ENDURE: Ensemble Based Robust Reinforcement Learning with Reduced Sample Complexity

Sarra Alqahtani
Department of Computer Science
Wake Forest University
Winston-Salem, United States of America
e-mail: alqahtas@wfu.edu

***Abstract*—Intelligent systems deployed in the real world must be both *robust* to distributional shifts and *efficient* in how they learn from interaction. Reinforcement Learning (RL) has delivered strong results in controlled settings, but practical adoption is often limited by high sample costs and brittle behavior on rarely seen states. We present ENDURE, an ensemble-based method for model-free RL that targets intelligent-systems requirements: reliability, data efficiency, and simple integration into existing stacks. ENDURE reuses policies saved from a *single* training run and selects a compact, diverse subset via an *on-policy state diversity* metric that requires no extra environment interaction. At execution time, ENDURE applies a *risk-aware voting* rule that chooses the action with the lowest estimated short-horizon failure probability, improving safety without retraining. Across benchmark control tasks, ENDURE reaches optimal performance with up to $10\times$ fewer samples (CartPole) and about $2\times$ fewer samples (InvertedPendulum-v2) than single-policy baselines, while maintaining strong behavior on underrepresented states. We discuss engineering considerations—guardrails, metrics, and integration points making ENDURE a practical component for real-world intelligent systems.**

Keywords*: *Reinforcement Learning*; *Ensemble Methods*; *Robust Control*; *Risk Estimation*; *Sample Efficiency.

I. INTRODUCTION

Modern intelligent systems increasingly rely on autonomous decision-making under uncertainty. While RL offers a principled framework for sequential control, two constraints frequently hinder deployment: (i) *sample efficiency*: collecting interactions is slow, costly, or risky—and (ii) *robustness*: policies can degrade on states that were rare during training. For intelligent systems operating in dynamic environments, these constraints translate directly into engineering risks, higher operating costs, and difficult validation/assurance cycles.

Deep RL has demonstrated impressive results in games and control [1]–[3], yet many approaches achieve peak performance only after large-scale interaction budgets and may exhibit brittle behavior on distributional edges [4]. Ensemble methods are a natural fit for the intelligent-systems goal of reliability: different policies often specialize in different regions of the state space, and combining them can improve coverage [5]–[8]. However, training many agents in parallel or sequence substantially *increases* sample and compute budgets at odds with the efficiency imperative.

This paper introduces ENDURE, an ensemble approach designed around intelligent-systems constraints. Instead of training multiple agents, ENDURE treats *policies saved across a single training run* as a pool of candidates. We select a small, diverse subset using an *on-policy state diversity* criterion that leverages where each snapshot tends to operate well, avoiding additional rollouts. At execution time, we replace majority/averaging with a *risk-estimation voting* rule: for each candidate policy, estimate the short-horizon probability of failure from the current state and choose the least risky action. The result is a drop-in mechanism that increases robustness and reduces interaction cost, while fitting standard RL stacks and metrics.

From an engineering standpoint, ENDURE emphasizes: (a) *resource awareness*—no extra environment interaction to build the ensemble; (b) *risk-aware control*—an explicit failure-probability proxy used at decision time; and (c) *measurable outcomes*—sample complexity reduction, coverage of hard states, and simple integration points for guardrails and human-in-the-loop review.

Contributions. This work makes the following contributions to intelligent systems design and operation:

1) **Single-run ensembles for data efficiency.** We form diverse policy ensembles by reusing checkpoints from a *single* training run, avoiding the interaction and compute overhead of multi-agent training.
2) **On-policy state diversity for selection.** We introduce a practical diversity metric that identifies complementary policies based on their on-policy state distributions, guiding ensemble selection without additional rollouts.
3) **Risk-aware voting at inference.** We propose a failure-probability–based action selector that prefers low-risk actions at run time, improving robustness on underrepresented states with minimal overhead.
4) **Evidence on control benchmarks.** On CartPole and InvertedPendulum-v2, ENDURE attains optimal performance with up to $10\times$ and about $2\times$ fewer samples, respectively, than single-policy baselines, illustrating readiness for resource-constrained intelligent systems.

Paper organization. Section II provides background and notation. Section III describes ENDURE, including on-policy state diversity (Section III-A1) and risk-estimation voting (Section III-B) with precise labeling and interaction accounting. Section IV presents experiments and discusses the validation budget, checkpoint-frequency sensitivity, and robustness evaluation protocols. Section V concludes.

Courtesy of IARIA Board and IARIA Press. Original source: ThinkMind Digital Library https://www.thinkmind.org

II. Background

RL studies how an intelligent *agent* learns to make a sequence of decisions by interacting with an *environment*. At each step, the agent observes the current *state* (what the world looks like), takes an *action* (what to do next), and receives a *reward* (a number that indicates how desirable the outcome was). Over time, the agent adjusts its *policy*—its rule for choosing actions from states—to maximize the total (discounted) reward it expects to collect in the future.

A. Core Concepts (Plain Language)

State. The information the agent uses to decide (e.g., the pole angle and cart position in CartPole).
Action. The control the agent applies (e.g., push cart left or right; set a continuous torque).
Reward. Immediate feedback that encodes the task goal (e.g., $+1$ for each balanced time step).
Policy. A mapping from states to actions. It can be a table, a set of rules, or a neural network.
Episode/Trajectory. One run from a start state until termination (success, failure, or timeout).
Return. The total future reward from now onward; commonly written $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$ with $0 \leq \gamma < 1$ a discount factor.
Value / Q-Value. Predict how good a state (or state–action) is, in terms of expected return, if you keep following the current policy.

Formally, many RL problems are modeled as a Markov Decision Process (MDP) with states $\mathcal{S}$, actions $\mathcal{A}$, transition dynamics $P(\cdot \mid s, a)$, reward function $R(s, a)$, and discount γ. The objective is to find a policy $\pi(a \mid s)$ that maximizes expected return.

B. Why RL Is Powerful—but Also Costly and Brittle

RL excels when it is hard to write explicit rules but easy to evaluate outcomes (games, control, operations). However, two practical issues often limit real-world use:
Sample complexity. Learning a good policy can require millions of interactions. In physical systems (robots, vehicles) each sample is slow or expensive.
Generalization and robustness. Policies learn from the states they *happen* to visit during training. When they later face unusual or infrequent situations, performance can drop sharply. In RL, we say the policy is strong on its *on-policy* states (the distribution it visits while acting) but may be weak on underrepresented parts of the state space.

C. How Ensembles Help in RL

Ensembles combine multiple policies and choose an action by "voting." In supervised learning this improves accuracy and robustness. In RL, ensembles can also broaden coverage: different policies tend to specialize in different regions of the state space or phases of the task. The challenge is that training many separate RL agents makes sample complexity even worse.

Our work avoids that cost by reusing *policies from a single training run* (e.g., checkpoints saved at different times). These snapshots already differ in what they have learned and where they are strong, giving us diversity without extra environment interactions.

D. On-Policy State Diversity (Intuition)

To select a small but useful subset of snapshot policies, we measure how different their *on-policy* state visitations are. Intuitively, if Policy A routinely visits states that Policy B rarely sees—and vice versa—then combining them should cover more situations. We call this property *on-policy state diversity* and use it to guide ensemble selection before we spend any budget evaluating candidates online.

E. Risk Estimation for Safer Action Selection

Even within an ensemble, we still need to pick one action at each step. Majority or averaging can work, but they ignore *risk*. We instead estimate, for each candidate policy, the probability that taking its suggested action will lead to failure within a short horizon (e.g., the next H steps). The ensemble then chooses the *least risky* action. Practically, we train a lightweight predictor that maps the current state to an estimated failure probability for each policy. This *risk-estimation voting* prefers actions that are both competent and less likely to unravel, improving robustness without retraining.

F. How This Connects to Our Contributions

This paper operationalizes the above ideas as ENDURE. We (i) harvest diverse policies from one training run, (ii) score candidate ensembles using on-policy state diversity to minimize evaluation budget, and (iii) deploy a risk-aware voting rule that picks the action with the lowest predicted short-term failure probability. The result is an ensemble that achieves competitive (often optimal) performance with far fewer samples and better behavior on hard or underrepresented states.

III. ENDURE: Robust Reinforcement Learning Through Ensemble

This work proposes ENDURE, a robust RL framework for learning and selecting an ensemble of diverse policies from *a single training run*. ENDURE has two components: (1) Ensemble policy selection via on-policy state diversity, and (2) Ensemble action selection via risk estimation voting. The ensemble policy selection component finds the best subset of K policies from the entire set of N policies collected during one training run. The policy selection is based on a metric called *on-policy state diversity* to be described in Section III-A1. The voting technique is developed based on risk estimation of each policy failing the task in the next H timesteps as described in Section III-B.

A. Ensemble Policy Selection via On-Policy State Diversity

The RL agent's policy changes throughout training and its optimality in a state is heavily correlated with the frequency the agent visits that state [4]. Motivating by this fact, we split the training process into N different training periods creating N different policies. Given the N policies for the agent throughout training, we here consider the problem of policy selection, or

Courtesy of IARIA Board and IARIA Press. Original source: ThinkMind Digital Library https://www.thinkmind.org

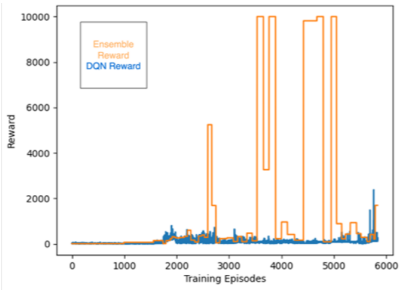
selecting K out of N policies to be in the final ensemble. In supervised learning, the problem of selecting K models to be in an ensemble from a set of N possible models is known as *model selection*. It has been shown in supervised learning settings that a necessary and sufficient criteria for an ensemble to perform better than its counterpart individual models is for the individual models to be both accurate and diverse [9]. In RL settings, we empirically found that accuracy, as defined by high reward, is insufficient for producing a good ensemble. Figure 1a shows the evaluation results of ensemble RL agents made up of the best policies, according to their rewards. However, those policies are not optimal in comparison to the randomly selected policies in Figure 1b. Further experiments in Section IV show that these policies are not optimal.

It is more challenging to quantify and optimize *diversity* in RL settings than in the supervised learning settings [6]. Prior works follow an ad-hoc approach. Particularly, they set $N{=}K$, and train these K policies end-to-end to include in the ensemble, using methods to ensure the resulting K policies are diverse [5]–[8]. However, in practice there is no guarantee of diversity once these policies are trained. Since these approaches only have K policies to choose from after training, it is infeasible to choose a new set of policies, should this be the case. Supervised learning researchers have long attempted to solve this problem. Model selection methods have been developed to pick the most accurate and diverse classifiers through examining model performance on a validation set, and the difference in model outputs within this validation set [10]–[12]. In RL settings, an analogous approach would require rolling out the potential ensemble for some number of episodes to gather information about the reward the ensemble achieves. These rollouts would contribute heavily to the sample complexity. We show in Figure 1b that, without considering the additional rollout sample complexity, we can find high-performing ensembles by simply testing ensembles with random assortments of policies. However, we want to minimize the amount of possible ensembles we have to test for reward. To this end, we develop a metric, *on-policy state diversity*, to measure the diversity of individual policies in a more efficient way.

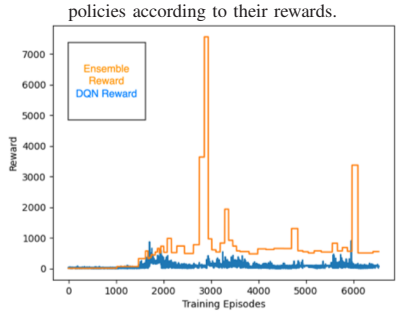
1) On-Policy State Diversity: In [4], an agent's on-policy states are defined as the states which the agent encounters during a standard training episode, and off-policy states as those states which are not encountered as frequently. They found that the percentage of the maximum return the policy achieves, starting in a state s, is correlated with how often the agent visits s [4]. We aim to choose a subset of $K{\leq}N$ policies which have diverse expertise within the set of possible environment states.

Given a set $S_i = \{s_{i,1}, \ldots, s_{i,T}\}$ for each policy i, containing a small ($T \leq 5000$) set of vector-valued on-policy states encountered by that policy, we summarize each S_i by its empirical mean μ_i and elementwise standard deviation σ_i:

$$\mu_i = \frac{1}{T}\sum_{t=1}^{T} s_{i,t}, \qquad \sigma_i = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(s_{i,t}-\mu_i)^2}. \tag{1}$$



(a) Ensemble vs DQN reward with the 4 best policies according to their rewards.



(b) Ensemble vs DQN reward with four random policies in the ensemble.

Figure 1. Performance of DQN [13] and ensemble policies across a training run in the CartPole environment. Plots differ in ensemble selection mechanisms.

We then define the pairwise on-policy state diversity metric between policy i and policy j as a normalized distance:

$$\lambda_{i,j} = \frac{1}{d}\left(\|\mu_i - \mu_j\|_2^2 + \|\sigma_i - \sigma_j\|_2^2\right), \tag{2}$$

where d is the state dimension. This yields a fast-to-compute proxy for differences in typical visited states and spread.

We then define on-policy state diversity of a sampled ensemble of policies as the average pairwise state diversity among the individual policies:

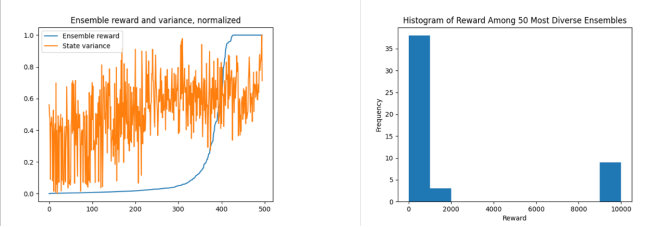
$$\lambda(\mathcal{E}) = \frac{2}{K(K-1)}\sum_{i\in\mathcal{E}}\sum_{\substack{j\in\mathcal{E}\\ j>i}}\lambda_{i,j}. \tag{3}$$

This metric can be computed across large S_i within seconds after precomputing (μ_i, σ_i) for each checkpoint. Computing all pairwise $\lambda_{i,j}$ costs $O(N^2 d)$ time, and evaluating $\lambda(\mathcal{E})$ for a candidate ensemble is $O(K^2)$ given the pairwise table.

Concrete values and feasibility. In our continuous-control experiments (Section IV), we save checkpoints uniformly across training: for CartPole we train for 1,000,000 timesteps and use $N = 20$ checkpoints, and for InvertedPendulum-v2 we train for 250,000 timesteps and use $N = 10$ checkpoints. We use ensemble size $K = 4$. For CartPole, full enumeration yields $\binom{20}{4} = 4845$ candidate ensembles, which is feasible to score by Equation 3. We then evaluate only a subset under a validation budget of M ensembles (below).

We propose to perform policy selection in a way that minimizes the number of ensemble rollouts we need to perform. Assume we have a validation budget such that we can test the

Courtesy of IARIA Board and IARIA Press. Original source: ThinkMind Digital Library https://www.thinkmind.org



(a) Average pairwise on-policy state diversity versus reward for 500 candidate ensembles.

(b) Histogram of the reward of the 50 most diverse ensembles according to on-policy state diversity.

Figure 2. Experiments conducted with the same training run in the CartPole environment.

performance of M ensembles. We sort all $\binom{N}{K}$ ensembles by their mutual on-policy state diversity values and choose the top M ensembles with respect to this metric. We then test these candidate ensembles in the environment, and choose the best ensemble according to the observed reward. Figure 2b shows how many 'good' ensembles we can expect from this method in the CartPole environment. Figure 2b shows that in the top 50 ensembles with respect to $\lambda(\mathcal{E})$ (Equation 3), 10 ensembles achieve what we consider a high reward, whereas the remaining 40 achieve a reward similar to the individual policies within the ensemble. In these experiments, we use $M = 50$ as the validation budget.

B. Ensemble Action Selection via Risk Estimation Voting

Given an ensemble with K policies, we aim to develop a method for selecting one action among the selected actions by the ensemble policies which will be fed into the environment. Specifically, given a set of actions $A_t = \{\pi_i(s_t) \mid 0 \leq i < K\}$, we wish to develop a function f such that $\hat{a}_t = f(A_t)$. A standard voting scheme is the majority vote approach, which takes an action according to:

$$\hat{a}_t = \arg\max_{a \in \mathcal{A}} \left(\sum_{i=0}^{K-1} \mathbb{1}[A_t[i] = a] \right) \tag{4}$$

where $\mathbb{1}$ is the indicator function. This voting scheme only works well when a_t is discrete. In the case of a continuous, vector-valued action space, we can use average voting. In this scheme, the action is selected as

$$\hat{a}_t = \frac{1}{K} \sum_{i=0}^{K-1} A_t[i]. \tag{5}$$

As an alternative to these approaches, we propose risk estimation voting. The intuition behind this approach is to pick the action from the policy that is the least likely to fail the task, from the current state.

Risk definition and labels. We assume failure and success are well-defined via the environment termination condition (e.g., pole falls, pendulum diverges), distinct from time-limit truncation. For checkpoint policy i, we define the short-horizon failure probability conditioned on *(state, action)*:

$$g_i(s,a) = \Pr\Big(\text{failure occurs within the next } H \text{ steps} \,\Big|\, s_t = s,\ a_t = a,\ \pi_i\Big). \tag{6}$$

We obtain binary risk labels from trajectories collected during training *while checkpoint policy i is active* (or within its training window), introducing no additional environment interaction beyond the single training run. For each logged transition $(s_{i,t}, a_{i,t})$, we label

$$y_{i,t}^{(H)} = \mathbb{1}\Big[\text{failure occurs in } \{t+1, \ldots, t+H\} \text{ within the same episode}\Big]. \tag{7}$$

This labeling uses only episode termination signals already produced during the single training run and does not use data from later checkpoints to label earlier ones (avoiding "future-data leakage" that could inflate sample-efficiency claims).

Suppose we have an oracle function, g_i, which computes the probability that policy i will fail the task in the next H timesteps from state s_t. Then, risk estimation voting can be derived as

$$\hat{a}_t = A_t \left[\arg\min_{0 \leq i < K} \{g_i(s_t)\} \right]. \tag{8}$$

Action-conditioned risk voting (implementation). To address action-dependence, we use the action proposed by policy i, $a_t^{(i)} = \pi_i(s_t)$, and select:

$$\hat{a}_t = a_t^{(i^\star)}, \quad i^\star = \arg\min_{0 \leq i < K} \hat{g}_i\left(s_t, a_t^{(i)}\right), \tag{9}$$

where $\hat{g}_i$ approximates g_i using supervised learning on (s,a) and labels $y_{i,t}^{(H)}$.

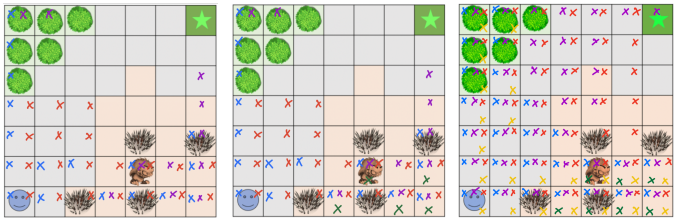
Given the risk estimation voting scheme, all we need to do is approximate the function g_i, such that $\hat{g}_i(s_t) \approx g_i(s_t)$ $\forall s_t$. Since s_t can potentially take infinite values, [14] chooses to approximate g_i through a neural network, trained with supervised learning. We follow their methodology, which involves first collecting a dataset $\{(s_t, g_i(s_t))\}$ that contains around $100,000$ samples. This dataset can be collected from observing the agent as it trains. We use a neural network with two fully-connected layers of 64 neurons each, and train it to predict $g_i(s_t)$ from s_t.

Overhead note. For larger systems, training a separate risk estimator per checkpoint can be expensive; in practice, we can train risk estimators only for the K selected ensemble policies rather than all N checkpoints, and we can reuse the same logged data collected during training (no extra interaction).

IV. Experiments

We perform experiments in discrete and continuous environments. First, we analyze risk estimation ensemble voting in a gridworld environment. Then, we examine the effect of risk estimation voting combined with on-policy state diversity model selection in continuous control environments CartPole and InvertedPendulum-v2.

Courtesy of IARIA Board and IARIA Press. Original source: ThinkMind Digital Library https://www.thinkmind.org



(a) Three phases of the RL agent. The blue, purple, and red x's are for the beginning, middle, and end of training.

(b) Three phases of the RL agent along with an ensemble (green) of the three; the ensemble uses risk-estimation voting.

(c) Three phases of a suboptimal RL agent, its risk-estimation ensemble (green), and its majority-vote ensemble (yellow).

Figure 3. Performance of individual and ensemble RL policies in the gridworld environment. X's mark states from which the corresponding policy fails to complete the task.

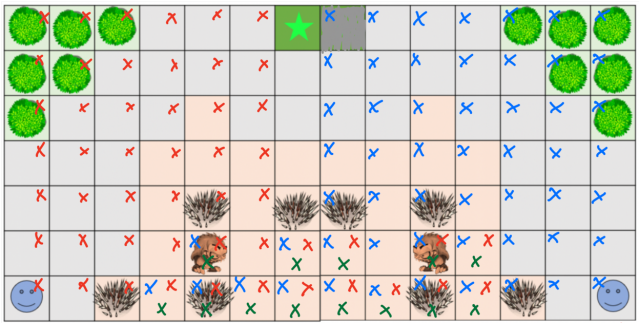


Figure 4. Two RL agents, the blue one starts in the bottom-left, the red one starts in the bottom-right, and their ensemble (green), using risk estimation voting where x's represent states from which the corresponding agent fails to complete the task.

A. Discrete Environments

We adapt the gridworld from [15], shown in Figure 3. In this environment, the agent starts in the bottom-left corner, and the goal is to navigate to the top-right corner using the actions up, left, down, and right. If the agent gets too close to the monster, or if it takes too many timesteps to reach the goal state, the agent fails and receives a large negative reward.

In Figure 3(a), we show the adeptness of three phases of an RL agent. We train an RL agent for a fixed number of timesteps in this environment and save its weights at the beginning, middle, and end of training. In Figure 3, we denote the states from which an agent fails to solve the task with an appropriately colored 'X'. We see that the agent at the beginning, middle, and end of training has different sets of states from which it can solve the task.

In Figure 3(b), we include the performance of an ensemble agent using risk estimation voting. Note that instead of using a neural network to approximate g_i in this environment, we used a tabular representation, since the gridworld environment has a finite set of possible states. We see that the ensemble agent only fails the task from the states in which all individual agents also fail the task, indicating that the ensemble can correctly choose which agent to act based on individual specialization. Figure 3(c) shows the same experiment, but with a worse base RL agent and a majority vote baseline. We observe that there are many states from which the individual policies fail the task, but the risk estimation ensemble method is able to successfully complete the task from many of these states. This shows that the risk estimation ensemble policies have the potential to perform much better than any individual policy. We also observe that the majority vote policy does better than the individual policies, but still has many states from which it fails the task. This shows that the improvement in the policy is due in part to risk estimation voting, not just the nature of ensembles.

To further visualize the benefit of ensembles when policies specialize in different states, we created an alternate gridworld, which is the original gridworld concatenated with a flipped version of itself (Figure 4). We trained one RL agent (blue) which starts in the bottom-left corner and one agent which starts in the bottom-right (red). When we combine these two agents in an ensemble with risk estimation, the ensemble agent can solve the task from all states except a handful from which the task is impossible.

B. Continuous Environments

Next, we examine the performance of the risk estimation ensemble in two environments with continuous state spaces, CartPole and InvertedPendulum-v2 [3]. Both these environments could continue forever if the control policy is good enough, so we cut the episodes off at 10,000 timesteps. The agent receives a reward of 1 for every timestep it is left standing, so the maximum reward is also 10,000.

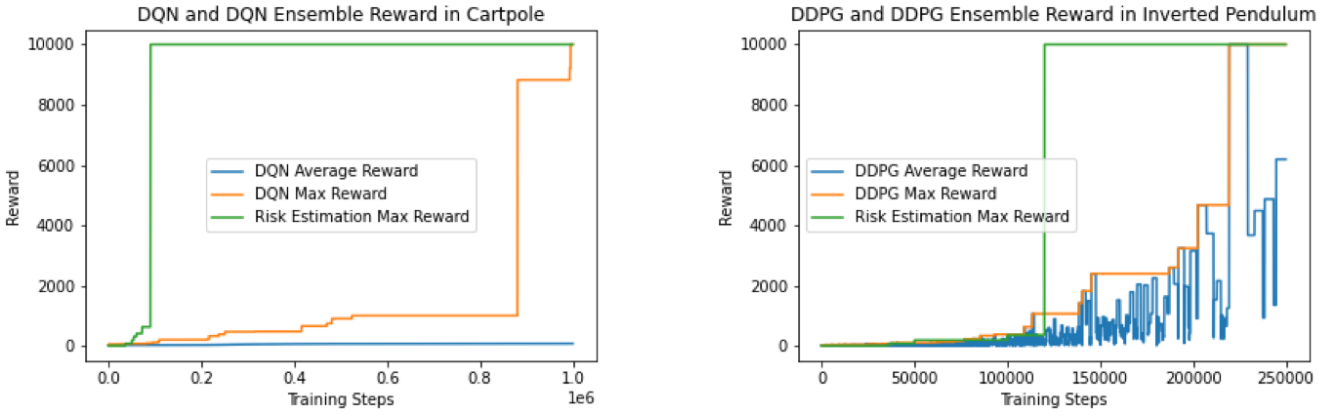
We trained the agents for both environments using Deep Q-Learning (DQN) [13] and Deep Deterministic Policy Gradient (DDPG) [16], but we present only the optimal performance of each of them. The agent for the CartPole environment was trained with DQN for 1,000,000 timesteps; the DDPG agent was trained on InvertedPendulum-v2 for 250,000 timesteps. For testing, we perform ensemble policy selection at set intervals and run the ensemble with risk estimation voting, to give an idea of what kind of ensemble performance is possible at various numbers of training timesteps.

Checkpointing parameters. In these experiments, we save $N = 20$ checkpoints for CartPole (every 50,000 timesteps) and $N = 10$ checkpoints for InvertedPendulum-v2 (every 25,000 timesteps). We use ensemble size $K = 4$. We use a validation budget of $M = 50$ candidate ensembles ranked by on-policy state diversity (Section III-A1) and selected by online evaluation.

Figure 5(a) shows the results for the CartPole environment. We show the average DQN agent reward, the maximum reward that the DQN agent ever achieves, and the reward of the best ensemble, which our algorithm finds. The risk estimation ensemble can achieve the maximum reward of 10,000 after only 100,000 timesteps of training, which is about a $10\times$ decrease in sample complexity (measured against the single-run training budget).

Figure 5(b) shows the results for the same experiment in the InvertedPendulum-v2 environment, except with DDPG RL

Courtesy of IARIA Board and IARIA Press. Original source: ThinkMind Digital Library https://www.thinkmind.org



(a) CartPole (DQN) **(b)** InvertedPendulum (DDPG)

Figure 5. Average single-policy performance, best checkpoint, and best ensemble. (a) CartPole; (b) InvertedPendulum.

training algorithm. We see that the risk estimation ensemble reaches the maximum reward at around 125,000 timesteps, about a $2\times$ decrease in sample complexity (against the single-run training budget).

Checkpoint frequency sensitivity (engineering consideration). Ensemble quality depends on N and checkpoint spacing. If checkpoints are too frequent, adjacent policies may be redundant and reduce effective diversity; if too sparse, the pool may miss complementary behaviors. Practical guidance is to save checkpoints across distinct learning phases and optionally use non-uniform spacing (denser early training, sparser late training) to increase candidate diversity at fixed N.

V. Conclusion

We presented ENDURE, an ensemble method for RL tailored to intelligent-systems constraints: robustness, efficiency, and ease of integration. By harvesting policy snapshots from a *single* training run, selecting a compact subset via on-policy state diversity, and applying a risk-aware voting rule at execution time, ENDURE improves behavior on underrepresented states while substantially reducing the interactions needed to reach optimal performance. On standard control tasks, ENDURE achieves up to $10\times$ (CartPole) and about $2\times$ (InvertedPendulum-v2) reductions in sample complexity relative to single-policy baselines.

For practitioners, ENDURE is a drop-in augmentation to existing RL stacks that aligns with common assurance practices: it exposes explicit selection criteria, admits guardrails (e.g., thresholds, escalation), and supports measurement via standard control metrics. Future work will study deployment aspects including uncertainty calibration for the risk estimator, integration with formal checks for safety constraints, extension to multi-agent settings, and evaluation on hardware-in-the-loop or edge scenarios where interaction budgets are tight.

References

[1] D. Silver et al., "Mastering the game of Go with deep neural networks and tree search", *Nature*, vol. 529, no. 7587, pp. 484–489, 2016. DOI: 10.1038/nature16961.

[2] O. Vinyals et al., "Starcraft II: A new challenge for reinforcement learning", arXiv preprint arXiv:1708.04782, 2017, arXiv: 1708.04782. [Online]. Available: https://arxiv.org/abs/1708.04782.

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[4] J. Uesato et al., "Rigorous agent evaluation: An adversarial approach to uncover catastrophic failures", arXiv preprint arXiv:1812.01647, 2018, arXiv: 1812.01647. [Online]. Available: https://arxiv.org/abs/1812.01647.

[5] K. Lee, M. Laskin, A. Srinivas, and P. Abbeel, "Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning", in *Proceedings of the 38th International Conference on Machine Learning*, M. Meila and T. Zhang, Eds., ser. Proceedings of Machine Learning Research, vol. 139, PMLR, 2021, pp. 6131–6141.

[6] R. Saphal, B. Ravindran, D. Mudigere, S. Avancha, and B. Kaul, "Seerl: Sample efficient ensemble reinforcement learning", arXiv preprint arXiv:2001.05209, 2020, arXiv: 2001.05209. [Online]. Available: https://arxiv.org/abs/2001.05209.

[7] Q. He, C. Gong, Y. Qu, X. Chen, X. Hou, and Y. Liu, "Mepg: A minimalist ensemble policy gradient framework for deep reinforcement learning", arXiv preprint arXiv:2109.10552, 2021, arXiv: 2109.10552. [Online]. Available: https://arxiv.org/abs/2109.10552.

[8] X. Chen, L. Cao, C. Li, Z. Xu, and J. Lai, "Ensemble network architecture for deep reinforcement learning", *Mathematical Problems in Engineering*, vol. 2018, pp. 1–6, 2018. DOI: 10.1155/2018/2129393.

[9] T. G. Dietterich, "Ensemble methods in machine learning", in *Multiple Classifier Systems*, ser. Lecture Notes in Computer Science, vol. 1857, Berlin, Heidelberg: Springer, 2000, pp. 1–15.

[10] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes, "Ensemble selection from libraries of models", in *Proceedings of the Twenty-First International Conference on Machine Learning*, ser. ICML '04, New York, NY, USA: Association for Computing Machinery, 2004, p. 18. DOI: 10.1145/1015330.1015432.

[11] T. Pang, K. Xu, C. Du, N. Chen, and J. Zhu, "Improving adversarial robustness via promoting ensemble diversity", arXiv preprint arXiv:1901.08846, 2019, arXiv: 1901.08846. [Online]. Available: https://arxiv.org/abs/1901.08846.

Courtesy of IARIA Board and IARIA Press. Original source: ThinkMind Digital Library https://www.thinkmind.org

[12] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on ensemble learning", *Frontiers of Computer Science*, vol. 14, pp. 241–258, 2020. DOI: 10.1007/s11704-019-8208-z.

[13] V. Mnih et al., "Playing atari with deep reinforcement learning", arXiv preprint arXiv:1312.5602, 2013, arXiv: 1312.5602. [Online]. Available: https://arxiv.org/abs/1312.5602.

[14] E. A. Abolfathi, J. Luo, P. Yadmellat, and K. Rezaee, "Coachnet: An adversarial sampling approach for reinforcement learning", arXiv preprint arXiv:2101.02649, 2021, arXiv: 2101.02649. [Online]. Available: https://arxiv.org/abs/2101.02649.

[15] J. van der Waa, J. van Diggelen, K. van den Bosch, and M. Neerincx, "Contrastive explanations for reinforcement learning in terms of expected consequences", arXiv preprint arXiv:1807.08706, 2018, arXiv: 1807.08706. [Online]. Available: https://arxiv.org/abs/1807.08706.

[16] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning", in *International Conference on Learning Representations (Poster)*, OpenReview.net, 2016.

Courtesy of IARIA Board and IARIA Press. Original source: ThinkMind Digital Library https://www.thinkmind.org