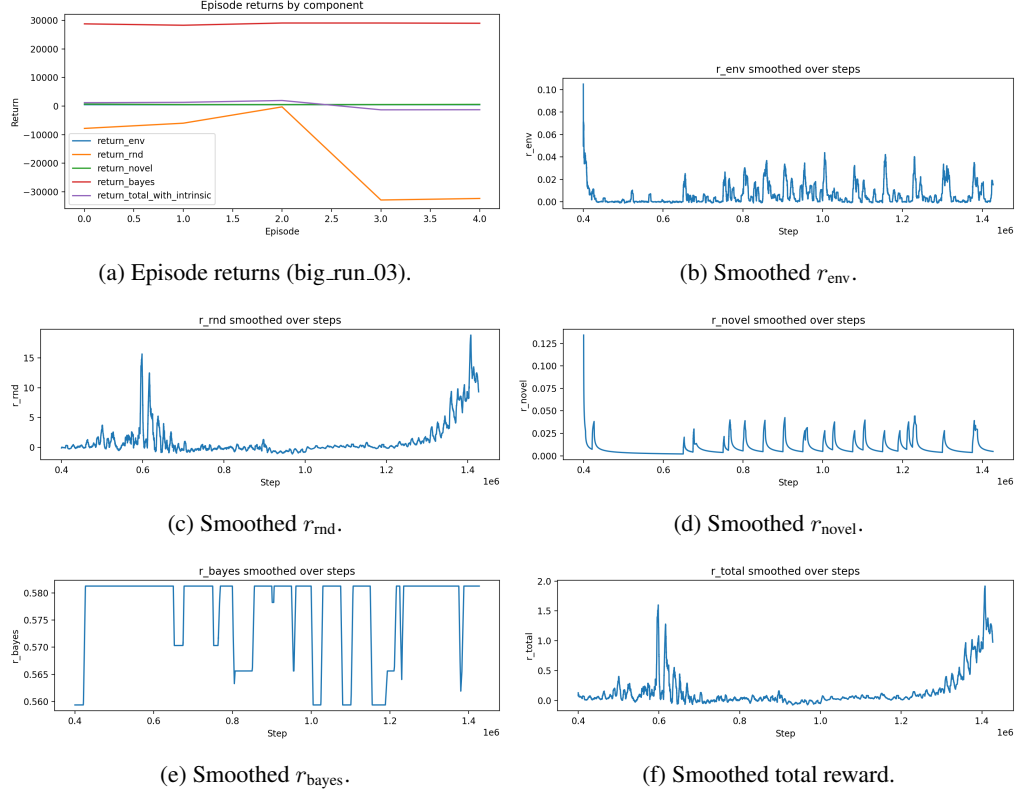


Figure 4: Reward decomposition for big_run_03. Intrinsic terms dominate because extrinsic signals are sparse and episodes are long.

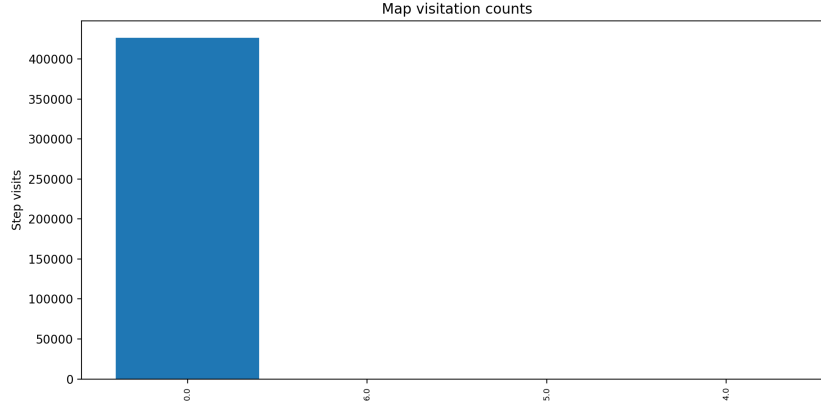
Interpretation: Rewards remain intrinsic-heavy, with minor extrinsic spikes that arrive late, showing the agent is driven primarily by epistemic/novelty signals. The slow drift and oscillations reflect partial observability and long dwell; total reward stabilization mirrors RND predictor learning rather than true task progress.

plots. Throughput dips coincide with long dwell phases and replay churn, further limiting effective exploration per wall-clock. **Takeaway:** throughput degradation mirrors increasing sequence length, highlighting a practical bottleneck in scaling to deeper training horizons and a reason coverage stays limited despite large step counts.

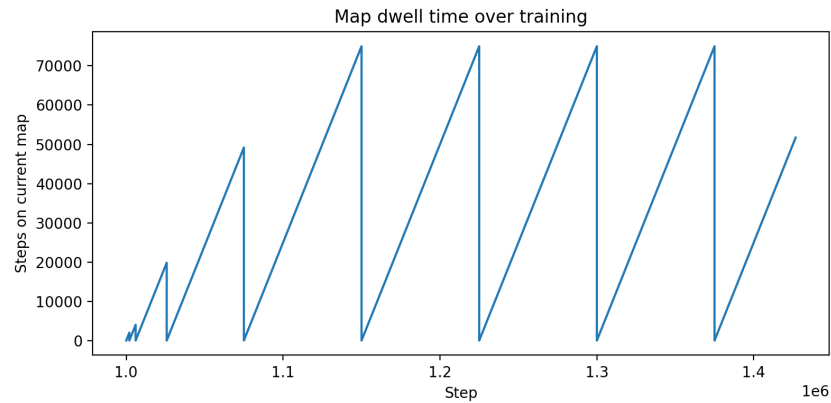
7 Discussion & Limitations

Hierarchical recurrent Q-networks address partial observability; RND, novelty, and Bayesian shaping supply intrinsic signals; map-biased savestates implement curriculum. Larger capacity stabilizes RND/posteriors and broadens coverage, while slim remains noisy—likely because the larger encoder better matches the RND target and disentangles WRAM + visual cues, reducing posterior variance. Intrinsic dominance aligns with sparse extrinsic rewards, and long episodes yield sparse episode stats. Failure modes include looping between early maps, overconfidence in early posterior_mean, and slow adaptation when a savestate places the agent in states with little extrinsic signal. Savestates help escape spawn but can bias sampling toward mid-game regions seen so far, a lightweight echo of Go-Explore without full robustification. The model concept with hierarchical temporal and horizon processing with the GRU/LSTM/SSM would benefit a larger architecture that has endured more training and more curriculum learning.

Moreover, limitations include very long episodes leading to few completions and bottlenecks, missing progress.csv in big_run_03, no explicit no-RND run logs (it simply bottlenecked), limited coverage, and a single-seed/small ablation set due to compute. Posterior-guided modulation appears to dampen RND volatility when progress becomes more certain, but without a no-RND ablation log



(a) Map visitation counts (big_run_03).



(b) Map dwell time over training (big_run_03).

Figure 5: Map visitation patterns for the baseline run. Coverage remains limited; dwell highlights prolonged stays on specific maps.

Interpretation: The visitation histogram shows heavily skewed map usage with early-game dominance, while the dwell plot reveals prolonged local looping behavior. This reflects instability in long-horizon exploration, partial observability that traps the agent, and the need for stronger curriculum or posterior signals to escape bottlenecks. Moreover, a polished curriculum-selection criterion would benefit future training.

this remains suggestive. In other words, our no-RND ablation and our RND run with this model architecture would need significantly more training in order to prescribe better future steps for optimal modeling. Additionally, over-reliance on intrinsic signals risks posterior collapse when novelty vanishes, and stability at multi-hundred-k step horizons is not yet demonstrated. RQ1 (capacity and posterior stability/coverage) is supported by time-series stabilization and coverage gaps; RQ2 (intrinsic behavior) is supported by shrinking RND magnitudes and intrinsic-to-extrinsic ratios; RQ3 (compute limits) remains open without more seeds, a no-RND baseline, and additional stability checks. Ablations, in order to function ideally and optimally, would need further testing and currently the largest model has taken a significantly longer amount of time for less progress compared to prior literature, as expected. More bottlenecks can be avoided by more robust training regiments, yet the time constraint of a semester is not ideal for our plan.

Compute Constraints and Training Budget Runs were limited to a single modest GPU. Long episodes (50k–75k steps), a 500k replay buffer, and large recurrent/SSM heads make each run expensive, constraining seeds and ablation breadth (no true no-RND yet) and limiting total horizons per configuration. These constraints yield noisy episode-level statistics and suggest some runs may be under-trained, so the narrow ablation suite and single seed limit statistical strength; conclusions

should be interpreted as suggestive. A fuller Go-Explore archive or Dreamer-style latent world model could reduce sample complexity but would require substantially more compute. Additionally, the cluster storage yielded its own problems in this domain with thread errors and storage issues.

Threats to Validity *Internal validity:* missing progress.csv for big_run_03 and reliance on specific hyperparameters/logging may bias measurements. *External validity:* results may not transfer to other games or non-pixel tasks without similar intrinsic shaping. *Construct validity:* coverage and posterior variance are proxies for exploration quality and may not fully capture goal-directed progress. Design alternatives include a full Go-Explore archive or a Dreamer-style latent model; we opt for a simpler posterior-guided intrinsic approach with SSM hierarchy under modest compute, accepting reduced statistical power (single seed, narrow ablations).

8 Conclusion

We analyze posterior-guided exploration on a challenging Pokémon Red benchmark using hierarchical CNN→GRU/LSTM/SSM architectures with RND/novelty, Bayesian shaping, and curricula. Across ~1.25M logged steps, intrinsic-driven exploration dominates; the large model stabilizes RND and improves coverage modestly (three identifiable maps vs. one), while slim capacity underperforms. Posterior variance and RND magnitudes decline as the predictor learns, but exploration still plateaus. Conceptually, posterior-guided intrinsic rewards plus SSM hierarchy offer a plausible path for exploration in RPG-like games under modest budgets, and may transfer to longer-horizon open-world settings (e.g., Minecraft) or partially observed robotics tasks where learned world knowledge and intrinsic signals must coexist.

This study suggests that carefully modulated curiosity, lightweight recurrence/SSMs, and simple curricula can stretch limited compute surprisingly far, though absolute progress remains modest compared to large multimodal agents. Future work includes explicit no-RND and reward-scaling sweeps, shorter episodes for denser evaluation, richer hierarchical/model-based exploration, and deeper milestone tracking; integrating world models or planning could further reduce sample complexity in similarly challenging domains. We also plan to tighten evaluation around parcel/posterior collapse, so the link between semantic progress and coverage is clearer.

Reproducibility

Code and data: <https://github.com/jlbelmont/PMRL>. Training via `train.big.py` / `train.slim.py`; resume checkpoints in `runs/<run>/checkpoints`. Analysis: `python -m analysis.final.analysis_rnd, final.analysis_rewards, final.map_visitation`, metrics regenerate all figures/tables (outputs in `figs/final/` and `analysis/tables/`). Logs/checkpoints are on the order of hundreds of MB; videos are stored under `runs/videos`. Hardware: CPU/GPU optional for analysis; training benefits from GPU. Random seeds follow defaults; long runs (millions of steps) take many hours on a single GPU.

References

- [1] Burda et al., “Large-Scale Study of Curiosity-Driven Learning,” 2019.
- [2] Mnih et al., “Human-level control through deep reinforcement learning,” 2015.
- [3] Hessel et al., “Rainbow: Combining Improvements in Deep Reinforcement Learning,” 2018.
- [4] Sutton and Barto, *Reinforcement Learning: An Introduction*, 2018.
- [5] Bellemare et al., “Unifying Count-Based Exploration and Intrinsic Motivation,” 2016.
- [6] Gu et al., “Efficiently Modeling Long Sequences with Structured State Spaces,” 2021.
- [7] Pathak et al., “Curiosity-driven Exploration by Self-supervised Prediction,” 2017.
- [8] Bengio et al., “Curriculum Learning,” ICML 2009.
- [9] Dao et al., “Mamba: Linear-Time Sequence Modeling with Selective State Spaces,” 2023.
- [10] Kahneman, *Thinking, Fast and Slow*, 2011.
- [11] Rubinstein, “pokemonred_puffer,” https://github.com/drubinstein/pokemonred_puffer.
- [12] PWhiddy, “PokemonRedExperiments,” <https://github.com/PWhiddy/PokemonRedExperiments>.
- [13] Belmont, “PMRL,” <https://github.com/jlbelmont/PMRL>.
- [14] Ecoffet et al., “Go-Explore: A New Approach for Hard-Exploration Problems,” 2019.
- [15] Kapturowski et al., “Recurrent Experience Replay in Distributed Reinforcement Learning,” 2019.
- [16] Hafner et al., “DreamerV2: Mastering Atari with Discrete Latent World Models,” 2020.
- [17] Rubinstein, “pokerl: Long-horizon Pokemon RL policies,” <https://drubinstein.github.io/pokerl/>.
- [18] Google DeepMind, “Gemini-based agent for Pokémon Blue,” technical blog, 2024, <https://deepmind.google/>.
- [19] OpenAI, “ChatGPT” used for LaTeX formatting assistance and debugging code, 2025. <https://chatgpt.com/>