REINFORCE SUN: A New Superclass of Reinforcement Learning Algorithms with Stochastic Synapses, Units, or Networks and Its Application to Continuous Environments

Syed Naveed Hussain Shah[1] , **Dean Frederick Hougen**[2]
[1]Microsoft Corporation, One Microsoft Way Redmond, WA 98052-6399
[2]Robotics, Evolution, Adaptation, and Learning Laboratory, University of Oklahoma
sayyed.naveed@gmail.com, hougen@ou.edu

Abstract

For decades, many of the leading algorithms for reinforcement learning in continuous action spaces have been members of the general class of connectionist methods known as REINFORCE that use stochastic units to sample the weight space and follow gradients to higher rewards. REINFORCE algorithms continue to form the core of many of today's most sophisticated and successful reinforcement learning deep neural networks and any improvements to the REINFORCE framework could have profound implications for learning in challenging reinforcement environments such as those with multiple dimensions. This paper introduces REINFORCE SUN, a superclass that generalizes REINFORCE to allow for stochasticity at the level of the synapse, the unit, or the network, and considers the advantages of each. Empirical results using multi-dimensional synthetic and robot forward and inverse kinematic data sets show that in continuous state and action spaces synapse-level stochasticity greatly outperforms traditional unit-level stochasticity over time.

1 Introduction

Reinforcement learning (RL) is one of the most widely studied types of machine learning, after supervised and unsupervised learning [Sutton and Barto, 2018]. In RL, the learning agent attempts to formulate a policy mapping perceived states to actions in order to maximize the reward gained from the environment over time, which results in a self-learning algorithm with important real-world applications.

1.1 Conceptual Overview

Classic RL algorithms, going back to foundational approaches such as BOXES [Michie and Chambers, 1968], and extensively studied approaches such as temporal difference learning (TD) [Sutton and Barto, 1981; Sutton, 1988] and Q-learning [Watkins, 1989; Watkins and Dayan, 1992] have focused primarily on discrete states and action.

Nonetheless, there has been substantial research on RL in continuous state and action spaces [Recht, 2018]—a more difficult problem due to the unlimited possible input and output values, compounded by their myriad combinations, and consequent need to approximate the policy.

To learn in such environments, policy gradient approaches such as REINFORCE [Williams, 1992] are typically used. These methods sample continuous values that are used to influence the agent's output and follow gradients based on reward values received.

These approaches are generally paired with connectionist systems, aka artificial neural networks (ANNs), that may be shallow or deep, depending on application, intended to provide appropriate policy approximation and generalization.

Today, REINFORCE forms the core of many of the most sophisticated RL algorithms used in continuous state and/or action spaces and is the de facto standard for such problems [Deisenroth *et al.*, 2011; Arulkumaran *et al.*, 2017; Sutton and Barto, 2018; Recht, 2018].

Despite its strong success, we believe that it is possible to substantially improve on REINFORCE and, by extension, on numerous REINFORCE-based systems. As our first step, we here introduce REINFORCE SUN, a superclass of REINFORCE that allows sampling to occur at the level of the synapse, unit, or network and empirically demonstrate that placing sampling in synapses greatly improves performance compared to placing it in units (as in the original REINFORCE) for environments with continuous state and action spaces.

1.2 Significance

REINFORCE is at the heart of numerous ML systems applied to many of today's most urgent research areas including climate, weather, energy, medicine, health care, and robotics. For brevity, we here concentrate only on the latter applications.

To detect pulmonary lesions in chest radiographs, the Recurrent Attention Model with Attention Feedback (RAMAF) is based on the partially observable Markov decision problem (POMDP) framework which allows the model to approximate the gradient using REINFORCE [Pesce *et al.*, 2019]. Similarly, the Reinforced Auto-Zoom Net (RAZN) for accurate and fast prediction of breast cancer segmentation learns a policy for deciding whether zooming is required in a given region of interest using REINFORCE [Dong *et al.*, 2018]. The method outperforms both single and multi scale baseline approaches. REINFORCE is likewise used to improve the policy for a model-based RL approach for treatment of sepsis (a life-threatening complication due to infection) [Raghu *et al.*, 2018].

For generating text for electronic medical records, the Medical Text Generative Adversarial Network (mtGAN), uses REINFORCE to train the GAN framework and optimize the model [Guan *et al.*, 2018]. In mtGAN, the generator tries to maximize the rewards it receives from the discriminator while the discriminator tries to distinguish real text from synthetic text. The model takes disease features as inputs and generates synthetic texts for the corresponding diseases. Similarly, the Hybrid Retrieval-Generation Reinforced (HRGR) agent for medical image report generation has sentence/word generation and template retrieval modules with policies that are updated via REINFORCE [Li *et al.*, 2018]. HRGR achieves state-of-the-art results on two medical report databases.

Robotics applications frequently have continuous states and/or action spaces, and REINFORCE has long been the go-to approach for RL in robotics and is now widely seen as the baseline for creating and comparing numerous derivatives and newer alternatives [Deisenroth *et al.*, 2011; Kober *et al.*, 2013; Peters *et al.*, 2016].

2 Hypothesis

We believe that it is possible to improve upon REINFORCE in several ways. In the present series of papers, we consider the placement of stochasticity within connectionist RL networks. In REINFORCE, stochastic sampling happens at the level of the unit. However, it is possible to move this sampling to the level of the synapse [Ma and Likharev, 2007; Shah and Hougen, 2017] or to the level of the network. To capture these ideas, we propose a superclass of REINFORCE we call *REINFORCE SUN* to reflect that stochasticity is possible at the level of the synapse, unit, or network.

Placing sampling at the level of the synapse increases the number of parameters in play in the system, whereas placing sampling at the level of the network reduces the number of parameters, either of which could be beneficial. More parameters provide finer-grained stochasticity which could lead to higher performance of the system if the additional parameters can be tuned appropriately. Conversely, fewer parameters could lead to faster learning and/or better long-term performance due to the simplified search space. While both outcomes are possible, we anticipate that the former effect will predominate, which leads to the following hypothesis:

H$_1$ *Stochastic synapses will outperform stochastic units which will, in turn, outperform stochastic networks.*

3 Approach

For REINFORCE in continuous action spaces, multiparameter distributions are typically used. For example, Gaussian units can learn both the mean μ and standard deviation σ of their sampling distributions [Williams, 1992]. This allows them to learn both appropriate output values (based on μ) and the degree to which they should be exploratory (based on σ).

However, in a connectionist system there is no reason for being equally exploratory with respect to each input to a given unit. By moving from Gaussian units to Gaussian synapses, those synapses that need more or less adjustment can learn their own sampling distributions (most notably, their own σ values). Conversely, given that the outputs from all units in the output layer contribute to the total reinforcement earned by the RL agent, we could consider a Gaussian network with a single sampling distribution (notably, just one σ value).

To put these concepts in a familiar ANN structure, we'll consider all μ values to be uniformly 0, so that the value at each synapse is determined by its weight w plus stochastic noise ϵ times the input value x on that connection. This presents a choice of *REINFORCE S*, *REINFORCE U*, or *REINFORCE N* determined by whether there is one σ and one corresponding noise sample ϵ on each learning step per synapse, per unit, or per network, respectively.

3.1 Neural Network Architecture

Here we consider only standard feed-forward, fully connected networks with summation units and logistic sigmoidal activation functions [Engelbrecht, 2007]. For simple linear relationships between inputs and outputs, it is sufficient to directly connect input units to output units; however, for more complex relationships to be learned, at least one intermediate layer of units with non-linear activation functions must be added to the ANN [Rumelhart *et al.*, 1986]. Here we consider networks both with and without a hidden layer. In either case, only the output layer of the network is trained using one of the REINFORCE SUN alternatives described herein; the previous layer of the network, if any, is training by backpropagating weight adjustments from the output layer through the hidden units.[1]

3.2 Stochastic Synapses

With stochastic synapses, noise is sampled from a distribution associated with each connection between the hidden layer and the output layer, using

$$\epsilon_{kj} \sim \Psi(\mu_{kj}, \sigma_{kj}) \tag{1}$$

where k and j represent hidden and output layer units, respectively, ϵ_{kj} is the noise sampled from the normal distribution at synapse kj, μ_{kj} is the mean of the noise distribution and σ_{kj} is the standard deviation at synapse kj.

The interior value net_j of each unit j in the output layer is computed using

$$net_j = \frac{\sum_{k=1}^{K} y_k(w_{kj} + \epsilon_{kj})}{K}, \tag{2}$$

where K is the total number of inputs from the hidden layer and w_{kj} is the synaptic weight between units k and j.

The output o_j of each unit in the output layer, then, is computed using the logistic activation function

$$o_j = \frac{1}{1 + e^{-\lambda net_j}}. \tag{3}$$

[1]While we here only consider a single hidden layer and only consider backpropagating weight updates, REINFORCE SUN (like REINFORCE) is compatible with networks of arbitrary depths and any form of gradient descent to update those layers.

Weight updates for the synapses between the hidden and the output layers are computed using

$$\delta_{w_{kj}}(\tau) = (r(\tau) - \hat{r}(\tau))\epsilon_{kj}(\tau) \tag{4}$$

and

$$w_{kj}(\tau + 1) = w_{kj}(\tau) + \eta_w \delta_{w_{kj}}(\tau), \tag{5}$$

where $w_{kj}(\tau)$ is the synaptic weight between units k and j at trial τ, $\delta_{w_{kj}}(\tau)$ is the change in weight on synapse kj computed at the end of trial τ, η_w is the learning rate for synaptic weights between the hidden and output layers, $r(\tau)$ is the reward/penalty collected at the end of trial τ, $\hat{r}(\tau)$ is the expected reward calculated using

$$\hat{r}(\tau + 1) = d * r(\tau) + (1 - d) * \hat{r}(\tau), \tag{6}$$

where $\hat{r}(\tau+1)$ is the expected reward to be computed for the next trial, $\hat{r}(\tau)$ is the expected reward at the current trial, $r(\tau)$ is the reward collected at the end of the current trial, and d is the discount factor.

For stochastic synapses, σ is updated using

$$\sigma_{kj}(\tau + 1) = \sigma_{kj}(\tau) + \eta_\sigma(r(\tau) - \hat{r}(\tau))\frac{\epsilon_{kj}(\tau)^2 - \sigma_{kj}(\tau)^2}{\sigma_{kj}(\tau)}, \tag{7}$$

subject to $\sigma \geq 0$. That is, if Equation 7 would result in a negative value for σ, σ is set euqal to 0.

3.3 Stochastic Units

Stochastic units function very similarly to stochastic synapses, except for the number of noise samples taken per trial and the number of distributions from which those samples are taken. In the case of stochastic synapses these values are each one per synapse. In the case of stochastic units, these values are each one per unit. Otherwise, all elements are the same.

In more detail, with stochastic units, noise is sampled from a distribution associated with each unit in the output layer using

$$\epsilon_j \sim \Psi(\mu_j, \sigma_j) \tag{8}$$

where ϵ_j is the noise sampled from the normal distribution at unit j, μ_j is the mean of the noise distribution, and σ_j is the standard deviation at unit j.

The interior value of each unit j in the output layer is then computed using

$$net_j = \frac{\sum_{k=1}^{K} y_k(w_{kj} + \epsilon_j)}{K}. \tag{9}$$

Weight updates for the synapses between the hidden and the output layers are computed using

$$\delta_{w_{kj}}(\tau) = (r(\tau) - \hat{r}(\tau))\epsilon_j(\tau) \tag{10}$$

where ϵ_j is the noise at unit j on trial τ.

Finally, σ is updated using

$$\sigma_j(\tau + 1) = \sigma_j(\tau) + \eta_\sigma(r(\tau) - \hat{r}(\tau))\frac{\epsilon_j(\tau)^2 - \sigma_j(\tau)^2}{\sigma_j(\tau)} \tag{11}$$

subject to $\sigma \geq 0$.

3.4 Stochastic Network

Likewise, stochastic networks function very similarly to stochastic synapses and stochastic units, except for the number of noise samples taken per trial and the number of distributions from which those samples are taken. In the case of stochastic networks these values are each one per network.

In more detail, with stochastic networks, noise is sampled from a distribution associated with the entire network using

$$\epsilon_N \sim \Psi(\mu_N, \sigma_N) \tag{12}$$

where ϵ_N is the noise sampled from the normal distribution for the entire network, μ_N is the mean of the noise distribution, and σ_N is the standard deviation for network N.

The interior value of each unit j in the output layer is then computed using

$$net_j = \frac{\sum_{k=1}^{K} y_k(w_{kj} + \epsilon_N)}{K}. \tag{13}$$

Weight update for the synapses between the hidden and the output layers is computed using

$$\delta_{w_{kj}}(\tau) = (r(\tau) - \hat{r}(\tau))\epsilon_N(\tau) \tag{14}$$

where ϵ_N is the noise at the network level on trial τ.

Finally, σ is updated using

$$\sigma_N(\tau + 1) = \sigma_N(\tau) + \eta_\sigma(r(\tau) - \hat{r}(\tau))\frac{(\epsilon_N(\tau))^2 - (\sigma_N(\tau))^2}{\sigma_N(\tau)}, \tag{15}$$

subject to $\sigma \geq 0$.

4 Experiments

We performed experiments using a variety of multidimensional data sets including a simple state-action synthetic data model and multiple non-linear data sets based on forward and inverse kinematics of a simplified model of a PUMA robotic arm with all values for all data sets scaled to be in $[0,\ 1]$. In each forward kinematics data set, each input-output vector pair is generated by randomly sampling input values from a uniform distribution and calculating target outputs using underlying continuous functions. In each inverse kinematics data set, input-output vector pairs are generated as with the forward kinematics sets, then swapped (input becoming output and vice versa). The robot arm data sets ensure that each input-output vector pair is almost certainly unique, and each pair is presented only once to the learning agent so it must be able to generalize in order to improve its performance over time. 50 such data sets are used for each experimental condition to collect statistically meaningful results.

Note that this data would be suitable for supervised learning if each input-out pair were presented to the network. However, to maintain the RL nature of the experiments, for each trial (learning experience) τ the network is presented only with the input vector and a scalar reward value $r(\tau)$ that is based on the similarity between the network-generated output response vector and the target output vector from the training data on that trial, calculated as one minus the sum of the absolute difference between the target and the output values divided by the number of output units, that is

$$r(\tau) = 1 - \frac{\sum_{j=1}^{J} |t_j - o_j|}{J}, \tag{16}$$

where J is the number of output units, t_j is the target output at unit j, and o_j is the network's output at unit j. Given the range of possible input and target values and the logistic activation functions of the output units, this gives reward values in [0, 1].

4.1 Problem Sets

As a simple starting problem, the first experiment used 3×3 synthetic data sets. For each data set, a network of random weights uniformly sampled from $[0,\ 1]$ was created and three continuous inputs were generated with equal probability. These, together with a bias value of $+1$ were passed to the network and target outputs were calculated by forward propagation. Repeating the random sampling and forward propagation process 8000 times produced each data set of 8000 trials.

After initial validation with this simple problem, we generated forward and inverse kinematic data sets using a simplified model of a PUMA robot arm. The forward kinematics 3×3 problem here is to predict the position of the end of the arm in Cartesian three-space (x, y, z) given the angles of the revolute waist, shoulder, and elbow joints, and vice versa for the inverse kinematics problem. The forward kinematics 6×6 problem is to predict the position in Cartesian three-space (x, y, z) and orientation as roll, pitch, and yaw of the end effector base plate given the three revolute angles of the waist, shoulder, and elbow joints and three additional revolute angles for a spherical wrist, a total of six angles, and vice versa for the inverse kinematics problem. The 3×6 forward and 6×3 inverse data sets are a subset of 6×6 data set considering only the waist, shoulder, and elbow joints or the Cartesian coordinates, as appropriate. Each data set's input-output pairs are generated using the forward kinematics model of the arm then each angle (for forward) or position component (for inverse) value is used as an input and vice versa.

4.2 Network Setup

Each network has a bias input of $+1$ in addition to the inputs of each data set. For the hidden layer we use hidden units equal to the number of the inputs plus one. The initial weights for each synapse are in $[-2, +2]$, $\lambda = 2$ for the logistic functions, all learning rates and reward discount factors are set to 0.5 (a moderate value), and σ is initialized uniformly randomly in $[0,\ 1]$ and is kept non-negative as suggested for REINFORCE [Williams, 1992].

5 Results and Discussion

Figure 1 shows average reward collected for agents with stochastic synapses, units, and networks at each trial across all kinematics runs. Figures 1a, 1b, and 1c show results for forward kinematics, while Figures 1d, 1e, and 1f show results for inverse kinematics. All inputs and outputs are continuous. Working across this figure from left to right, the first column (Figures 1a and 1d) shows results for 3 inputs and 3 outputs (3×3), the second column (Figures 1b and 1e) shows results for 3 inputs and 6 outputs (3×6) and the opposite with the swap (6×3), and the third column (Figures 1c and 1f) shows results for 6 inputs and 6 outputs (6×6). The apparent differences between these curves are statistically significant (randomized ANOVA, $p < 0.001$ for both algorithm and interaction for all experiments and all pairwise comparisons [Piater *et al.*, 1998]).

Figure 2 shows box and whisker plots of reward collected for agents with stochastic synapses, units, and networks across all kinematics runs. The order of these subfigures is the same as for Figure 1. Here, within each individual sub-figure, the three boxes on the left side represent the cumulative reward collected during each run while the three on the right side represent the cumulative reward in the last 5% of each run when the algorithms have had time to converge.

The apparent differences between these results are statistically significant (Quade test, $p < 0.0075$ for both pre and post-hoc statistics for all experiments and all comparisons, $d.f._N = 2$, $d.f._D = 98$ [Quade, 1979]).

Table 1 shows the average of the average reward collected for all algorithms on all data sets as well as the average of the standard deviation of the reward collected on all data sets for both the entire runs and the last 5% of each run.

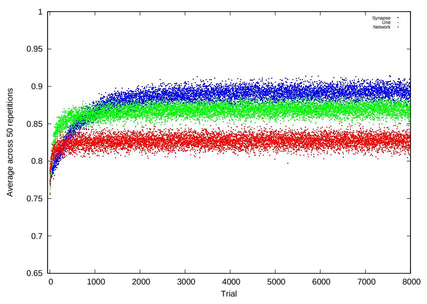
Results for the simple 3×3 problem are not shown graphically due to space limitations but show similar trends.

The results in aggregate clearly show that stochastic synapses substantially and significantly outperform both stochastic units and stochastic networks, at least in the long term (after approximately 1000 trials for all of the kinematics data sets; after only a few hundred trials in the simpler synthetic data sets). It is also worth noting that stochastic synapses show the least variability in the average cumulative reward collected, making this version of the algorithm by far the most consistent in performance. On the other hand, it is also worth noting that stochastic units appear to learn fastest initially on many of the data sets.

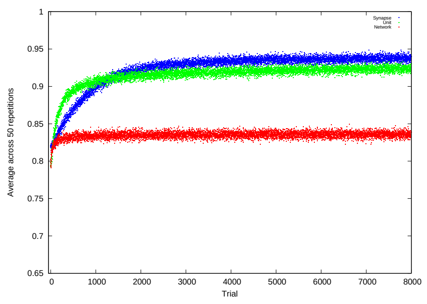
6 Conclusions

This empirical study considers REINFORCE SUN, a superclass that generalizes REINFORCE to allow for stochasticity at the level of the synapse, the unit, or the network. This study contributes to the class of connectionist RL methods that form the basis of several state-of-the-art RL approaches.

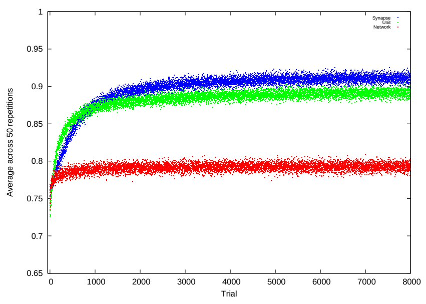
This study supports the general hypothesis that finer-grained stochasticity outperforms coarser-grained stochasticity, at least for a range of interesting continuous-continuous problems, as well as the more specific hypothesis that stochastic synapses outperform both stochastic units and networks, at least in the long term. As one would expect, however, stochastic synapses have to explore at a more granular level and thus take longer to learn initially due to tuning substantially more parameters compared to stochastic units and networks. This suggests that a hybrid approach might combine the best of all variants; that is, using stochastic units and/or networks initially might give faster early learning, and



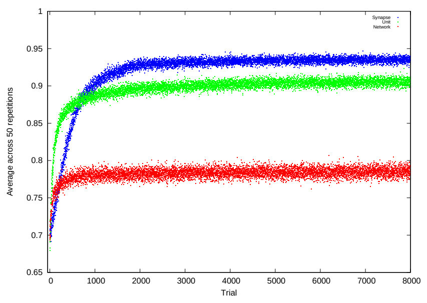
(a) 3×3 Forward Kinematics Data set.



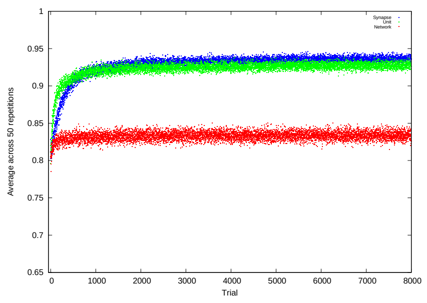
(b) 3×6 Forward Kinematics Data set.



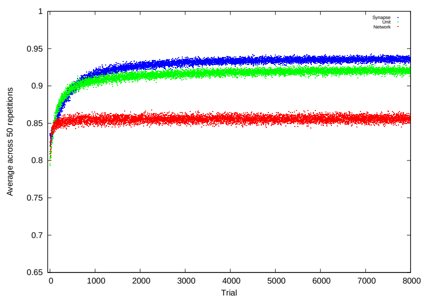
(c) 6×6 Forward Kinematics Data set.



(d) 3×3 Inverse Kinematics Data set.

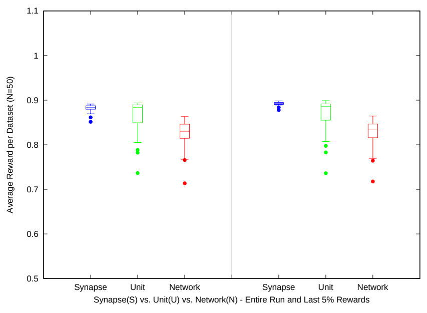


(e) 6×3 Inverse Kinematics Data set.

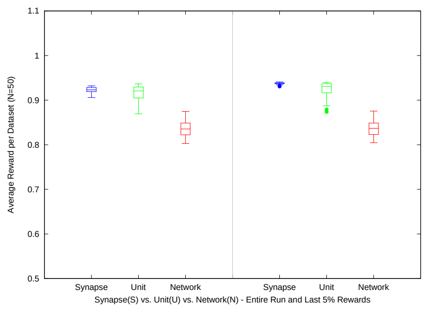


(f) 6×6 Inverse Kinematics Data set.

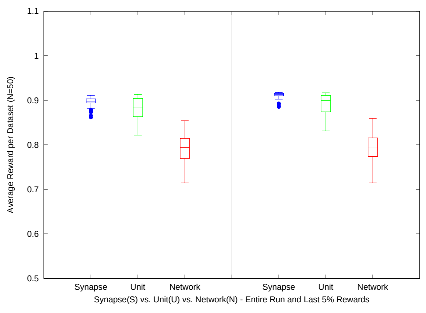
Figure 1: SUN average performance curves for kinematics data sets. Average reward collected per trial across 50 runs by Synapse, Unit, and Network algorithms.



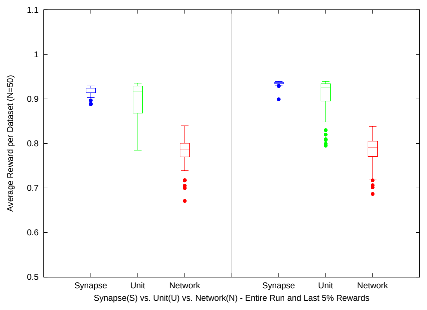
(a) 3×3 Forward Kinematics Data set.



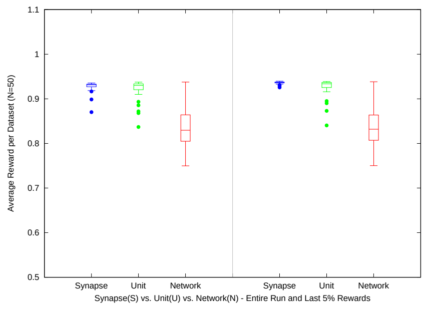
(b) 3×6 Forward Kinematics Data set.



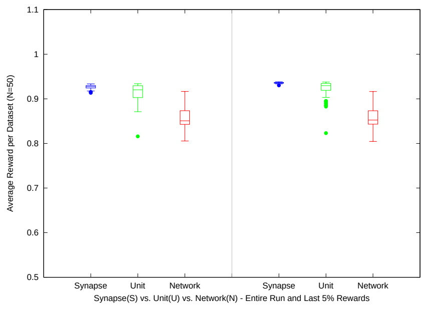
(c) 6×6 Forward Kinematics Data set.



(d) 3×3 Inverse Kinematics Data set.



(e) 6×3 Inverse Kinematics Data set.



(f) 6×6 Inverse Kinematics Data set.

Figure 2: SUN average cumulative reward performance for kinematics data sets. Box and whisker plots with outliers of average cumulative reward per run across 50 runs by Synapse, Unit, and Network algorithms. Left half in each plot shows average cumulative reward received over entire run while the right half shows same for the last 5% of trials in each run.

Data set	Entire run						Last 400 trials					
	Synapse		Unit		Network		Synapse		Unit		Network	
	$\overline{\overline{r}}$	$\overline{\sigma_r}$	$\overline{\overline{r}}$	$\overline{\sigma_r}$	$\overline{\overline{r}}$	$\overline{\sigma_r}$	$\overline{\overline{r}}$	$\overline{\sigma_r}$	$\overline{\overline{r}}$	$\overline{\sigma_r}$	$\overline{\overline{r}}$	$\overline{\sigma_r}$
3×3 SYN	*92.2 %*	*2.8 %*	91.0 %	3.7 %	82.0 %	6.9 %	*92.4 %*	*2.9 %*	91.5 %	3.7 %	82.0 %	7.0 %
3×3 FWD	*88.2 %*	*0.7 %*	86.6 %	3.6 %	82.6 %	2.9 %	*89.2 %*	*0.4 %*	86.9 %	3.6 %	82.7 %	2.9 %
3×3 INV	*91.9 %*	*0.9 %*	89.6 %	4.3 %	78.2 %	3.5 %	*93.5 %*	*0.6 %*	90.5 %	4.3 %	78.5 %	3.5 %
3×6 FWD	*92.3 %*	*0.6 %*	91.5 %	1.8 %	83.5 %	1.8 %	*93.8 %*	*0.2 %*	92.4 %	1.7 %	83.6 %	1.8 %
6×3 INV	*92.8 %*	*1.0 %*	92.3 %	2.0 %	83.3 %	4.2 %	*93.6 %*	*0.3 %*	92.8 %	1.8 %	83.4 %	4.2 %
6×6 FWD	*89.6 %*	*1.1 %*	88.1 %	2.4 %	79.1 %	3.3 %	*91.1 %*	*0.7 %*	89.1 %	2.2 %	79.3 %	3.3 %
6×6 INV	*92.6 %*	*0.5 %*	91.4 %	2.2 %	85.5 %	2.3 %	*93.6 %*	*0.1 %*	92.0 %	2.2 %	85.6 %	2.4 %

Table 1: Averages of average rewards ($\overline{\overline{r}}$) and averages of standard deviations of rewards ($\overline{\sigma_r}$) across all runs. Table depicts averages for both the entire run (left) and last 5% of each run (right). Numbers in italics highlight the highest average reward and lowest standard deviation for each problem. SYN: Synthetic data, FWD: Forward Kinematics, INV: Inverse Kinematics

switching to stochastic synapses later might achieve optimal performance in a long run.

Despite the strength of these results, there is likely room for improvement as suggested by the fact that performance of stochastic synapse never goes above 90% to 95% success, depending on problem. While it may be tempting to attribute this to the fact that data pairs are not repeated, we suspect that it is inherent in the REINFORCE framework. This leaves the need for a better algorithm that can potentially outperform both traditional REINFORCE with stochastic units and REINFORCE SUN with stochastic synapses. Nonetheless, this study contributes significantly to the REINFORCE framework and thereby to state-of-the-art stochastic policy gradient algorithms based on it.

This work also makes available a superior baseline for comparisons. For example, Besson, et al. use a model-based RL approach for a rare disease diagnostic task where they train the baseline using REINFORCE for comparison to a deep Q-network Monte Carlo (DQN-MC) algorithm [Besson *et al.*, 2018]. The choice of training the policy using REINFORCE is influenced by the small number of parameters to learn. They further suggest that they execute a DQN-MC algorithm on the subtasks because they expected DQN-MC to find a better solution than REINFORCE as the DQN has many more parameters and therefore could handle many more different situations than their baseline, i.e, REINFORCE.

7 Future Work

After this research there are several future studies that naturally follow.

First, while this study shows REINFORCE S outperforms the other variants in static environments with continuous states and actions, other environments such as those with discrete states and continuous actions, and/or dynamic environments should also be considered.

Next, as already suggested, a hybrid approach with better long-term performance (the strength of stochastic synapses) yet faster initial learning (the strength of both stochastic units and networks), potentially can achieve even better performance overall and should be studied.

Further, deep reinforcement learning studies with very high dimensional state and action spaces using stochastic synapses can potentially improve on existing results. Similarly, a future study on comparing deterministic policy gradient algorithms to REINFORCE SUN's stochastic synapse method can shed more light on whether our proposed method performs better in spite of stochastic policy gradient algorithms requiring more samples as claimed [Silver *et al.*, 2014].

Additionally, the tasks used in the current study are based on immediate reward; however, a future study using delayed and extended delayed reward scenarios is a logical next step in exploring the vast range of RL tasks as they bring in more complications. Stochastic Synapse Reinforcement Learning (SSRL) [Shah and Hougen, 2017] has shown promise in the domain of continuous actions with delayed reward, unknown reward structures, and perceptual aliasing.

Another future study can be done on comparing stochastic synapses to both off-policy and model-based learning algorithms. Further, the evolution of learning can help find better algorithm dynamics from a population of stochastic synapse variants.

More mundanely, in the current study each sigmoidal activation function uses a λ of 2 which extends the range of unit's output values; however, experimenting with different λ values should be considered, along with other parameter sensitivity tests.

Finally, an interesting question that arises following this study: Is there a correlation that exists among synapse, unit, and network stochasticity where the increasing number of stochastic synapses may or may not affect its performance compared to the other two? Furthermore, since we know after this empirical study that stochastic synapses take longer to learn initially, does it mean there exists a relationship between the number of sigmas and the number of trials needed to outperform the unit and the network algorithms, especially in difficult environments.

References

[Arulkumaran *et al.*, 2017] Kai Arulkumaran, Marc Deisenroth, Miles Brundage, and Anil Anthony Bharath. A brief survey of deep reinforcement learning. *IEEE Signal Processing Magazine*, 34, 08 2017.

[Besson *et al.*, 2018] Rémi Besson, Erwan Le Pennec, St'ephanie Allassonniere, Julien J Stirnemann, Emmanuel Spaggiari, and Antoine Neuraz. A model-based reinforcement learning approach for a rare disease diagnostic task. *CoRR*, abs/1811.10112, 2018.

[Deisenroth *et al.*, 2011] Marc Peter Deisenroth, Gerhard Neumann, and Jan Peters. A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2(1-2):1–142, 2011.

[Dong *et al.*, 2018] Nanqing Dong, Michael Kampffmeyer, Xiaodan Liang, Zeya Wang, Wei Dai, and Eric Xing. Reinforced Auto-Zoom Net: Towards Accurate and Fast Breast Cancer Segmentation in Whole-Slide Images. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 317–325, Cham, 2018. Springer International Publishing.

[Engelbrecht, 2007] Andries P. Engelbrecht. *Computational Intelligence: An Introduction*. Wiley, second edition, 2007.

[Guan *et al.*, 2018] Jiaqi Guan, Runzhe Li, Sheng Yu, and Xuegong Zhang. Generation of Synthetic Electronic Medical Record Text. *arXiv:1812.02793 [cs]*, December 2018. arXiv: 1812.02793.

[Kober *et al.*, 2013] J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, September 2013.

[Li *et al.*, 2018] Christy Y. Li, Xiaodan Liang, Zhiting Hu, and Eric P. Xing. Hybrid retrieval-generation reinforced agent for medical image report generation. In *NeurIPS*, 2018.

[Ma and Likharev, 2007] X. Ma and K. K. Likharev. Global reinforcement learning in neural networks. *IEEE Transactions on Neural Networks*, 18(2):573–577, March 2007.

[Michie and Chambers, 1968] Donald Michie and R. A. Chambers. BOXES: An experiment in adaptive control. In E. Dale and Donald Michie, editors, *Machine Intelligence*. Oliver and Boyd, Edinburgh, 1968.

[Pesce *et al.*, 2019] Emanuele Pesce, Samuel Joseph Withey, Petros-Pavlos Ypsilantis, Robert Bakewell, Vicky Goh, and Giovanni Montana. Learning to detect chest radiographs containing pulmonary lesions using visual attention networks. *Medical Image Analysis*, 53:26 – 38, 2019.

[Peters *et al.*, 2016] Jan Peters, Daniel D. Lee, Jens Kober, Duy Nguyen-Tuong, J. Andrew Bagnell, and Stefan Schaal. Robot learning. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, pages 357–398. Springer International Publishing, Cham, 2016.

[Piater *et al.*, 1998] Justus H. Piater, Paul R. Cohen, Xiaoqin Zhang, and Michael Atighetchi. A randomized ANOVA procedure for comparing performance curves. In *Proceedings of the International Conference on Machine Learning*, volume 98, pages 430–438, 1998.

[Quade, 1979] Dana Quade. Using weighted rankings in the analysis of complete blocks with additive block effects. *Journal of the American Statistical Association*, 74(367):680–683, 1979.

[Raghu *et al.*, 2018] Aniruddh Raghu, Matthieu Komorowski, and Sumeetpal Singh. Model-based reinforcement learning for sepsis treatment. *CoRR*, abs/1811.09602, 2018.

[Recht, 2018] Benjamin Recht. A tour of reinforcement learning: The view from continuous control. *arXiv*, 1806.09460v2, June 2018.

[Rumelhart *et al.*, 1986] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986.

[Shah and Hougen, 2017] S. N. H. Shah and D. F. Hougen. Stochastic synapse reinforcement learning (ssrl). In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8, Nov 2017.

[Silver *et al.*, 2014] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages I–387–I–395. JMLR.org, 2014.

[Sutton and Barto, 1981] Richard S. Sutton and Andrew G. Barto. Toward a modern theory of adaptive networks: Expectation and prediction. *Psychological Review*, 88(2):135–170, 1981.

[Sutton and Barto, 2018] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[Sutton, 1988] Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[Watkins and Dayan, 1992] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, May 1992.

[Watkins, 1989] Christopher John Cornish Hellaby Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, London, UK, May 1989.

[Williams, 1992] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, May 1992.