SEERL: Sample Efficient Ensemble Reinforcement Learning

Rohan Saphal[1*] , **Balaraman Ravindran**[1] , **Dheevatse Mudigere**[2†] , **Sasikanth Avancha**[2] and **Bharat Kaul**[2]

[1]Indian Institute of Technology Madras

[2]Parallel Computing Lab, Intel Labs

rohansaphal@gmail.com, ravi@cse.iitm.ac.in, dheevatsa@gmail.com, sasikanth.avancha@intel.com, bharat.kaul@intel.com

Abstract

Reinforcement learning algorithms are sensitive to hyper-parameters and require tuning and tweaking for specific environments for improving performance. Ensembles of reinforcement learning models on the other hand are known to be much more robust and stable. However, training multiple models independently on an environment suffers from high sample complexity. Leveraging the theory of cyclic learning rates from deep learning, we present here a methodology to create multiple models from a single training instance that can be used in an ensemble. This allows training a single model that converges to several local minima during its optimization process. By saving the model parameters (policy) at each such instance, we obtain multiple policies during training that are ensembled during evaluation. 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 other baseline approaches.

1 Introduction

Deep reinforcement learning over the years has made considerable advancements applicable across a variety of domains. From learning to play Atari 2600 suite from raw visual inputs[Mnih *et al.*, 2015], mastering the game of Go [Silver *et al.*, 2017], learning locomotion skills for robotics [Schulman *et al.*, 2015b; Schulman *et al.*, 2015a; Lillicrap *et al.*, 2017] and most recently, the development of Alpha Fold [Evans *et al.*,] to predict the 3D structure of a protein solely based on its genetic sequence, have all pushed the boundaries of research in reinforcement learning.

However, the task of creating a single agent that can give robust and stable performance across multiple environments without having to tune and tweak the hyper-parameters is not a trivial one. The more tuning and tweaking required for an agent, the less "autonomous" it is and reduces the chance for it to generalize to novel environments.

Typically, the performance of an agent on an environment is improved by, modification of neural network architectures to learn better state representations, trying different activation functions, reward scaling and different weight initialization for the neural network architecture . However, this trial and error approach to improving the agent's performance simply cannot be used in practice owing to high sample complexity and computational expense involved. Ensemble methods,however, are known to be extremely valuable for tackling complex problems. The relative success of ensemble methods can be attributed to its ability to tackle a wide range of instances that require different low-level approaches.

Traditionally, the idea of using ensembles in reinforcement learning settings is associated with combining multiple value functions or policies from different models. These models could be the same algorithm trained across different hyper-parameter settings or different algorithms altogether.Training multiple such models is an approach that simply cannot be used in practice.

Our work tackles the above drawbacks by leveraging the theory of cyclic learning rates to learn multiple models from a single training instance. Stochastic gradient descent [Bottou, 2010] and its accelerated variants [Kingma and Ba, 2014; Duchi *et al.*, 2011] have become the de-facto approach to optimizing neural networks and have become popular for its ability to avoid local minima and spurious saddle points [Dauphin *et al.*, 2014].[Huang *et al.*, 2017] have shown that these local minima contain useful information that can improve the model performance. Although there are both good and bad local minima,[Keskar *et al.*, 2016] argue that local minima with flat basins tend to generalize better. At larger learning rates, the random motion across gradient steps prevents the optimizer from reaching any of the sharp basins and moves into the general vicinity of the local minima. Lowering the learning rates at such an instance leads the optimizer to converge to some final local minima. We leverage the diversity of the policies learned at these different local minima for the ensemble. Our main contributions are:

- A sample efficient framework for learning M diverse models from a single training instance at no additional cost.
- A practical approach to selecting robust policies, from a

*Contact Author

†Author is now affiliated with Facebook.

diverse set, for ensemble

- Performance of various ensemble strategies for discrete and continuous action spaces.

Since we use models from a single training instance instead of training M different models independently from scratch, we refer to our approach as Sample Efficient Ensemble Reinforcement Learning (SEERL).

2 Related work

There has recently been a good body of work on using ensembles for reinforcement learning. Most of the work is associated with ensembles during the training phase in order to reduce the variance and improve the robustness of the policy being learned.

[Anschel *et al.*, 2017] trains multiple Q networks in parallel with different weight initialization and averages the Q values from all the different networks in order to reduce variance. This results in policies being learned that outperforms baselines and 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.

Another interesting work falls under the umbrella of model-based reinforcement learning [Kurutach *et al.*, 2018]. Multiple neural networks are initialized to learn a model of the environment using samples from the real system. Although the framework is sample efficient, it is costly to train multiple models. Using cycling learning rates, we could obtain multiple models from a single training instance for training the agent at no additional computation costs.

Earlier works [Wiering and Van Hasselt, 2008; Duell and Udluft, 2013; Faußer and Schwenker, 2015a; Faußer and Schwenker, 2015b] 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 fixed points and thereby have different bias and variance.[Marivate and Littman, 2013] tries to tackle this problem by having a meta-learner linearly combine the value functions from the different algorithms during training time so as to adjust for the inherent bias and variance. Although training multiple algorithms in parallel is sample efficient, it is still far away from being a practical approach.

Our method combines the best of both approaches and improves the performance of the algorithm by balancing sample complexity with the computational expense. Our work is inspired by the recent findings of [Smith, 2015; Loshchilov and Hutter, 2016; Huang *et al.*, 2017], who showed that cycling learning rates are effective for training convolutional neural networks and ensembling in supervised learning settings. The authors show that each cycle produces models which are almost competitive to those learned with traditional learning rate schedules. Though the models are seen to temporarily suffer in performance once the cycle restarts, they eventually surpass the previous cycle as the learning rate is annealed. The authors suggest that cycling the learning rate perturbs the parameters of a converged model, allowing the model to find a better local minimum. [Huang *et al.*, 2017] have shown that there is significant diversity in the local minima visited during each cycle and exploiting this diversity using ensembles can lead to better performance.

3 Preliminaries

Reinforcement learning deals with sequential decision making and considers 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, ρ_t, r, γ), where S denotes the set of states, A denotes the set of actions, $P : S \times A \rightarrow$ S the transition function,ρ_t, the probability distribution over the initial states, $r : S \times S' \times A \rightarrow$ R, the reward function and γ the discount factor. A policy dictates the behaviour of an agent at a particular state in an environment. More formally, a policy is defined by $\pi(s) : S \rightarrow P(A)$ where P(A) denotes the probability distribution over actions $a \in A$ in a state $s \in S$. The objective of the agent is to maximize the discounted return $R_t = \Sigma_{i=t}^{T} \gamma^{i-t} r(s_i,s_{i+1},a_i)$, where $r(s_i,s_{i+1},a_i)$ is the reward function.

4 Sample Efficient Ensemble Reinforcement Learning

SEERL results in an ensemble of diverse policies obtained from a single training instance. Unlike supervised learning, where the dataset can be reused for training different models, in reinforcement learning, the agent has to interact with the environment to learn. The resultant markov chain cannot be reused for training another model free agent from scratch. Therefore, training multiple agents independently for ensemble suffers 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 $M \times C$ in computational cost. If trained in parallel, only N samples are required but the computational cost remains at $M \times C$. Though training multiple agents in parallel is a sound solution to tackle sample complexity, it is computationally expensive and limits the diversity among the learned policies, since every policy observes the same state at each instance.

Our approach saves policies during training at periodic intervals when the learning rate anneals to a small value and ensembles them during evaluation time. SEERL requires only N number of samples, the computational expense is C and yet we obtain M models for the ensemble. Since the policies have been saved at different local minima, the policies are diverse in nature.

4.1 Cycling cosine annealing

We use the cycling learning rate schedule proposed by [Loshchilov and Hutter, 2016]. Depending on the number of epochs to train the agent and the number of models needed for the ensemble, the learning rate schedule is calculated. As the learning rate anneals to a small value, the model converges to a local minimum and the first policy is obtained. By increasing the learning rate, the model is perturbed and dislodged from its local minima. In other words, if M models required, we split the training process into M different training cycles

wherein each cycle the model starts at a large learning rate and anneals to a small value. The large learning rate is significant as it provides energy to the policy to escape the local minima and the small learning rate traps it into a well behaved local minima. The formulation is as follows:

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

where α_0 is the initial learning rate, t is the episode number and T is the total number of episodes for which the agent is trained.

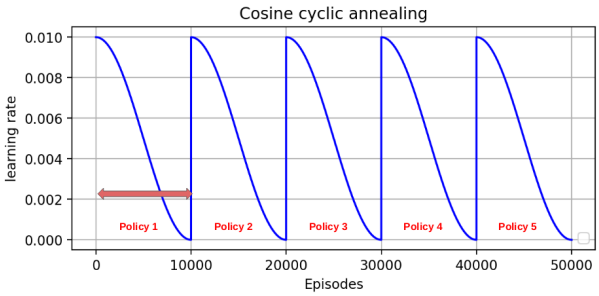


Figure 1: Cycling cosine annealing learning rate schedule. α_0 is set at 0.01, number of models $M = 5$ and training episodes $T = 50000$

4.2 Learning policies

As training starts, a single policy is initialized and the learning rate starts at a large value. Given the algorithm is required to train for T epochs and we would like to acquire M models, the learning rate starts annealing until it reaches its lowest value at T/M epochs. The first policy is saved at this instance and at $(T/M + 1)$ epoch, the learning rate resets to its high value. M models are obtained at the end of the training cycle using this learning rate schedule. It is possible to learn multiple policies using other learning rate schedules but it is difficult to ascertain whether the policies learned will be diverse. In the case of following a step-wise learning rate schedule where the learning rate decreases after a certain number of epochs, it is possible to save the model at each of the instances where the learning rate steps down. However, each of the policies learned in this process could be from the same local minima. Cycling learning rates on the other hand completely remove this ambiguity.

4.3 Policy selection

The policies obtained during the training instance are diverse in nature and of different performance capabilities. Ideally, the best m policies are selected for the ensemble so as to avoid bias from the poor policies. We can obtain the performance capacity of each policy by evaluating the policy in the environment. However, evaluating multiple policies requires samples from the environment and is not a practical approach to selecting the m ($m \leq M$) policies. We make a calculated decision to choose the last m of M trained policies. Since the final M^{th} policy is closest to the optimal policy and the $(M-1)^{th}$ policy is naturally the closest in comparison to other policies, to the optimal policy. Hence choosing the last m policies is justified. By compromising on a fraction of improvement in performance, SEERL becomes highly sample efficient even during evaluation phase. However, for our baseline methods, it is not evidently clear how to select the m policies from the total M without evaluating it individually in the environment. If each policy is evaluated using the method suggested by [Faußer and Schwenker, 2015b] using N' samples from the environment, a total of $M \times N'$ samples are used up in selecting the m policies.

Algorithm 1 SEERL

Input: Intialize a policy π_θ, training episodes T,evaluation episodes T', number of policies M, maximum learning rate α_0, number of policies to ensemble m, ensemble strategy E, D to store the policies
Output: Average reward during evaluation
Training
while $t \leq T$ **do**
  Calculate the learning rate, based on the inputs to the cosine annealing learning rate schedule f
  $\alpha(t) = f(\alpha_0, t, T, M)$
  Train the agent and optimize using $\alpha(t)$
  if t mod (T/M) **then**
    Save policy π_θ^i in D for $i = 1, 2 \ldots, M$
  end if
end while
Evaluation
Select the last $m(m \leq M)$ policies from D
Select an ensemble strategy E
while $t \leq T'$ **do**
  Collect actions from the m policies, $a_1, a_2 \ldots, a_m$ for environment state s
  Find the optimal action, a^* using E
  Perform action a^* on the environment
  Obtain cumulative reward for the episode, r_t
end while
return Average reward obtained during evaluation

4.4 Ensemble at evaluation time

The policies learned during training are ensembled strategically during evaluation phase. We compare our work with two other baseline methods. If there are A algorithms and M policies are required for ensemble, our first baseline would consist of M independently trained policies from a single algorithm with various hyper-parameter settings. The second baseline would consist of M/A independently trained policies from each of the A algorithms. Policy selection could be applied over the set of policies and the best m can be chosen.

5 Ensemble techniques

Once the m policies are chosen,selecting the optimal ensemble strategy is a challenging task. Depending on the complexity of the action space, discrete or continuous, there are multiple strategies to ensemble the actions in the environment. When building ensembles, each of the m policies are loaded

in parallel and provided with an observation from the environment. Based on the observation, every policy outputs an action. The ensemble strategy decides which action to select based on the available set of actions. We divide the ensemble strategy into two categories, based on the complexity of the action space, into strategies for discrete and continuous action spaces.

5.1 Ensemble in discrete action spaces

In discrete action spaces, majority voting is considered 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.

$$\pi(a|s) = \underset{a \in A(s)}{\operatorname{argmax}} \left[\sum_{m \in M} N_m(s,a) \right]$$

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

5.2 Ensemble in continuous action spaces

In continuous action spaces,[Duell and Udluft, 2013]proposes multiple strategies to find the optimal action. However, performance comparison of the strategies is not provided and environments considered are too simple. The different strategies are as follows:

- Averaging: This is a naive approach to finding the optimal action. We take the average of all the actions as part of the ensemble. In case of vector actions, the average is calculated for each of the vector dimensions. 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 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 simply 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 only parameter to be specified here is the number of bins to be selected.
- Density based Selection : This approach tries to search for the action with the highest density in the action space. Given M action vectors, a, each of k dimensions to be ensembled, we calculate the density of each action vector using Parzen windows as follows:

  $$d_i = \sum_{j=1}^{M} e^{-\frac{\sum_{l=1}^{k}(a_{il}-a_{jl})^2}{r^2}}$$

  The action with the highest density is selected as the final action. The only parameter to be specified is r and we have chosen $r = 0.0001$ in our experiments.
- Selection through Elimination: This approach eliminates action based on the Euclidean distance. We calculate the mean of all the action vectors and compute the euclidean distance to each of the actions 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.

6 Experiments

Through our experiments, we answer the following questions:

- How does SEERL compare against traditional ensembles in terms of sample complexity and final performance?
- Performance of the different ensemble strategies in continuous action spaces

6.1 Environments and Algorithms

To answer the above questions, we evaluate our method over environments having raw visual inputs and vector observations. We consider the Breakout environment from the Atari 2600 game suite and Half Cheetah from Mujoco [Todorov *et al.*, 2012]. We conduct our experiments on the following algorithms, A2C [Mnih *et al.*, 2016], ACER [Wang *et al.*, 2016], ACTKR [Wu *et al.*, 2017], DDPG [Lillicrap *et al.*, 2017], PPO [Schulman *et al.*, 2017] and TRPO [Schulman *et al.*, 2015a]

6.2 Training and evaluation setup

Each algorithm is trained across multiple seeds and different hyper-parameter settings. We choose the seed and hyper-parameters of the best performing combination for training SEERL and we compare with the same. The baseline methods use linearly decreasing learning rate schedule throughout all the experiments and the maximum learning rate value is the same in both cases.

6.3 Results

Training results

We compare the training performance of SEERL with the baseline training strategy in Figure 2. The ambiguity that cosine cycling annealing learning rate 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 par and sometimes better than the baseline.

Evaluation results

We present two baseline ensemble methods, B1 and B2, to compare against SEERL. B1 ensembles multiple independently trained policies from the same algorithm. B2,on the other hand, ensembles multiple independently trained policies from different algorithms. The main results of SEERL with $m = 5$ and its performance comparison with baseline methods is summarized in Table 1 and Table 2. Each of the methods is evaluated in the true environment for 20 episodes and the average reward is shown.We observe that Binning and Density based strategies seem to perform well in most cases,

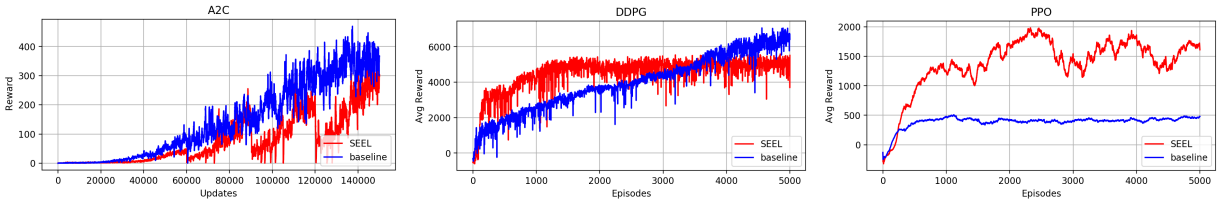


Figure 2: Comparison between SEERL and baseline during training. Breakout training performance on A2C and Half Cheetah training performance on DDPG and PPO is shown

followed by Selection through Elimination and finally Averaging. Experiments show that SEERL is at par with or outperforms baseline methods during evaluation. Our framework is therefore much more sample efficient than baseline methods.

Method	Algorithm	MV
SEERL	A2C	**340**
	ACER	352
	ACKTR	**372**
B1	A2C	331
	ACER	356
	ACKTR	360
B2	A2C (1)	
	ACER (2)	358
	ACKTR (2)	

Table 1: Comparison of SEERL with baseline ensemble methods during evaluation in Breakout. Majority voting (MV) is employed as the ensemble strategy. The number shown in brackets for algorithms in B2 indicates the number of models of the algorithm used for ensemble.

Method	Algorithm	AVG	BNG	DB	STE
SEERL	DDPG	4722	**5112**	4961	4882
	PPO	1330	**1492**	1360	1394
	TRPO	604	590	**612**	595
B1	DDPG	4360	**4975**	4680	4960
	PPO	540	570	**640**	512
	TRPO	560	**632**	578	584
B2	DDPG (1)				
	PPO (2)	1150	**1340**	1284	1240
	TRPO(2)				

Table 2: Comparison of SEERL with baseline methods in Half Cheetah. Different ensemble strategies, Averaging(AV),Binning(BNG), Density based(DB) and Selection through Elimination(STE) are employed during evaluation.

Performance of individual policies

We evaluate the individual policies obtained with SEERL and baselines to gauge their performance. The analysis helps in understanding how the diversity in policies improve performance during ensemble. We evaluate the individual policies for the case, where $M = 5$. By understanding the performance of individual policies, we can justify our policy selection strategy of choosing the last m ($m \leq M$) of M policies. Table 3 and Table 4 show the analysis for discrete and continuous control tasks.

Method	Algorithm	Policy no.	Performance
SEERL	A2C	1	25
		2	80
		3	144
		4	192
		5 (final policy)	**296**
B1	A2C	1	305
		2	290
		3	312
		4	330
		5	278
B2	A2C	1	314
	ACER	2	330
	ACER	3	345
	ACKTR	4	372
	ACKTR	5	**388**

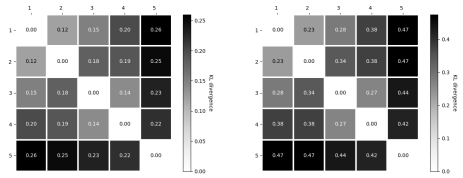
Table 3: The performance of individual policies used during ensemble for Breakout. For B1, the five policies are from A2C trained independently. For B2, the five policies are from the different algorithms trained independently

Diversity of Policies

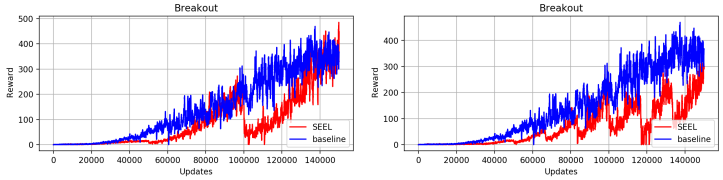
The performance of SEERL depends on the diversity of the individual policies being learned. We have shown earlier, the performance of the individual policies being learned and the diverse nature in their individual performance. We hope to establish more concretely the diversity of the individual policies by understanding the action distribution across states for each policy. We compute the KL divergence between the policies based on the action distribution across a number of states. The greater the KL divergence between the policies, the more diverse the policies are. We see that the KL divergence between policies in SEERL is larger in magnitude and varies sharply between policies. We present the pairwise KL divergence between A2C policies in Fig 3a

Effect of varying the number of cycles

The performance of SEERL is significantly affected through the selection of M. Given a fixed training budget(number of episodes to train the agent), there is a certain range of values of M that can be chosen to achieve good performance. For a fixed training budget, if the value of M chosen to be is very



(a) **Left**: Baseline. **Right**: SEERL



(b) Training performance of SEERL with A2C as number of cycles (M) vary

Figure 3

Method	Algorithm	Policy no.	Performance
SEERL	DDPG	1	3890
		2	4400
		3	4322
		4	4954
		5 (final policy)	4718
B1	DDPG	1	3834
		2	4650
		3	5200
		4	3924
		5	4340
B2	DDPG	1	5200
	PPO	2	690
	PPO	3	880
	TRPO	4	544
	TRPO	5	612

Table 4: The performance of individual policies used during ensemble for continuous action space is evaluated. For B1, the five policies are from DDPG trained independently. For B2, the five policies are from the different algorithms trained independently

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 to a good local minima before it is perturbed again. In practice we find that setting the value of M between 3 to 7 works reasonably well. Fig 3b and Table 5 compares the performance of SEERL varying M

M	Performance
3	340
5	**392**
7	322
9	308

Table 5: Performance of SEERL on Breakout as the number of cycles (M) varies

Effect of varying maximum learning rate value

The maximum leaning 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 being learned. In practice, we have seen that having a larger value tends to perform better, owing to the string perturbation it causes at different local minima leading to diverse policies. We have used a value ranging between 0.01 to 0.0001 throughout our experiments. Figure 4 and Table 6 compares the performance of SEERL with different values of α_0 and $M = 9$

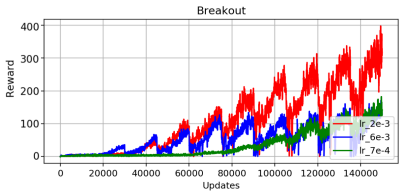


Figure 4: Training performance of SEERL with A2C as maximum learning rate (α_0) varies

α_0	Performance
0.006	158
0.002	**374**
0.0007	122

Table 6: Evaluation performance of SEERL with A2C on Breakout as maximum learning rate (α_0) varies

7 Conclusion and Future work

In this paper, we introduce SEERL, a framework to ensemble multiple policies obtained from a single training instance. We utilize the ability of SGD to converge to and jump from local minima as the learning rate cycles between low and high values. Through our experiments, we show that the policies learned at the different local minima are diverse in their performance and hence are well suited for ensemble. SEERL outperforms two commonly used baseline methods in complex environments having discrete and continuous action spaces. We show our results using various reinforcement learning algorithms and therefore show that it is not limited to its performance in any particular setting. Future work will explore better ways to select the best performing policies from the entire set instead of selecting the last m of M models. We would also investigate how to combine the learned policies during training time as a growing ensemble to stabilize training.

References

[Anschel *et al.*, 2017] Oron Anschel, Nir Baram, and Nahum Shimkin. Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning. In *Proceedings of*

the 34th International Conference on Machine Learning-Volume 70, pages 176–185. JMLR. org, 2017.

[Bottou, 2010] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[Dauphin *et al.*, 2014] Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in neural information processing systems*, pages 2933–2941, 2014.

[Duchi *et al.*, 2011] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[Duell and Udluft, 2013] Siegmund Duell and Steffen Udluft. Ensembles for continuous actions in reinforcement learning. In *ESANN*, 2013.

[Evans *et al.*,] R Evans, J Jumper, J Kirkpatrick, L Sifre, TFG Green, C Qin, A Zidek, A Nelson, A Bridgland, H Penedones, et al. De novo structure prediction with deeplearning based scoring. *Annu Rev Biochem*, 77:363–382.

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

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

[Huang *et al.*, 2017] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*, 2017.

[Keskar *et al.*, 2016] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *CoRR*, abs/1609.04836, 2016.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Kurutach *et al.*, 2018] Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. *arXiv preprint arXiv:1802.10592*, 2018.

[Lillicrap *et al.*, 2017] Timothy Paul Lillicrap, Jonathan James Hunt, Alexander Pritzel, Nicolas Manfred Otto Heess, Tom Erez, Yuval Tassa, David Silver, and Daniel Pieter Wierstra. Continuous control with deep reinforcement learning, January 26 2017. US Patent App. 15/217,758.

[Loshchilov and Hutter, 2016] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[Marivate and Littman, 2013] Vukosi Ntsakisi Marivate and Michael Littman. An ensemble of linearly combined reinforcement-learning agents. In *Workshops at the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[Mnih *et al.*, 2015] 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. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[Mnih *et al.*, 2016] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *CoRR*, abs/1602.01783, 2016.

[Schulman *et al.*, 2015a] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

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

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[Silver *et al.*, 2017] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[Smith, 2015] Leslie N Smith. No more pesky learning rate guessing games. *arXiv preprint arXiv:1506.01186*, 2015.

[Todorov *et al.*, 2012] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[Wang *et al.*, 2016] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Rémi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. *CoRR*, abs/1611.01224, 2016.

[Wiering and Van Hasselt, 2008] Marco A Wiering and Hado Van Hasselt. Ensemble algorithms in reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(4):930–936, 2008.

[Wu *et al.*, 2017] Yuhuai Wu, Elman Mansimov, Shun Liao, Roger B. Grosse, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. *CoRR*, abs/1708.05144, 2017.