Strategic Research Report: Reinforcement Learning Control for the Esper Morphogenetic Framework

Executive Summary

The engineering of morphogenetic neural networks—systems that grow, evolve, and prune their own topology during training—represents a frontier in AutoML. The **Esper** framework, which utilizes a lifecycle of "seed" modules (germination, training, blending, fossilisation) managed by a controller, presents a uniquely challenging Reinforcement Learning (RL) environment. This report provides a comprehensive, actionable strategy for training a Proximal Policy Optimization (PPO) agent to replace the heuristic controller, **Tamiyo**, and effectively manage the non-stationary dynamics of the host model's training process.
Our analysis characterizes Esper not merely as a standard control task, but as a **non-stationary, long-horizon meta-optimization problem**. The agent does not control a static physics engine; it steers the loss landscape of a learning algorithm. This introduces profound challenges in credit assignment, observation normalization, and reward shaping. This document details the system dynamics, probable failure modes, a three-stage curriculum, a specialized "Esper-Transformer" policy architecture, and a rigorous experimental plan. The strategy leverages state-of-the-art techniques in **Set Transformers** for permutation-invariant slot processing, **Adaptive Reward Annealing** to balance rent-seeking versus contribution, and **Autoregressive Action Decoding** to handle factored constraints.

A) System Understanding and Control Dynamics

To engineer a robust policy, we must first rigorously reconstruct the control loop and define the Markov Decision Process (MDP) governing the interaction between the PPO agent and the Esper morphogenetic substrate.

1. Reconstruction of the Control Loop

The interaction between the RL agent and the Esper environment operates on a timescale distinct from the atomic steps of the supervised training loop. Unlike robotic control where Δt is milliseconds, here Δt represents a significant computational quantum (e.g., one epoch or a fraction thereof).

- **The Environment Step (env.step):**
  - The environment advances the *host* model's training by a fixed interval (e.g., one epoch). During this interval, the underlying supervised learning optimizer (SGD

    with momentum) updates the weights of the host and any active seeds.[1]
    - Metrics are accumulated: training loss, validation accuracy, gradient norms, and "counterfactual" contributions of seeds (how much better the loss is *with* the seed than without it).
    - **State Transition:** The state S_t transitions to S_{t+1} based on the *learning dynamics* of the neural network. This is non-deterministic from the agent's perspective, as it depends on the stochasticity of data batching and the complex optimization landscape.[3]
- **The Agent's Role (Meta-Controller):**
    - The PPO agent does not adjust weights directly. It acts as a **structural manager**. It determines *topology* (which blueprint to instantiate), *connectivity* (blending alphas), and *lifecycle* (when to prune or fossilise).[4]
    - **Action Execution:** At state S_t, the agent emits a categorical action vector A_t. The environment executes this operation (e.g., GERMINATE in Slot 3). The physics engine here is the *training logic*: initializing weights, setting alpha masks, or freezing tensors.

2. The True Credit Assignment Horizon

A critical insight for this strategy is realizing that the default PPO hyperparameters (specifically $\gamma=0.99$) are likely insufficient for the "gestation period" of a seed module.

- **The Lifecycle Horizon:** Consider the "Germinate" action.
    - $t=0$: Agent chooses GERMINATE. Immediate reward is likely near zero or negative (action cost/rent).[6]
    - $t=10$: Seed is in TRAINING stage. It consumes compute "rent" but provides no benefit to the host yet.
    - $t=50$: Seed begins BLENDING. It may cause a temporary "alpha shock" (loss spike) as it disturbs the host.[7]
    - $t=80$: Seed fully integrates. The host loss drops significantly. The attribution reward spikes.
    - $t=100$: Seed is FOSSILISED.
- **The Valuation Gap:** The reward at $t=80$ must be credited to the decision at $t=0$. With $\gamma=0.99$, the discount factor after 80 steps is $0.99^{80} \approx 0.44$. While non-zero, this decay combined with the noise of stochastic gradient descent means the signal-to-noise ratio (SNR) for the germination action is incredibly low.[8]
- **Implication:** The "effective horizon" of the credit assignment must match the biological lifecycle of the seeds. We will likely need $\gamma \ge 0.995$ or even 0.999, requiring specialized Generalized Advantage Estimation (GAE) tuning to manage variance.[9]

3. Dominant Variables of Non-Stationarity

Esper is a **Partially Observable Non-Stationary MDP**. Understanding the *nature* of this non-stationarity is prerequisite to solving observation normalization.

- **The Drifting Manifold:** The "physics" of the environment changes over time.
  - **Early Training (Epoch 1-20):** High gradients, rapid loss reduction. "Improvement" rewards are large and easy to obtain.
  - **Late Training (Epoch 100+):** Vanishing gradients, plateauing loss. "Improvement" rewards become sparse and orders of magnitude smaller.[10]
  - **Implication for PPO:** A RunningMeanStd observation normalizer that adapts too slowly will squash late-training features to zero. Conversely, if it adapts too quickly, it masks the fact that the model is converging. The agent must *know* where it is in the training trajectory (time-awareness) to interpret the magnitude of the loss delta correctly.[11]

4. Action Masking vs. Fallback Dynamics

The system prompt notes a critical design choice: *"Invalid actions fall back to WAIT."* This introduces a dangerous **Aliasing** problem in the policy's learning signal.

- **The Problem:** Suppose PRUNE on an empty slot is invalid. The agent selects PRUNE. The environment executes WAIT. The reward received is R_{wait}.
- **The Learning Signal:** The policy updates the probability of PRUNE based on R_{wait}. Effectively, the agent learns that PRUNE is just another way to WAIT.
- **Entropy Waste:** This wastes probability mass on invalid logits. In a 9-head action space, if 50% of combinations are invalid but yield safe WAIT rewards, the agent might converge to a high-entropy "noise" policy where it randomly mashes invalid buttons to safely wait, rather than learning the specific WAIT op.[12]
- **Correction:** We must implement **True Action Masking** where invalid logits are set to $-\infty$ *before* the softmax. This forces the gradients to flow only to valid actions, accelerating convergence significantly.[14]

B) Audit of Likely Failure Modes

Based on the mechanics of PPO and the specific dynamics of morphogenetic networks, we identify 12 probable failure modes. Each represents a distinct risk to the stability or convergence of the Esper controller.

1. The "Rent-Seeking" Inactivity Trap

- **Symptom:** The agent quickly converges to a policy of purely WAIT or PRUNE, refusing to germinate seeds. seed_utilization drops to near zero.
- **Mechanism:** The reward function includes "compute rent" (R_{rent}), a penalty for active slots. R_{rent} is immediate and deterministic. The "contribution reward" ($R_{contrib}$) is delayed, stochastic, and sparse. In the early stages of training, the Value function $V(s)$ fails to capture the long-term return of germination. The agent, being risk-averse, optimizes the immediate objective: minimizing rent.[6]
- **Telemetry:** Frequency of WAIT > 90%; rent_penalty is minimized; contribution_reward is

flat.
- **Mitigation: Rent Annealing.** Disable rent penalties entirely during the first 30% of the curriculum (Stage 1). Ramp them up only after the agent has learned that germination leads to positive contribution.[16]

2. PBRS Cycle Hacking

- **Symptom:** The agent rapidly oscillates seeds between stages (e.g., GERMINATE $\to$ PRUNE $\to$ GERMINATE) without meaningful training.
- **Mechanism:** Potential-Based Reward Shaping (PBRS) provides rewards for transitions: $\Delta \Phi = \Phi(S_{t+1}) - \Phi(S_t)$. Ideally, $\sum \Delta \Phi$ over a cycle should be ≤ 0 (due to γ). However, if the time-steps between transitions are short, or if γ is high, the agent can farm positive potentials. For example, if $GERM \to TRAIN$ gives +1.0 and $TRAIN \to PRUNE$ gives -0.5, the agent will cycle this loop.[17]
- **Telemetry:** High frequency of state transitions; low average seed age; cyclical rew_pbrs in logs.
- **Mitigation:**
  - Ensure Potentials satisfy strictly conservative difference constraints.
  - Enforce "Kasmina gate thresholds" (min training epochs) not just as success gates but as *validity masks*, preventing the agent from pruning a seed immediately after germination.[19]

3. Entropy Collapse in Factored Heads

- **Symptom:** The agent explores Ops well but fixates on a single blueprint or alpha_target prematurely (e.g., always choosing Blueprint A, even when suboptimal).
- **Mechanism:** In multi-head policies with a shared backbone, gradients from the dominant head (usually Op or Slot) dominate the updates. The features learned by the shared encoder optimize for the dominant head, causing the "subordinate" heads (like Blueprint) to lose feature discrimination. Their entropy drops to zero as they piggyback on the deterministic features.[1]
- **Telemetry:** Plot entropy_per_head. entropy_blueprint drops to near zero while entropy_op remains high.
- **Mitigation: Per-Head Entropy Regularization.** Assign distinct entropy coefficients (β) to each head. Specifically, set higher β for high-cardinality heads like Blueprint and Slot to force prolonged exploration.[21]

4. Observation Normalization Instability

- **Symptom:** Performance degrades late in the host training process; gradients explode or vanish; agent behaves randomly in late epochs.
- **Mechanism:** RunningMeanStd calculates cumulative statistics. In supervised training, metrics like train_loss drop by orders of magnitude (e.g., 2.0 $\to$ 0.02). If the normalizer momentum is high, it "freezes" on the large early values. Late-training inputs of 0.02 are normalized to ≈ 0, vanishing the signal. Conversely, if it adapts too

fast, it normalizes the "convergence plateau" to noise.[10]

- **Telemetry:** obs_rms_mean and obs_rms_var plots; Agent entropy spikes in late episodes.
- **Mitigation:**
    - Use **Layer Normalization** in the policy network instead of (or in addition to) input normalization.[23]
    - Feed raw log-features ($\log(loss)$) rather than linear features to compress the dynamic range.

5. Slot Permutation Variance

- **Symptom:** The agent learns to use Slot 1 effectively but ignores Slot 2, even though they are functionally identical.
- **Mechanism:** A standard MLP policy sees the input vector as $[F_{slot1}, F_{slot2}, \dots]$. It must independently learn the physics of Slot 1 and Slot 2. This breaks sample efficiency and leads to permutation bias (preferring Slot 1 because it learned it first).[24]
- **Telemetry:** Heatmap of Action(Slot) usage shows skew toward specific indices; retraining the same policy yields different slot preferences.
- **Mitigation: Permutation Invariant Architecture.** Use a **Set Transformer** or **Deep Sets** architecture to process slot features. Pool them into a global context before decoding actions. This forces the agent to treat all slots as equivalent instances of the class "Slot".[26]

6. Invalid Action Fallback Poisoning

- **Symptom:** Slow convergence; agent learns "superstitious" behavior (e.g., toggling alpha targets to wait).
- **Mechanism:** As detailed in Section A.4, falling back to WAIT for invalid actions aliases the action space. The agent learns that "Set Alpha 0.5 on Empty Slot" is a valid way to Wait. This pollutes the Q-values for the "Set Alpha" action, making it harder to learn its *true* utility when a slot is actually active.[12]
- **Telemetry:** High rate of invalid_actions in logs; Q-values for invalid actions are non-negative.
- **Mitigation: True Action Masking.** Mask invalid logits to $-\infty$. If technical constraints prevent this, apply a **penalty** ($R_{invalid} < 0$) to fallback actions to distinguish them from valid WAIT.[14]

7. Telemetry Noise Overload

- **Symptom:** Training destabilizes when use_telemetry is enabled (adding gradients/health features).
- **Mechanism:** The optional telemetry adds 17 dimensions per slot, including high-frequency noise like gradient norms. In the early stages, this noise is uncorrelated with reward (which is sparse). The policy overfits to these noise patterns, creating "superstitious" correlations that break generalization.[28]

- **Telemetry:** High variance in policy loss; explained_variance of the Value function drops when telemetry is added.
- **Mitigation: Feature Curriculum.** Start training without telemetry. Introduce telemetry features only in Stage 3 (Fine-tuning) or use **Feature Dropout** to prevent over-reliance on noisy signals.[29]

8. Alpha Shock Aversion

- **Symptom:** Agent refuses to BLEND or SET_ALPHA_TARGET, keeping seeds isolated or static.
- **Mechanism:** The "Alpha Shock" reward penalty (convex penalty on alpha deltas) is intended to prevent destabilizing the host. If this penalty is too high relative to the *expected* future contribution, the agent encounters a "valley of death": it must incur a shock penalty *now* to get a contribution reward *later*. If γ is too low or the shock penalty too high, it chooses the local optimum: do nothing.[7]
- **Telemetry:** Alpha values remain static; shock_penalty is 0; contribution_reward is low.
- **Mitigation: Shock Annealing.** Start with shock_penalty_weight = 0. Linearly ramp it up as the agent learns the value of blending.

9. Fossilisation Cliff

- **Symptom:** Agent cycles seeds or Holds indefinitely, rarely choosing FOSSILIZE.
- **Mechanism:** FOSSILIZE is a terminal operation for a seed. It removes the seed (and its future potential rewards) for a one-time "Terminal Bonus." If the discounted future value of "Holding" (even with rent) exceeds the Terminal Bonus, the agent will never fossilise.[30]
- **Telemetry:** stage_occupancy shows accumulation in HOLDING stage; FOSSILIZE op count is near zero.
- **Mitigation: PBRS Tuning.** Ensure the potential difference $\Phi(FOSSILISED) - \Phi(HOLDING)$ is sufficiently positive to outweigh the "opportunity cost" of losing the seed.

10. Partial Observability of Generalization

- **Symptom:** Agent optimizes *training* loss effectively but *validation* loss degrades (Overfitting).
- **Mechanism:** The agent observes train_loss and val_loss. However, the *cause* of generalization (e.g., the geometry of the minima, flatness) is hidden. The agent acts as a meta-optimizer. If it aggressively tunes the host to minimize training loss (e.g., by adding many parameters), it might induce overfitting. PPO is strictly maximizing the reward (which is usually based on validation), but if the reward signal is noisy, it might latch onto training dynamics.[31]
- **Telemetry:** Divergence between train_contribution and val_contribution metrics.
- **Mitigation: Generalization Features.** Explicitly feed (val_loss - train_loss) (Generalization Gap) as a feature. Base rewards strictly on **Validation Improvement**,

never training improvement.

C) Training Curriculum Plan

We propose a **Three-Stage Curriculum** to guide the agent from basic mechanical competence to architectural mastery. This leverages the "Start Small" principle essential for Neural Architecture Search (NAS) and complex control tasks.[32]

Stage 1: The Mechanic (Lifecycle Grammar)

- **Objective:** Learn valid state transitions, action masking, and basic resource management (e.g., "Don't prune what you just planted").
- **Environment Configuration:**
  - slots: 2 (Reduced complexity to speed up credit assignment).
  - max_seeds: Infinite (allow rapid cycling).
  - max_epochs: Short (e.g., 20 epochs). The host resets frequently.
- **Reward Configuration:**
  - **Dense & Shaping-Heavy.**
  - **PBRS:** High weights. Reward the transition $DORMANT \to GERMINATED \to TRAINING$. This teaches the agent *how* to operate the machine.
  - **Rent/Shock:** Disabled (Weight = 0.0). Do not punish exploration.
  - **Invalid Action Penalty:** -0.1 (Strictly enforce rules).
- **Features:** Base + Per-slot (No telemetry).
- **Outcome Criteria:** Agent effectively utilizes slots; Invalid action rate < 1%; Stage transitions follow logical progression.

Stage 2: The Optimizer (Metric Surfing)

- **Objective:** Learn to correlate actions with host metric improvements. Discover that "Germinating" is only valuable if the seed *contributes*.
- **Environment Configuration:**
  - slots: 4.
  - max_epochs: Medium (50 epochs). Allows enough time for seeds to mature.
- **Reward Configuration:**
  - **Contribution-Dominant.**
  - Enable attribution reward (counterfactual gain).
  - Enable rent (low weight, e.g., 0.01) to discourage useless seeds.
  - **Disable PBRS:** Remove the training wheels. The agent must now generate value to get rewards.
- **Features:** Add SeedTelemetry (gradients, health). The agent now has the data to discern *why* a seed is good.
- **Outcome Criteria:** seed_utilization correlates with validation improvement; Agent prunes underperforming seeds.

Stage 3: The Architect (Full Complexity)

- **Objective:** Generalization, long-term stability, and efficiency.
- **Environment Configuration:**
  - slots: Full (e.g., 8 or 16).
  - max_epochs: Full (e.g., 100+ or convergence).
- **Reward Configuration:**
  - **Sparse & Constrained.**
  - High rent and shock penalties.
  - Reward based on *final* test accuracy or Area Under Curve (AUC) of validation accuracy.
- **Features:** Full feature set.
- **Outcome Criteria:** Agent outperforms Tamiyo benchmark; demonstrates novel lifecycle strategies (e.g., deferred blending).

D) PPO Hyperparameter and Architecture Plan

This section defines the PPO configuration, specifically tailored for the "vectorized, shaped reward, multi-head" nature of Esper.

1. The "Esper-Transformer" Policy Architecture

Standard Multi-Layer Perceptrons (MLPs) are structurally insufficient for the "Slot" based state space of Esper. An MLP treats Slot 1 and Slot 2 as distinct features with fixed positions. This prevents the transfer of learning between slots (i.e., learning how to manage Slot 1 doesn't help manage Slot 2).[26]

We propose a **Permutation-Invariant Transformer Encoder** architecture:

- **Inputs:**
  - Global Features ($F_g \in \mathbb{R}^{23}$): Host state.
  - Slot Features ($F_s \in \mathbb{R}^{N_{slots} \times D_{slot}}$): State of each seed.
- **Embedding Layer:**
  - Project $F_g$ to a "Global Token" $T_g$ (dim $d_{model}$).
  - Project each row of $F_s$ to "Slot Tokens" $T_{s_1}, T_{s_2}, \dots, T_{s_N}$ (dim $d_{model}$).
  - Add a learnable "Slot Position Embedding" only if slot *position* matters (unlikely in Esper; likely permutation invariant).
- **Encoder:**
  - Pass the sequence $$ through a **Transformer Encoder** (2-3 layers, 4 heads).
  - **Self-Attention:** Allows the Global token to query the Slots ("Which slot is ready?") and Slots to query each other ("Is another seed blending right now?").
- **Decoders (Action Heads):**

   - **Global Actions (Op, Blueprint):** Decode from the contextualized Global Token T'_g.
      - $Logits_{op} = MLP_{op}(T'_g)$
   - **Slot Selection:** Decode from the Slot Tokens T'_s via a Pointer Network or simple projection.
      - $Logits_{slot} =$
   - **Value Head:** Decode from T'_g.

Rationale: This architecture is mathematically **Permutation Invariant** (regarding the slots). It learns a general "Seed Management Policy" that applies to any number of slots, enabling the Curriculum strategy (scaling from 2 to 16 slots without retraining).[24]

2. Hyperparameter Plan

Parameter	**Initial Value**	**Adaptation Rule**	**Rationale**
learning_rate	3e-4	Linear decay to 1e-5	Standard PPO start; decay ensures stability as the host dynamics settle.[37]
gamma (γ)	**0.995**	Fixed	**Critical:** Standard 0.99 (horizon ≈ 100) is too short for seed lifecycles. 0.995 implies a half-life of ≈ 140 steps, covering the germination-to-fossilisation arc.[8]
gae_lambda (λ)	0.95	Fixed	Balances bias/variance in the long-horizon estimates.
clip_range	0.2	Fixed	Standard PPO stability constraint.
ent_coef (Global)	0.01	Anneal to 0.001	Encourage early exploration.
vf_coef	0.5	Fixed	Balance policy and value loss.
max_grad_norm	0.5	Fixed	Prevents gradients from exploding due to RunningMeanStd instabilities.
n_steps	2048	Fixed	**Long Rollouts:** Essential. We need

			long trajectories to capture delayed rewards. Short rollouts with high gamma introduce truncation bias.[38]
batch_size	64	-	Standard mini-batch size for updates.
n_epochs	10	-	Number of optimization passes per rollout buffer.

3. Per-Head Entropy Strategy

Since the action space is factored (Op, Slot, Blueprint, Alpha), a global entropy coefficient is suboptimal. The agent might maximize entropy on irrelevant heads (e.g., Alpha when Op is WAIT) to satisfy the constraint while collapsing the important heads.

- **Strategy:** Assign specific entropy weights $w_H$ to each head $H$.
  - $w_{slot} = 0.05$: Force exploration of all slots.
  - $w_{blueprint} = 0.05$: Force exploration of architectures.
  - $w_{op} = 0.01$: Allow faster convergence on lifecycle logic.
  - $w_{alpha} = 0.01$.
- **Implementation:** $L_{ent} = \sum_H w_H \cdot H(\pi_H)$.[1]

E) Reward Shaping Audit & Landscape Analysis

We treat the reward function as a non-convex optimization landscape defined by:

$$R_{total} = w_1 R_{contrib} + w_2 R_{PBRS} - w_3 R_{rent} - w_4 R_{shock} + w_5 R_{terminal}$$

1. Landscape Conflicts

- **Rent vs. Contribution (The "Valley of Death"):**
  - $R_{rent}$ is immediate and deterministic (negative).
  - $R_{contrib}$ is delayed and stochastic (positive).
  - **Analysis:** If the agent is initialized randomly, $E \approx 0$. Thus, the gradient of the value function points towards "Kill All Seeds" to maximize $R_{total} \approx -R_{rent} \to 0$. This is a stable local optimum (Failure Mode 1).
- **Attribution vs. PBRS (The "Cycle Trap"):**
  - $R_{PBRS}$ rewards state change.
  - $R_{attribution}$ rewards loss improvement.
  - **Analysis:** If $w_2 R_{PBRS} > w_1 R_{contrib}$, the optimal policy is to cycle

states ($Dormant \leftrightarrow Germinated$) regardless of host performance. The agent becomes a "lifecycle farmer" rather than a "model trainer".[17]

2. Proposed Reward Schedules

To navigate this landscape, we cannot use static weights. We must use **Dynamic Reward Scheduling**.[16]

- **Schedule 1: The "Startup" (Stage 1 Curriculum)**
    - $w_{PBRS} = 1.0$ (High): Guide the agent to valid states.
    - $w_{rent} = 0.0$ (Zero): Subsidize exploration.
    - $w_{contrib} = 0.1$ (Low): Just a signal.
    - **Goal:** Learn the mechanics.
- **Schedule 2: The "Growth" (Stage 2 Curriculum)**
    - $w_{PBRS} \to 0.0$: Remove training wheels.
    - $w_{rent} = 0.01$: Introduce minor cost.
    - $w_{contrib} = 1.0$: Main objective.
    - **Goal:** Learn value.
- **Schedule 3: The "Enterprise" (Stage 3 Curriculum)**
    - $w_{rent} \to 1.0$: Full cost accounting.
    - $w_{shock} \to 1.0$: Enforce stability.
    - **Goal:** Efficiency and precision.

3. Reducing Goodhart's Law

To prevent the agent from gaming the "Counterfactual Contribution" (e.g., by creating a seed that helps *only* because the host was sabotaged by another seed):

- **The "Global Improvement Gate":**
    - Logic: $R_{final} = R_{contrib} \cdot \mathbb{I}(\Delta Loss_{host} < 0)$
    - Mechanism: Even if a seed has a high counterfactual score (i.e., "Host would be worse without me"), it receives **zero reward** if the host's absolute performance did not improve. This forces the agent to align with the global objective, not just the local attribution metric.

F) Observation and Action Representation Improvements

1. Feature Engineering

- **Log-Scaling:** Neural network parameters (weights, gradients) span orders of magnitude. Inputting raw values confuses the normalizer.
    - *Change:* total_params $\to$ $\log_{10}(total\_params)$.
    - *Change:* gradients $\to$ $\log(1 + ||\nabla||)$.
- **Relative Metrics:** Raw loss is unbounded and architecture-dependent.

    - *Change:* loss $\to$ relative_loss = current_loss / initial_loss.
    - *Change:* val_acc $\to$ val_acc_delta = val_acc - best_val_acc.
- **Time-Awareness:** PPO policies are static mappings $S \to A$. They do not know if it is Epoch 1 or Epoch 100 unless explicitly told.
    - *Change:* Add progress = epoch / max_epochs to the global feature vector. This allows the agent to learn "Endgame" strategies (e.g., Fossilise everything at 95% progress).[40]

2. Autoregressive Action Parameterisation

The current "Factored" action space assumes independence between Op and Blueprint. This is false. GERMINATE requires Blueprint; WAIT ignores it. Predicting Blueprint when Op=WAIT adds noise to the gradients.

- **Proposal: Autoregressive (Conditional) Heads**.[41]
    - **Step 1:** The network outputs logits for Op. Sample $op \sim \pi_{op}$.
    - **Step 2:** Embed the sampled $op$ into a vector $E_{op}$.
    - **Step 3:** Pass $$ into the Blueprint and Slot heads.
    - **Mathematical Form:** $P(Action) = P(Op) \cdot P(Blueprint | Op) \cdot P(Slot | Op)$.
    - **Implementation:** This requires a custom PPO distribution class but significantly improves sample efficiency by removing the need for the agent to learn "dummy" correlations for irrelevant parameters.

G) Experiment Plan

We define 12 experiments to systematically isolate dynamics and ascend the curriculum.

Exp ID	**Hypothesis**	**Change / Condition**	**Primary Metric**	**Stopping Criteria**
E01	**Baseline Sanity**	PPO vs Random Policy (Stage 1 Env)	Ep Length, Total Reward	1M Steps
E02	**Masking Efficacy**	True Masking vs Fallback-to-Wait	Invalid Action Rate	Convergence or 2M Steps
E03	**Horizon Test**	$\gamma=0.99$ vs $0.995$ vs $0.999$	Best Val Acc, Seed Age	5M Steps
E04	**Rent Sensitivity**	Rent Coef 0.0 vs 0.1 vs 1.0 (Fixed)	Seed Utilization %	5M Steps
E05	**PBRS Cycling**	PBRS Only (No Contrib Reward)	State Transition Rate	Detect Cycles (Oscillation)
E06	**Telemetry Noise**	With vs Without Telemetry	Value Loss, Exp Variance	5M Steps

		Features		
E07	**Transformer Pol**	MLP vs Set Transformer Arch	Sample Efficiency (AUC)	5M Steps
E08	**Alpha Shock**	Annealed vs Fixed Shock Penalty	Blend Activity	5M Steps
E09	**Curriculum**	Stage 1 $\to$ 2 vs Direct Stage 2	Final Val Loss	10M Steps
E10	**Tamiyo Duel**	PPO (Best Config) vs Tamiyo	Area Under Loss Curve	100 Episodes
E11	**Generalization**	Reward on Train vs Val Loss	Train/Val Gap	10M Steps
E12	**Ablation: Heads**	Independent vs Autoregressive	Entropy per Head	5M Steps

Metric Definitions:

- **Policy Health:** approx_kl, clip_fraction, entropy_op, explained_variance (Target > 0.5).
- **Behavioral:** seed_utilization (Active / Max Slots), action_wait_ratio, fossilization_rate.
- **Task:** final_val_acc, integrated_compute_cost.

H) Instrumentation and Debugging Checklist

To diagnose the "black box" of PPO training on a morphogenetic net, the following must be logged at every step (or aggregated per epoch):

1. Reward Component Breakdown (Stacked Area Chart)

- **Log:** rew_contribution, rew_pbrs, rew_rent, rew_shock, rew_terminal.
- **Diagnosis:** If rew_rent (negative) matches rew_pbrs (positive) and rew_contribution is near zero, the agent is **farming PBRS to pay Rent** without actually training the model (Failure Mode 2).

2. Action Head Distributions (Heatmaps)

- **Log:** p_op (Probability of each Op over time), p_slot (Probability of selecting each slot).
- **Diagnosis:**
  - If p_op becomes 100% WAIT, diagnosis: **Rent-Seeking Trap**.
  - If p_slot shows vertical stripes (e.g., always Slot 1), diagnosis: **Permutation Variance** (Need Set Transformer).

3. Masking and Invalidity

- **Log:** % actions_masked_out, % invalid_actions_attempted.
- **Diagnosis:** If % invalid_actions_attempted > 5% after 1M steps, the agent is struggling

with the rules. Check if fallback penalties are sufficient.

4. Value Estimation Health

- **Log:** value_target vs value_prediction, explained_variance.
- **Diagnosis:** If explained_variance < 0, the Critic is worse than a random guesser. This usually implies **Non-Stationarity** (normalization failed) or **Horizon Mismatch** (gamma is too low for the seed lifecycle).

5. Esper Dynamics

- **Log:** host_loss (Train/Val), active_seeds_count, avg_seed_age, stage_counts (Bar chart of seeds in Germinating/Training/Blending).
- **Diagnosis:** avg_seed_age is a critical proxy for strategy.
    - Low Age = Churning/Cycling.
    - High Age = Holding/Stagnation.
    - Medium Age + High Variance = Healthy Lifecycle.

Conclusion

The training of a PPO policy for Esper is a complex engineering challenge that requires moving beyond standard RL baselines. The default assumptions of static environments, short horizons, and independent observation features will lead to failure in this domain.

Recommended Baseline Config (Iteration 0):

- **Curriculum:** Stage 1 (Short episodes, No Rent, High PBRS, Masked Actions).
- **Architecture:** MLP with LayerNorm (easier to debug than Transformer initially).
- **PPO:** $\gamma=0.995$, n_steps=2048, ent_coef={Op: 0.01, Blueprint: 0.05}.
- **Rewards:** Schedule 1 (Startup).

Recommended Next Two Iterations:

1. **Iteration 1 (Architecture):** Implement the **Permutation-Invariant Set Transformer** encoder. This is the highest ROI architectural change to solve slot management scalability.
2. **Iteration 2 (Reward Dynamics):** Implement **Rent Annealing** and **Autoregressive Action Heads** to transition the agent from "learning the rules" to "optimizing efficiency" without falling into local optima.

By treating the PPO agent as a meta-optimizer with a lifecycle-matched time horizon and a permutation-invariant view of the topology, we can plausibly outperform the Tamiyo heuristic.

Works cited

1. rl/torchrl/objectives/ppo.py at main · pytorch/rl - GitHub, accessed December 22, 2025, https://github.com/pytorch/rl/blob/main/torchrl/objectives/ppo.py
2. Proximal Policy Gradient (PPO) - CleanRL, accessed December 22, 2025, https://docs.cleanrl.dev/rl-algorithms/ppo/

3. Comprehensive Overview of Reward Engineering and Shaping in Advancing Reinforcement Learning Applications - IEEE Xplore, accessed December 22, 2025, https://ieeexplore.ieee.org/iel8/6287639/6514899/10763475.pdf
4. Multimodal Knowledge Alignment with Reinforcement Learning - GitHub Pages, accessed December 22, 2025, https://yj-yu.github.io/home/data/esper.pdf
5. Fusing Pre-Trained Language Models With Multimodal Prompts Through Reinforcement Learning - CVF Open Access, accessed December 22, 2025, https://openaccess.thecvf.com/content/CVPR2023/papers/Yu_Fusing_Pre-Trained_Language_Models_With_Multimodal_Prompts_Through_Reinforcement_Learning_CVPR_2023_paper.pdf
6. Learning to Explore in Diverse Reward Settings via Temporal-Difference-Error Maximization, accessed December 22, 2025, https://arxiv.org/html/2506.13345v2
7. Fusing Pre-Trained Language Models with Multimodal Prompts through Reinforcement Learning - Semantic Scholar, accessed December 22, 2025, https://www.semanticscholar.org/paper/Fusing-Pre-Trained-Language-Models-with-Multimodal-Yu-Chung/bee79110f7b89292955984c7110ed0de8ae719a1
8. 4 Counter-Intuitive Truths About Building Smarter AI Agents - Hackernoon, accessed December 22, 2025, https://hackernoon.com/4-counter-intuitive-truths-about-building-smarter-ai-agents
9. Policy Gradient without Boostrapping via Truncated Value Learning - OpenReview, accessed December 22, 2025, https://openreview.net/forum?id=nBYDP46s5N
10. Hyperspherical Normalization for Scalable Deep Reinforcement Learning - UT Austin Computer Science, accessed December 22, 2025, https://www.cs.utexas.edu/~pstone/Papers/bib2html-links/pstone_simba.pdf
11. Reinforcement Learning-Based Energy Management in Community Microgrids: A Comparative Study - Preprints.org, accessed December 22, 2025, https://www.preprints.org/frontend/manuscript/9f603afc4bd8afa14e60d28cbfa6f061/download_pub
12. Integrating Multi-Agent Planning and Reinforcement Learning through Reward and Exploration Machines - IRIS, accessed December 22, 2025, https://iris.uniroma1.it/retrieve/c2f43fa1-1835-4352-960a-2b65352a6e01/Tesi_dottorato_Trapasso.pdf
13. Action Space Reduction Strategies for Reinforcement Learning in Autonomous Driving, accessed December 22, 2025, https://www.researchgate.net/publication/393476611_Action_Space_Reduction_Strategies_for_Reinforcement_Learning_in_Autonomous_Driving
14. HOPE: A Reinforcement Learning-based Hybrid Policy Path Planner for Diverse Parking Scenarios - arXiv, accessed December 22, 2025, https://arxiv.org/html/2405.20579v2
15. Reward Wrappers - Gymnasium Documentation, accessed December 22, 2025, https://gymnasium.farama.org/api/wrappers/reward_wrappers/
16. ARS: Adaptive Reward Scaling for Multi-Task Reinforcement Learning | OpenReview, accessed December 22, 2025,

https://openreview.net/forum?id=U0mI6M7IvI
17. Bootstrapped Reward Shaping - arXiv, accessed December 22, 2025, https://arxiv.org/html/2501.00989v1
18. Efficient Potential-based Exploration in Reinforcement Learning using Inverse Dynamic Bisimulation Metric | OpenReview, accessed December 22, 2025, https://openreview.net/forum?id=0FhKURbTyF
19. in · Scryfall Magic: The Gathering Search, accessed December 22, 2025, https://scryfall.com/search?as=full&order=color&page=80&q=In&unique=cards
20. CDE: Curiosity-Driven Exploration for Efficient Reinforcement Learning in Large Language Models - arXiv, accessed December 22, 2025, https://arxiv.org/html/2509.09675v1
21. Exploration in Policy Gradient Methods: Entropy, Curiosity, and Intrinsic Motivation, accessed December 22, 2025, https://satyamcser.medium.com/exploration-in-policy-gradient-methods-entropy-curiosity-and-intrinsic-motivation-dd6f0ddbefd2?source=rss------reinforcement_learning-5
22. Hyperspherical Normalization for Scalable Deep Reinforcement Learning - arXiv, accessed December 22, 2025, https://arxiv.org/pdf/2502.15280
23. Hyperspherical Normalization for Scalable Deep Reinforcement Learning - arXiv, accessed December 22, 2025, https://arxiv.org/html/2502.15280v1
24. The Transformer Family Version 2.0 - Lil'Log, accessed December 22, 2025, https://lilianweng.github.io/posts/2023-01-27-the-transformer-family-v2/
25. Efficient reinforcement learning in resource allocation problems through permutation invariant multi-task learning - Institutional Knowledge (InK) @ SMU, accessed December 22, 2025, https://ink.library.smu.edu.sg/cgi/viewcontent.cgi?article=11364&context=sis_research
26. Invariant Slot Attention: Object Discovery with Slot-Centric Reference Frames - Proceedings of Machine Learning Research, accessed December 22, 2025, https://proceedings.mlr.press/v202/biza23a/biza23a.pdf
27. Set Transformers Tutorial - DL - Kaggle, accessed December 22, 2025, https://www.kaggle.com/code/auxeno/set-transformers-tutorial-dl
28. PPO on Atari - RL - Kaggle, accessed December 22, 2025, https://www.kaggle.com/code/auxeno/ppo-on-atari-rl
29. Adaptive Auxiliary Task Weighting for Reinforcement Learning, accessed December 22, 2025, http://papers.neurips.cc/paper/8724-adaptive-auxiliary-task-weighting-for-reinforcement-learning.pdf
30. profile booklet - Oxford University Innovation, accessed December 22, 2025, https://innovation.ox.ac.uk/wp-content/uploads/2022/02/202202_OUIbrochure_LifeSciences_final-compressed.pdf
31. Eternal Weekend: Is GWx Depths viable? - Legacy Maverick, accessed December 22, 2025, https://greensunszenith.com/eternal-weekend-gwx-depths-viability/
32. Neural Architecture Search with Reinforcement Learning - ResearchGate, accessed December 22, 2025,

https://www.researchgate.net/publication/309738632_Neural_Architecture_Search_with_Reinforcement_Learning
33. (PDF) ToolSample: Dual Dynamic Sampling Methods with Curriculum Learning for RL-based Tool Learning - ResearchGate, accessed December 22, 2025, https://www.researchgate.net/publication/395648912_ToolSample_Dual_Dynamic_Sampling_Methods_with_Curriculum_Learning_for_RL-based_Tool_Learning
34. Curriculum learning - Wikipedia, accessed December 22, 2025, https://en.wikipedia.org/wiki/Curriculum_learning
35. Unveiling Set Transformers: A New Approach to Particle Filter Reconstruction - Medium, accessed December 22, 2025, https://medium.com/correll-lab/unveiling-set-transformers-a-new-approacparticle-filter-reconstruction-30730a33c305
36. Set Norm and Equivariant Skip Connections: Putting the Deep in Deep Sets - PMC - NIH, accessed December 22, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC10465016/
37. ML-agents/docs/Training-PPO.md at master - GitHub, accessed December 22, 2025, https://github.com/gzrjzcx/ML-agents/blob/master/docs/Training-PPO.md
38. The Complete Practical Guide to PPO with Stable-Baselines3, accessed December 22, 2025, https://www.reinforcementlearningpath.com/the-complete-practical-guide-to-ppo-with-stable-baselines3/
39. Adaptive Reward Design for Reinforcement Learning - Computer Science, accessed December 22, 2025, https://www.cs.virginia.edu/~lufeng/papers/uai2025.pdf
40. Dynamic construction site layout optimization using deep reinforcement learning with PPO, accessed December 22, 2025, https://www.emerald.com/ecam/article/doi/10.1108/ECAM-07-2025-1103/1301532/Dynamic-construction-site-layout-optimization
41. Challenges for Reinforcement Learning in Grid World Environments - Queen Mary University of London, accessed December 22, 2025, https://qmro.qmul.ac.uk/xmlui/bitstream/handle/123456789/89875/thesis%20(1).pdf?sequence=2
42. Large Language Model-Brained GUI Agents: A Survey - arXiv, accessed December 22, 2025, https://arxiv.org/html/2411.18279v12