SEERL : Sample Efficient Ensemble Reinforcement Learning

Rohan Saphal  
Indian Institute of Technology Madras  
Chennai, India  
rohansaphal@gmail.com

Balaraman Ravindran  
Robert Bosch Center for Data Science and Artificial Intelligence  
Indian Institute of Technology Madras  
Chennai, India  
ravi@cse.iitm.ac.in

Dheevatsa Mudigere  
Facebook Inc  
Menlo Park, USA  
dheevatsa@fb.com

Sasikant Avancha  
Intel Corporation  
Bangalore, India  
sasikanth.avancha@intel.com

Bharat Kaul  
Intel Corporation  
Bangalore, India  
bharat.kaul@intel.com

ABSTRACT

Ensemble learning is a very prevalent method employed in machine learning. The relative success of ensemble methods is attributed to their ability to tackle a wide range of instances and complex problems that require different low-level approaches. However, ensemble methods are relatively less popular in reinforcement learning owing to the high sample complexity and computational expense involved in obtaining a diverse ensemble. We present a novel training and model selection framework for model-free reinforcement algorithms that use ensembles of policies obtained from a single training run. These policies are diverse in nature and are learned through directed perturbation of the model parameters at regular intervals. We show that learning and selecting an adequately diverse set of policies is required for a good ensemble while extreme diversity can prove detrimental to overall performance. Selection of an adequately diverse set of policies is done through our novel policy selection framework. We evaluate our approach on challenging discrete and continuous control tasks and also discuss various ensembling strategies. Our framework is substantially sample efficient, computationally inexpensive and is seen to outperform state-of-the-art (SOTA) scores in Atari 2600 and Mujoco.

KEYWORDS

Deep Reinforcement Learning, Ensemble methods, Combining policies

ACM Reference Format:
Rohan Saphal, Balaraman Ravindran, Dheevatsa Mudigere, Sasikant Avancha, and Bharat Kaul. 2021. SEERL : Sample Efficient Ensemble Reinforcement Learning. In Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), Online, May 3–7, 2021, IFAAMAS, 9 pages.

1 INTRODUCTION

Deep reinforcement learning over the years has made considerable advancements with applications across a variety of domains – from learning to play Atari 2600 suite from raw visual inputs [20], mastering board games [23, 26], learning locomotion skills for robotics [16, 24, 25], mastering Starcraft [30], the development of Alpha Fold [6] to predict the 3D structure of a protein and most recently, the improvements in model-based [13] and off-policy reinforcement learning [9, 10]. Nonetheless, it is a challenging task to create a single agent that performs well, is sample efficient and is robust. There are considerable hurdles surrounding training and optimization of deep reinforcement learning algorithms such as the sensitivity to hyper-parameters, the high variance associated with the learning process and high sample complexity. In order to overcome these problems, we exploit the well-known concept of ensemble learning [4] and adapt it for reinforcement learning in a novel way. Traditionally, the idea of using ensembles in reinforcement learning settings is associated with combining multiple value functions or policies from different agents. These agents could be the same algorithm trained across different hyper-parameter settings, generated by different algorithms altogether [5, 7, 32] or by training multiple networks of an agent in parallel [14]. Training multiple such agents is an approach that cannot be used in practice owing to high sample complexity and computational expense.

We tackle this problem by creating sufficiently diverse policies from a single training run. The policies are generated in a serial manner, one after the other, where the subsequent policies take into account the diversity from earlier policies. Our framework is seen to achieve state-of-the-art (SOTA) performance on popular reinforcement learning domains. Our approach to sequentially generate policies is inspired by the recent developments in the deep learning literature studying the effects of learning rate schedules and their impact on generalization [15, 21]. It is shown that learning rate annealing generalizes better than using a small constant learning rate and high initial learning rate impacts the model’s selection of the local minimum to converge [12, 15]. We leverage these properties of neural network training to learn a diverse ensemble of policies. The diversity among the policies are obtained by the directed perturbation of the model weights at regular intervals. The directed perturbation is induced by sudden and sharp variations in the learning rate, and for doing so, we employ cyclical learning rates [17]. When the model weights are perturbed using larger learning rates, the directed motion along the gradient direction prevents the optimizer from settling in any sharp basins and directs the model into the
We also show through experimentation that directed perturbation we introduce a novel framework that selects the best subset of policies to include in the ensemble. Our approach uses the trajectories obtained during training the policies to find this subset. Since we use models from a single training run instead of training different models independently from scratch, we refer to our approach as Sample Efficient Ensemble Reinforcement Learning (SEERL). To summarize, our main contributions are:

- A sample efficient framework for learning $M$ diverse models in a serial manner from a single training run with no additional computation cost. The framework can adopt any existing reinforcement learning algorithm to create an ensemble.
- A novel optimization framework to select the best subset of policies for the ensemble. It acts as a filter to choose the best performing and diverse subset of policies without using any additional samples from the environment.
- Evaluation of various ensemble strategies for discrete and continuous action spaces with SOTA performance across multiple environments.

We demonstrate the effectiveness of SEERL for discrete (Atari 2600 [3]) and continuous control benchmarks (Mujoco [29]). Our experiments show that SEERL consistently outperforms popular model-free and model-based reinforcement learning methods including those that use ensembles.

2 RELATED WORK

There has recently been a small body of work on using ensembles for reinforcement learning, primarily the use of ensembles during the training phase to reduce the variance and improve robustness of the policy.

Value function based methods such as Averaged DQN [1] train multiple Q networks in parallel with different weight initialization and average the Q values from all the different networks to reduce variance. It results in learning policies that are much more stable. However, the approach requires training multiple networks simultaneously and a possibility that the model might diverge if either of the Q values being averaged is biased. Bootstrapped DQN [22] uses an ensemble of Q functions for efficient exploration but does not enforce diversity in the Q networks. The Q head is chosen at random and used to generate actions for that episode. The multiple heads are also trained in parallel while we train our models in a serial manner.

SUNRISE [14] also uses an ensemble of Q networks that are randomly initialized and each head trained with a different set of samples to stabilize learning and enforce diversity. The actions are selected by examining the Upper Confidence Bound (UCB) of the Q values and choosing the action with the highest upper confidence bound to allow for exploration. However, the framework does not account for bias from the different Q heads which in turn would affect the UCB. Our policy selection framework prevents such biases by filtering out such models during evaluation. Additionally, SUNRISE works only where the base learner is an off-policy algorithm whereas our framework works with both on-policy and off-policy methods.

Earlier works [5, 7, 8, 32] explore the idea of value function ensembles and policy ensembles during evaluation phase. However, value function ensembles from different algorithms trained independently could degrade performance as they tend to converge to different estimates of the value function that are mutually inconsistent. An alternative method [18] tries to tackle this problem by having a meta-learner linearly combine the value functions from different algorithms during training time to adjust for the inherent bias and variance. Although training multiple policies or value functions in parallel is sample efficient, it tends to reduce the diversity among them and is computationally expensive. Our method combines the best of both approaches and improves the performance of the algorithm by balancing sample complexity with the computational expense. Training deep neural network architectures with cycling learning rates [17, 27] and using ensembles in supervised learning settings, [11] have shown to be useful. The authors [11] show that in each cycle, the models obtained are comparable with those learned using traditional learning rate schedules. Even though the model is seen to degrade in performance temporarily, the new model surpasses the previous one, as the learning rate anneals.

3 PRELIMINARIES

Reinforcement learning is associated with sequential decision making and involves the interaction of an agent with an environment. In this paper, we consider a discrete-time finite-horizon Markov Decision Process (MDP) defined by $(S, A, P, r, γ, ρ)$, where $S$ denotes the state space, $A$ denotes the action space, $P \equiv P(s'|s,a)$ the transition function, $ρ$, the probability distribution over the initial states, $r(s,a) ⊆ [0,1]$, the reward function with $R_{t+1}$ being the scalar reward at time $t+1$ when the agent in state $s_t$ took the action $a_t$, and $γ ∈ (0,1)$ the discount factor. The policy dictates the behavior of an agent at a particular state in an environment. More formally, a policy is defined by $π : S → P(A)$ where $P(A)$ denotes the probability distribution over actions $a ∈ A$. The objective of the agent is to maximize the discounted return $G_t = Σ_{i=0}^{T−1} γ^{T−1} R_i$.

4 SEERL

In this paper, we propose SEERL, a novel framework for learning and selecting an ensemble of diverse policies obtained from a single training instance. Unlike supervised learning, where the dataset can be reused for training different models, training multiple RL agents on the same trajectories can result in a lack of diversity among the policies. Instead, multiple RL agents can be trained independently for an ensemble but would suffer from high sample complexity. If each agent requires $N$ number of samples and the computational expense for training a single agent is $C$, then training $M$ agents independently require $M \times N$ samples and $M \times C$ computational cost. If trained in parallel, only $N$ samples are required, but the...
As the learning rate anneals to a small value, the model converges. The annealing helps in generalization and converging to a local minima during training at periodic intervals when the learning rate annealing traps it into a well behaved local minima. The formulation is as follows:

$$\alpha(t) = \frac{\alpha_0}{2} \left( \cos \left( \frac{\pi \mod (t - 1, [T/M])}{[T/M]} \right) + 1 \right)$$  \hspace{1cm} (1)$$

where $\alpha_0$ is the initial learning rate, $t$ is the time-step, and $T$ is the total number of time steps for which the agent is trained, and $M$ is the number of models.

### 4.2 Policy selection

In order to avoid bias from the more inferior policies, the best $m$ policies should ideally be selected for the ensemble. At the same time, the policies also need to be diverse to obtain good performance in different parts of the state space when used in an ensemble. Near identical policies would not yield much improvement in an ensemble.

#### 4.2.1 Framework

We propose an optimization framework to select the best subset of policies. The formulation has two parts, a policy error term and a Kullback-Liebler (KL) divergence term indicative of diversity. Only optimizing for the policy error term would result in the selection of policies with excellent performance. The addition of the KL divergence term helps balance the requirements of performance and diversity. We have a hyper-parameter $\beta \in [1, 2]$, that balances between diversity and performance. The KL divergence is calculated based on the action distribution between the two policies over the state space. The formulation is as follows:

$$J(w) = \sum_{s \in S} P(s) \left[ \sum_{i \in M} w_i B_i(s) \right]^2$$  \hspace{1cm} (2)$$

$$B_i(s) = \left( \sum_{a \in A} L(s, a) - \frac{\beta}{M - 1} \sum_{k \in M, k \neq i} KL(\pi_i(s) || \pi_k(s)) \right)$$  \hspace{1cm} (3)$$

with the following constraints $\sum_{i \in M} w_i = 1$, $w_i \geq 0 \forall i$. $S$ is the set of states and $P(s)$ is the probability of observing a particular state. $L(s, a)$ is the weighted error associated with the model and is indicative of the performance of the model. We weigh the loss in a manner that we give more weight when the loss is above a certain threshold value, and the action taken by the model, $a$, matches the action taken by the ensemble $a_e$. We formalize $L(s, a)$ as follows:

$$L(s, a) = \begin{cases} 1, & \text{if } |L'(s, a)| \geq T_{\text{err}} \text{ and } |a - a_e| < \epsilon \text{, is continuous} \\ 1, & \text{if } |L'(s, a)| \geq T_{\text{err}} \text{ and } a = a_e \text{, is discrete} \quad (4) \\ 0, & \text{otherwise} \end{cases}$$

$L'(s, a)$ is the total error for a particular state-action pair. While the overall process is algorithm agnostic, the details of the total error depend on the actual RL algorithm adopted. $T_{\text{err}}$ is the threshold value that is used to distinguish the right actions from the wrong. If $L'(s, a)$ is above the threshold error and the actions match, then we would like to consider that action from the policy as a wrong one.

In a discrete action space, it is relatively easy to determine if the action taken by the model and ensemble are the same. However, in continuous action space, the final ensemble action and the action taken by the policy might not coincide. Therefore, we introduce a $\epsilon$ bound on the ensemble action. If the action from the model is within a $\epsilon$ distance from the ensemble action, we consider it as a match. $\epsilon$ ranges between 0.005 to 0.01, depending on the environment.

We use a squared loss formulation to capture the inter-dependencies among the policies. Instead, if the degree were 1, the objective function would be weighted the sum of the loss, and the one with the

Figure 1: Cyclic cosine annealing learning rate schedule. $\alpha_0$ is set at 0.05, number of models $M = 5$ and training timesteps $T = 1000000$. Computational cost remains at $M \times C$. Though training agents in parallel is a possible solution to tackle sample complexity, it is computationally expensive and limits the diversity among the learned policies since every policy observes the same state. SEERL follows a two-step process-learning policies and policy selection, summarized in Algorithm 1. Learning policies involves saving $M$ policies during training at periodic intervals when the learning rate anneals to a small value. This is followed by policy selection that finds the best subset of $m$ policies from the entire set $M$. Policy selection uses an optimization framework that uses the $N$ samples obtained during training to find the $m$ policies. These policies are then used as an ensemble during the evaluation phase. Thus SEERL produces $M$ models for the ensemble, requiring only $N$ number of samples and a computational expense of $C$. In the later sections, we empirically verify that the policies from different local minima are diverse in nature.

#### 4.1 Learning policies

Learning rate annealing [15, 21] is shown to improve generalization and the use of high initial learning rate [12] determines the local minima for the model. We use cosine cyclical annealing learning rate schedule [17] to introduce learning rate annealing and learn multiple policies, as shown in Figure 1. Depending on the number of time-steps needed to train the agent, and the number of models needed for the ensemble, the learning rate schedule can be calculated.

As the learning rate anneals to a small value, the model converges to a local minimum, and we obtain the first policy. By increasing the learning rate, the model is perturbed along the gradient direction and dislodged from its local minima. In other words, if $M$ models are required, we split the training process into $M$ different training cycles where in each cycle the model starts at a high learning rate and anneals to a small value. The high learning rate is significant as it provides energy to the policy to escape the local minima and the learning rate annealing traps it into a well behaved local minima. The annealing helps in generalization and converging to a local minima while the directed perturbation re-orients the model to converge to a different local minima. The formulation is as follows:
lowest error would be the best policy. By having a higher degree, we are capturing the dependencies among the policies. The dependencies arise because the initialization of a new policy is at the termination of the previous policy and the policy network weights are shared.

Let us consider the computation of the total error for the case of A2C, the total error, \( L'(s, a) \), for a state action pair is the weighted sum of the policy gradient loss and the value function loss. \( V(s) \) and \( A(s, a) \) is the value function and the advantage function respectively at state \( s \) obtained by using policy \( \pi(a|s) \).

\[
V_{\text{loss}}(s) = r(s, a) + y V(s') - V(s) \tag{5}
\]

\[
\pi_{\text{loss}}(a|s) = -\log(\pi(a|s))A(s, a) \tag{6}
\]

\[
L'(s, a) = V_{\text{loss}}(s) + \pi_{\text{loss}}(a|s) + C_o \tag{7}
\]

\( y \) is the discount factor, \( C_o \) is the coefficient used for weighting the value function against the policy gradient loss.

By minimizing this objective function, we obtain the values of \( w_i \), the Lagrange multipliers to this optimization framework. To choose the best ensemble, we select the \( m \) policies with the highest corresponding lagrange multipliers.

### 4.2.2 Implementation detail

In order to solve the optimization problem, we can re-frame it as a quadratic programming problem with box constraints as follows:

\[
J(w) = w^T B w \tag{8}
\]

and,

\[
B_{ij} = \sum_{s \in S} \bar{p}(s)b_ib_j \tag{9}
\]

where,

\[
b_i = \left[ \frac{\sum_{a \in A} L(s, a) - \frac{\beta}{M - 1} \sum_{k \in M, k \neq i} \text{KL}(\pi_i(s)||\pi_k(s))}{\sum_{a \in A} L(s, a)} \right] \tag{10}
\]

The formulation is still subjected to all the linear constraints as earlier. The matrix \( B \) is a positive definite matrix since it is an inner product of the loss terms. This results in a convex objective function for which the global minimum can be found. In order to run this optimization, we select the states from multiple trajectories in the training samples. This framework efficiently reuses data and is better in comparison to evaluating the policies directly in the environment. Without this framework, the method to find the best ensemble subset would be to evaluate all possible combinations. For comparison, if each ensemble model is evaluated using \( N' \) samples from the environment, a total of \( m \times N' \) samples will be used in selecting the \( m \) policies.

### 4.3 Ensemble techniques

Once the \( m \) policies are chosen, depending on the complexity of the action space, discrete or continuous, there are multiple strategies to ensemble the actions in the environment. We divide the ensemble strategy into two categories, for discrete and continuous action spaces.

---

**Algorithm 1 SEERL**

1. **Input** Initialize a policy \( \pi_0 \), training time-steps \( T \), evaluation time-steps \( T' \), number of policies \( M \), maximum learning rate \( \alpha_0 \), number of policies to ensemble \( m \), ensemble strategy \( E \)
2. **Output** Average reward during evaluation
3. **Training**
   4. while \( t \leq T \) do
   5. Calculate the learning rate, based on the inputs to the cosine annealing learning rate schedule
   6. \( \alpha(t) = f(\alpha_0, t, T, M) \) //Equation 1
   7. Train the agent using \( a(t) \)
   8. if \( t \mod (T/M) \) then
   9. Save policy \( \pi_i^t \) for \( i = 1, 2, \ldots, M \)
   10. end if
   11. end while
12. **Evaluation**
13. Select the \( m \) policies using the Policy selection process //Section 4.2.1
14. Select an ensemble strategy \( E \)
15. while \( t \leq T' \) do
16. Collect actions from the \( m \) policies, \( a_1, a_2, \ldots, a_m \) for environment state \( s_t \)
17. Find the optimal action, \( a^* \) using \( E // \) Section 4.3
18. Perform action \( a^* \) on the environment
19. Obtain cumulative reward for the episode, \( r_t \) and the next state \( s_{t+1} \)
20. end while
21. return Average reward obtained during evaluation

---

### 4.3.1 Ensemble in discrete action spaces

In discrete action spaces, we consider majority voting as a good solution. Due to different fixed point convergences of value functions of algorithms trained independently, it is not possible to compare actions by their Q values.

\[
\pi(a|s) = \arg\max_{a \in A(s)} \sum_{i \in m} N_i(s, a) \tag{11}
\]

where \( N_i(s, a) \) is one if the agent \( i \) takes action \( a \) in state \( s \), else zero. It the case of a tie, random action is chosen among the set of actions having the tie.

### 4.3.2 Ensemble in continuous action spaces

In continuous action spaces, [5] proposes multiple strategies to find the optimal action. However, the performance comparison of the strategies is not provided and environments considered are too simple. The different strategies are as follows:

- **Averaging:** We take the average of all the actions as part of the ensemble. This strategy could fail in settings where one or more of the actions are extremely biased and thereby shifts the calculated value away from the true mean value.
- **Binning:** This is the equivalent of majority voting in a continuous action space setting. We discretize the action space into multiple bins of equal size and average the bin with the most number of actions. The average value obtained is the optimal action to take. Through this method, we have discretized the action space, sorted the bins based on its bin-count, and
calculated the mean of the bin with the highest bin-count.
The hyper-parameter to be specified here is the number of bins. We use five bins in our experiments
• Density-based Selection (DBS): This approach tries to search
  for the action with the highest density in the action space. Given $M$ action vectors, $a_i$ each of $k$ dimensions to be ensem-
  bled, we calculate the density of each action vector using
Parzen windows as follows:
\[
d_i = \sum_{j=1}^{M} \exp \left( -\frac{\sum_{l=1}^{k} (a_{il} - a_{jl})^2}{h^2} \right)
\] (12)
The action with the highest density, $d_i$, is selected as the final
action. The only parameter to be specified is $h$, the window
width, and we have chosen $h = 0.0001$ in our experiments.
• Selection through Elimination (STE): This approach elimi-
  nates action based on the Euclidean distance. We calculate
  the mean of the action vectors and compute the euclidean
distance to each action from the mean. The action with the
  largest euclidean distance is eliminated, and the mean is re-
  computed. The process is repeated until two actions remain.
  The final action is chosen as the average of the two actions.

5 EXPERIMENTS
Through our experiments, we answer the following questions:
• RQ1 : How does SEERL compare against traditional ensem-
  bles and SOTA reinforcement learning algorithms in terms
  of sample complexity and performance?
• RQ2 : How does the diversity among policies contribute to
  the final performance?
• RQ3 : How does the policy selection framework help to find
  the best subset of policies?
• RQ4 : How are the policies obtained from SEERL any different
  from those obtained through random perturbation?
In the following sections, we describe the setup used for experi-
mentation, the training procedure, the evaluation procedure and
analysis of the results obtained. We answer the questions posed
above in detail in the analysis section.

5.1 Setup
In order to answer the above questions, we consider the environ-
ments from the Atari 2600 game suite [3] for its discrete action
space and Mujoco [29] for its continuous action space. We conduct
three sets of experiments to validate our framework :
• The first set uses the following algorithms as the underlying
  reinforcement learning method (base learner) for SEERL -
  A2C [19], ACER [31], ACTKR [33], DDPG [16], SAC [9] and
  TRPO [24]. The goal is to show that SEERL can be adapted
to any reinforcement learning algorithm, both off-policy and
on-policy.
• The second set compares SEERL with three baseline ensem-
  ble methods that we have created. The three baselines are as
  follows:
  – B1 : Ensembles of policies trained independently from a
    single algorithm. E.g., Five models of A2C.
Table 1: Evaluation scores across Atari 2600 games averaged over 100 episodes. We report the published scores for DQN [20], Distributional DQN [2] and Rainbow DQN. SEERL uses A2C as base learner is the best performing of the methods, topping in 11 of the 15 games. All results represent the average of 5 random training runs. We run each game for 20M environment time steps.

| Game   | DQN | A2C | Distrib. DQN | Rainbow DQN | SEERL |
|--------|-----|-----|--------------|-------------|-------|
| Alien  | 634 | 518.4 | 1997.5 | 6022.4 | 1924 |
| Assault| 178.4 | 263.9 | 237.7 | 202.8 | 341.2 |
| Bank Heist | 312.7 | 970.1 | 835.6 | 826 | 1124.1 |
| Battle zone | 23750 | 12950 | 32250 | 52040 | 28580 |
| Breakout | 354.0 | 681.9 | 584.9 | 379.5 | 821 |
| Freeway | 26.9 | 0.1 | 28.8 | 29.1 | 33.1 |
| Frostbite | 496.1 | 190.5 | 2813.9 | 4141.1 | 1226 |
| Krull | 6206.0 | 5560.0 | 6757.8 | 6715.5 | 6795.2 |
| MsPacman | 1092.3 | 653.7 | 2064.1 | 2570.2 | 2614.2 |
| Pong | 18 | 5.6 | 18.9 | 19.1 | 19.8 |
| PrivateEye | 207.9 | 206.9 | 5717.4 | 1704.4 | 721.1 |
| Qbert | 9271.5 | 15148.8 | 15035.9 | 18397.6 | 18834.2 |
| Road Runner | 35215 | 34216 | 56086 | 54261 | 58624.2 |
| Robotank | 58.7 | 32.8 | 49.8 | 55.2 | 61.2 |
| Seaquest | 4216.7 | 2355.4 | 3275.4 | 19176.0 | 4811.2 |

Table 2: Evaluation scores across Mujoco environments averaged over 100 episodes. SEERL using SAC as base learner is seen to outperform across 5 of the 6 environments. All results represent the average of 5 random training runs.

| Environment | TRPO | PPO | DDPG | SAC | SEERL |
|-------------|------|-----|------|-----|-------|
| Ant         | 2342.2 | 962 | 342.4 | 1958 | 2564 |
| Half Cheetah | 4233 | 1820 | 5440 | 7269 | 11658 |
| Hopper      | 2252 | 1112 | 1233 | 3379.2 | 3156 |
| Humanoid    | 3882 | 735 | 101.4 | 4380 | 4845 |
| Swimmer     | 121.2 | 42.4 | 43.4 | 44.6 | 162.4 |
| Walker 2D   | 3215 | 1892 | 782 | 2112 | 4366 |

- B2: Ensembles of policies trained independently from different algorithms. E.g., two models of A2C, two models of ACER and one model from ACKTR.
- B3: Ensembles of policies generated from random perturbation of model parameters at regular intervals. E.g., Five models of A2C, each of which has been obtained by perturbing at regular intervals and saving the parameters.

The goal is to understand how the traditional ensembles (B1 and B2) compare to SEERL and the role of diverse policies in an ensemble. Comparison with B3 is intended to show the importance of directed perturbation against random perturbation.

- The third and final set compares SEERL with SOTA reinforcement learning algorithms that aim to be sample efficient. We compare SEERL with SimPLe [13], CURL [28], Rainbow DQN [10] and SUNRISE [14]. In these experiments SEERL uses Rainbow DQN as the base learner in order to make a fair comparison with SUNRISE. The goal is to understand whether SEERL can be competitive with these algorithms that are sample efficient and be trained using the limited interactions it has with the environment. Comparison with SUNRISE is intended to understand whether SEERL is competitive with a method that trains multiple critics in parallel.

5.2 Training

SEERL uses a base learner to learn an ensemble of policies. This base learner can be any reinforcement learning algorithm such as A2C, SAC or Rainbow DQN. After selecting the base learner, SEERL is trained using the same hyper-parameters configurations as in the original implementation of the base learner. The ambiguity that SEERL will lead to poor convergence as a result of shifting from zero to the maximum learning rate multiple times is mitigated through our results. SEERL performance during training is at least at par or better than the base learner, as shown in Figure 2(a, b). We train the models across different values of $M$ ranging from 3 to 9. In order to make the comparison of SEERL with the base learner fair, we train both models on the same number of timesteps as specified in the original paper of the base learner. However, for comparison with SUNRISE, SimPLe, CURL and Rainbow DQN, we train SEERL using Rainbow DQN as base learner for 100k timesteps for a fair comparison. Additional training results on the environments are presented in the supplementary under section 1.

5.3 Evaluation

During evaluation, the policy selection framework is used to select the best subset of policies for the ensemble with $m = 5$. All $m$ policies are loaded in parallel and provided with an observation from the environment. Based on the observation, every policy outputs an action, and the ensemble strategy decides the final action to be used on the environment. We perform this evaluation process for 100 episodes, and the average reward over these 100 episodes is
Table 3: Evaluation scores across Atari 2600 games averaged over 10 episodes using Rainbow DQN [10] as base learner for SEERL at 100K interactions. We report the published scores for SimPLe [13], CURL[28] and SUNRISE[14] at 100K interactions. All results represent the average of 3 random training runs.

| Game        | Rainbow DQN | SimPLe | CURL          | SUNRISE | SEERL  |
|-------------|-------------|--------|---------------|---------|--------|
| Allen       | 789         | 616.9  | 558.2         | 872     | 800    |
| Amidar      | 118.5       | 88.0   | 142.1         | 122.6   | 208.3  |
| Assault     | 413.0       | 527.2  | 600.6         | 594.8   | 627.7  |
| BankHeist   | 97.7        | 34.2   | 131.6         | 266.7   | 508.0  |
| BattleZone  | 7833.3      | 5184.4 | 14870.0       | 15700.0 | 19400.0 |
| Breakout    | 2.3         | 16.4   | 4.9           | 1.8     | 3.2    |
| Freeway     | 28.7        | 20.3   | 26.7          | 30.2    | 31.3   |
| Frostbite   | 1478.3      | 254.7  | 1181.3        | 2026.7  | 2010.0 |
| Krull       | 3282.7      | 4539.9 | 4229.6        | 3171.9  | 3203.1 |
| MsPacman    | 1118.7      | 1480   | 1465.5        | 1482.3  | 1838.0 |
| Pong        | -16.9       | 12.8   | -16.5         | -13.8   | -14.2  |
| PrivateEye  | 97.8        | 58.3   | 218.4         | 100     | 100    |
| Qbert       | 646.7       | 1288.8 | 1042.4        | 1830.8  | 2125   |
| RoadRunner  | 9923.3      | 5640.6 | 5661.0        | 11913.3 | 15290.0 |
| Seaquest    | 396.0       | 683.3  | 384.5         | 570.7   | 458.0  |

5.4 Analysis

We try to understand why and how SEERL gives such superior performance in comparison to baselines. We analyze the performance and sample efficiency of SEERL, the diversity among the policies, and finally, the comparison between a randomly perturbed model and SEERL. Analysis on individual performance of the SEERL policies and the dominance among policies in the ensemble are presented in the supplementary material under section 2.4.

5.4.1 RQ1: Performance and sample efficiency of SEERL.

The training curve in Figure 2(a,b) illustrates the performance of SEERL with the base learner and is observed to consistently outperform it. In Figure 2(a), it can be observed that SEERL achieves the same performance in 6 million timesteps which requires the base learner to achieve in 20 million timesteps. Figure 2(c) and 2(d) illustrates the performance of SEERL in comparison with B1, B2 and is seen to outperform both baselines. Table 1 shows that SEERL outperforms the SOTA score from Rainbow DQN on most games. Similarly Table 2 shows that SEERL outperforms across 5 of the 6 environments. The results in Table 3 illustrate comparison of SEERL with Rainbow DQN, SimPLe, CURL and SUNRISE at 100k timesteps. We observe that SEERL outperforms SUNRISE in 11 of the 15 Atari games used for comparison and achieves SOTA score for 100k interactions in 9 games. The performance of SEERL trained on 100k interactions highlights its sample efficiency. It is also observed that SEERL is 4-6 times faster than SUNRISE since it only uses a single network to sequentially train the policies while SUNRISE has multiple networks in parallel. As the number of networks grows in SUNRISE, the training time is longer while for SEERL, it remains the same. To summarize, SEERL outperforms most methods whether it is trained on millions of interactions or just 100k.

5.4.2 RQ2 and RQ3: Diversity of Policies and Policy Selection in SEERL.

We establish the diversity among the individual policies concretely by computing the KL divergence between them using the action distribution across a diverse number of states. The higher the KL divergence between the policies, the more diverse the policies are. From Figure 3(b), we can observe that the SEERL policies are diverse, and diversity continues to exist as new models are formed. Conversely, for the baseline models, B1 and B3 (Figure 3(a) and Figure 3(c)), the KL divergence between the policies is substantial. This observation can be used to explain why the baseline ensembles failed to perform. The larger KL divergence among the baseline policies indicate that the policies do not have much overlap in the action space, and hence ensemble techniques such as majority voting were unable to find a good action. Additionally, the analysis of the different subsets of policies in Figure 5 and their performance in Figure 2(e) confirms that diversity among policies is required, but too much diversity does not help. This empirically proves that extremely diverse models are not that helpful in an ensemble.

We can, therefore conclude that SEERL can generate policies with sufficient diversity for a good ensemble.

5.4.3 RQ4: Random perturbation vs. SEERL.

To emphasize that directed perturbation of the weights will lead to better models, we show the comparison between SEERL and a randomly perturbed model in Figure 4(c). This model has been perturbed with random values of gradients instead of the actual values. We observe that doing a single perturbation...
5.5 Ablation studies

5.5.1 Effect of varying the number of cycles. The performance of SEERL is affected by the selection of \( M \). For a fixed training budget, if the value of \( M \) chosen is very large, the performance is seen to degrade. With larger \( M \), the training cycle for each policy is reduced, thereby reducing the chance for the policy to settle at a good local minimum before it is perturbed again. In practice, we find that setting the value of \( M \) between 3 to 7 works reasonably well. Figure 4(a) compares the performance of SEERL with varying \( M \) values between 3 and 9.

5.5.2 Effect of varying maximum learning rate. The maximum learning rate value influences the performance of the policies and therefore affects the performance of SEERL. It directly impacts the perturbation of the local minima and hence, the diversity of the policies. In practice, we have seen that having a larger value tends to perform better, owing to the strong perturbation it causes at different local minima leading to reasonably different policies. We have used values ranging between 0.01 to 0.001 throughout our experiments. Figure 4(b) compares the performance of SEERL with different values of \( \alpha \) with \( M = 5 \).

6 CONCLUSION AND FUTURE WORK

In this paper, we introduce SEERL, a framework to ensemble multiple policies obtained from a single training run. We show that the policies learned at the different local minima are diverse in their performance and our policy selection framework helps to select the best subset of policies for the ensemble during evaluation. SEERL outperforms the three baseline methods and beats SOTA scores in complex environments having discrete and continuous action spaces. We show our results using both off-policy and on-policy reinforcement learning algorithms and therefore showcase the scalability of the framework. Our analysis shows that, in comparison to baselines, SEERL achieves comparable, and sometimes better performance using a significantly low number of samples, making it an extremely sample efficient algorithm. Future work will explore how to combine the learned policies during training time as a growing ensemble to stabilize training and increase diversity.
REFERENCES

[1] Oron Anschel, Nir Baram, and Nahum Shimkin. 2017. Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning. In *International Conference on Machine Learning*. PMLR, 176–185.

[2] Marc G Bellemare, Will Dabney, and Rémi Munos. 2017. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*. PMLR, 449–458.

[3] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of Artiﬁcial Intelligence Research* 47 (2013), 253–279.

[4] Thomas G Dietterich. 2000. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*. Springer, 1–15.

[5] Siegmund Duell and Steffen Udluft. 2013. Ensembles for Continuous Actions in Reinforcement Learning. In *ESANN*.

[6] Stefan Faußer and Friedhelm Schwenker. 2015. Selective neural network ensembles in reinforcement learning. *Neural Processing Letters* 41, 1 (2015), 55–69.

[7] Stefan Faußer and Friedhelm Schwenker. 2015. Selective neural network ensembles in reinforcement learning: taking the advantage of many agents. *Neurocomputing* 169 (2015), 350–357.

[8] Vukosi Ntsakisi Marivate and Michael Littman. 2013. An ensemble of linearly factored approximation. In *International Conference on Machine Learning*. PMLR, 1769–1776 (2013), 350–354.

[9] Vukosi Ntsakisi Marivate and Michael Littman. 2013. A Simple Unified Framework for Ensemble Learning in Deep Reinforcement Learning. *arXiv preprint arXiv:2004.04136* (2020).

[10] Orai Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, An-Sung Ahn, Yuhuai Wu, Elman Mansimov, Shun Liao, Roger B. Grosse, and Jimmy Ba. 2017. Rainbow: Combining improvements in deep reinforcement learning. *arXiv preprint arXiv:1710.02293* (2017).

[11] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. Mujoco: A physics engine for model-based control. In *Robots and Systems*.

[12] Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2019. Mastering atari, go, chess and shogi by planning with a learned model. *arXiv preprint arXiv:1911.08265* (2019).

[13] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International conference on machine learning*. PMLR, 1889–1897.

[14] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2015. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438* (2015).

[15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–535.

[16] Preetum Nakkiran. 2020. Learning Rate Annealing Can Provably Help Generalization, even for Convex Problems. *arXiv preprint arXiv:2005.07360* (2020).

[17] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. 2016. Deep exploration via bootstrapped DQN. In *Advances in neural information processing systems*. 4026–4034.

[18] Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–535.