Stochastic Activation Actor Critic Methods*

Wenling Shang[0000−0001−9973−3092], Douwe van der Wal, Herke van Hoof[0000−0002−1583−3692], and Max Welling[0000−0003−1484−2121]

University of Amsterdam-Bosch-Deltalab, NL
{w.shang, h.c.vanhoof, m.welling}@uva.nl

Abstract. Stochastic elements in reinforcement learning (RL) have shown promise to improve exploration and handling of uncertainty, such as the utilization of stochastic weights in NoisyNets and stochastic policies in the maximum entropy RL frameworks. Yet effective and general approaches to include such elements in actor-critic models are still lacking. Inspired by the aforementioned techniques, we propose an effective way to inject randomness into actor-critic models to improve general exploratory behavior and reflect environment uncertainty. Specifically, randomness is added at the level of intermediate activations that feed into both policy and value functions to achieve better correlated and more complex perturbations. The proposed framework also features flexibility and simplicity, which allows straightforward adaptation to a variety of tasks. We test several actor-critic models enhanced with stochastic activations and demonstrate their effectiveness in a wide range of Atari 2600 games, a continuous control problem and a car racing task. Lastly, in a qualitative analysis, we present evidence of the proposed model adapting the noise in the policy and value functions to reflect uncertainty and ambiguity in the environment.

Keywords: Stochastic Neural Networks · Actor Critic Methods · Deep Reinforcement Learning

1 Introduction

Deep reinforcement learning (DRL)—that is, using deep neural networks (DNNs) in reinforcement learning—has allowed tremendous progress in areas from game playing [25] to continuous control [20]. These DNNs generally serve to approximate value functions [38], such as in deep Q-network (DQN) and its variants [25], or to represent policies [38] such as in policy-gradient methods [35]. Another family of Deep RL (DRL) methods is the hybrid actor-critic approach, which employs DNNs to represent value functions as well as policies [24, 43] and has achieved state-of-the-art performances on highly complex RL problems.

Uncertainties play a crucial role in RL, including probabilistic state transitions, noisy reward functions, non-determinisitic action outcomes [11], and exploration of infrequently tested actions. Earlier DRL works addressing uncertainty have

*We thank anonymous reviewers for their feedback and NVIDIA for GPU donations.

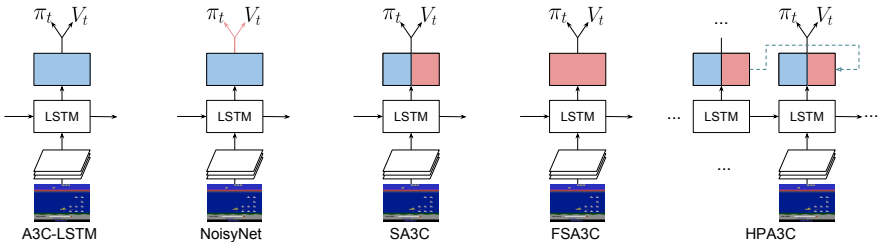

Fig. 1: The baseline **A3C-LSTM** has deterministic latent units and weights only (blue). NoisyNet has stochastic weights (red) over policy and value networks independently. Our proposed methods have stochastic units shared by the two pathway with different configurations:, **SA3C** has half deterministic and half stochastic units in the intermediate layer; **FSA3C** has only stochastic units; **HPA3C** is the same as SA3C but regularized with hierarchical prior from preceding time step during training.

proposed the use of stochastic neural networks (SNNs). SNNs such as Bayesian Neural Networks (BNNs) and NoisyNets [6, 10, 32] improve exploration through injecting parametric noise. Nevertheless, parametric noise has not been equally successful in actor-critic methods ([10], [32]), which are of particular interest because they have performed at a state-of-the-art level in many environments, including Atari games [5] and continuous robotics control [43]. Similar to other model-free approaches, DRL-based actor-critic methods are also highly sensitive with respect to model architecture and other hyperparameter selections, it is therefore important yet non-trivial to discover means to strengthen actor-critic methods with stochastic modeling components.

We propose to directly sample intermediate latent representations shared by both the policy and value network to propagate more complex, structured perturbations, contrasting parametric noise where the weights for the two networks are jittered independently. Particularly, we contribute to the development of a family of stochastic activation A3C models that effectively incorporate stochastic activations on top of LSTM-Asynchronous Advantage Actor Critic [24, A3C-LSTM], a framework representing the current state-of-the-art in many RL tasks.

An important subsequent contribution of this work is a thorough investigation of the empirical performance of stochastic activation A3C on 7 Atari 2600 benchmark games with stochastic skip frames, where our models generally outperform both the SOTA baseline A3C-LSTM and its NoisyNet variant with stochastic weights. Further examination over these experiments demonstrates the decrease of variance over approximated values from multiple samples of stochastic activations during the course of training, indicating a reduction of model uncertainty. Empirical analysis on the converged value and policy networks also show signs of our proposed models reflecting the intrinsic stochasticity of the environment. We then provide a mathematical link between stochastic activations and a special case of stochastic weights yet highlight their essential practical

discrepancies. As an additional contribution, we advance beyond the on-policy A3C-LSTM and incorporate stochastic activations to methods with experience replay and continuous action spaces, namely deep deterministic policy gradients (DDPG) [20] and actor critic with experience replay (ACER) [43]. Pseudocode and full experimental details are in the Appendix; code and video demos are in the Supplementary Materials.

The rest of this paper is organized as follows: first, we discuss related works and preliminaries. Then, we motivate and introduce stochastic activation A3C and our primary model, Stochastic A3C (SA3C), along with two important variants to underline the flexibility of this technique. Next, we present our experimental setup and results, evaluating the overall performance of stochastic units against baselines and stochastic weights. Finally, provide practical advice such as model and hyperparameter selections along with algorithm limitations.

2 Related Work

The treatment of uncertainty has been a long-standing challenge in RL and several lines of research have studied how to address this challenge. Our work is most connected to two general directions, incorporating stochastic components during (1) exploration and (2) inference process.

Epistemic uncertainty, i.e. model uncertainty, reduces as the agents gather more information via exploration. Many exploration mechanisms employ randomized actions instead of always using the best current model (exploitation) to gather more information. These mechanisms include Bayesian methods such as Thompson sampling [13], action-dithering schemes such as ϵ-greedy [38], value randomization such as randomized least-squares value iteration (RLSVI) or with Gaussian Processes [19, 30], et cetera. Many of these mechanisms have also been adapted to the context of DRL, such as Thompson sampling via BNNs [6] and deep value randomization [29].

One approach that was developed in the related field of stochastic optimal control (SOC) uses inference techniques for finding the best actions under uncertain dynamics. In order to do so, the return (or related RL objectives) is defined as a factor in a graphical model and probabilistic inference is applied to determine a sequence of actions optimizing this objective [41]. These probabilistic frameworks have inspired DRL algorithms, such as distributive DQN [4] and deep probabilistic inference for learning control (PILCO) [12]. Recently, DRL models built upon the maximum entropy framework [45] by augmenting the standard RL objective with an entropy maximization term to achieve probabilistic inference have gained much attention, thanks to the potential of improving exploration and generalization in the face of uncertainty [14, 28]. These works also shed light on our proposed framework in retaining a distributive perspective over values and allowing stochastic policies.

The partially-observable setting explicitly addresses uncertainty about the state that the agent is in. A common strategy compresses the unbounded history of observations into belief states, and then subseuqently applies RL to the belief

states [26]. Analytical belief state updates require knowledge of the observation model and transition model – even then is exponential in the number of state variables [26]. DRL-based algorithms that incorporate recurrent modules [24] implicitly maintain analogous internal states. However, these internal states are usually deterministic; in contrast, our model samples its internal states from Gaussian distributions, more similar to belief state approximations in continuous state systems [33].

Our proposed technique fills an important gap in DRL-based actor-critic methods such as A3C-LSTM, where there has been lacking a general yet effective way to include stochastic elements. We apply high-level insights from Bayesian deep learning [17], in particular the use of SNNs, to RL. Applications of SNNs in ML have a long history, [27] [39] to list a few. In the regime of RL, SNNs have also shown promising results. For instance, recently, [10] and [32] concurrently proposed to add independent parametric noises to the FC layers for better exploration, resembling BNNs but without the convergence to a posterior. In contrast, our model perturbs (part of) the intermediate activations which are eventually shared by the actor and critic, allowing structured exploration via better correlated randomness on both paths. Similar SNNs have been employed in several hierarchical RL systems to embed complex skills in an abstract form for higher level tasks [9]. [34] leverages a special case, the variational autoencoder [18], to extract latent representations from raw observations for measuring similarities between states. In these works, the SNNs are separately trained. In our work, we directly alter part of the deterministic units within the baseline models to become stochastic and train the model end-to-end. Finally, recent works propose to measure model uncertainty using DNNs with a special type of stochastic units, dropout units, in the context of e.g. better safety [11, 16].

3 Preliminaries

We consider the standard discrete time step, discounted RL setup. An agent at time t observes o_t, which is a function of its state s_t, and chooses an action a_t guided by a policy π_t. Its ultimate objective is to maximize the accumulative expected return over time $R = \mathbb{E}_{(s_t, o_t, a_t) \sim \pi_t}[r_t]$, where r_t is the reward at time t. This section focuses on introducing the primary baseline algorithm used in our work, batch A3C-LSTM. To demonstrate the generalizability of our proposed method, we perform additional experiments using actor-critic methods with off-policy replays and the descriptions of these models are introduced later in Section 4.2.

Asynchronous advantage actor-critic (A3C) [24] is a model-free, on-policy RL algorithm. Multiple agents are spawned to concurrently interact with the environments with different random seeds and optimize a shared model that approximates both the policy and value functions through asynchronous gradient descent to trade-off bias and variance. A3C models can either be composed of only Convolutional Neural Networks (CNNs) or with an additional recurrent module, usually an LSTM cell. We choose the latter, for it is able to learn more

Game	Human*	A3C†	NN-A3C†	A3C	NN-A3C	Stochastic Act.
Seaquest	28010	1744±0	943±41	13922±4920	894±313	29656±5317
BeamRider	5775	9214±608	11237±1582	9214±608	6117±1808	13779±3605
MsPacman	15693	2436±249	3401±761	4670±1864	4096±1351	5590±1521
Boxing	4.3	91±1	100±0	99.5±1.0	94±4.4	100±0.0
Breakout	31	496±56	347±27	588±180	570±252	621±194
Qbert	13455	18586±574	17896±1522	15333±2462	14352±1335	16045±556
Freeway	29.6	0±0	18±13	23.3±1.2	22.4±0.8	23.9±1.3

Table 1: Results with * and † are cited from [25] and [10]. Due to stochastic frame skipping in our setups which generally yield more difficult environments, our results (last 3 columns) are not precisely comparable to †. Nonetheless we can still clearly conclude the competitiveness of our baseline implementation. The last column presents the results from the optimal stochastic activation models.

complex state representations to tackle e.g. partially observable environments with longer time dependencies. Recently, batch A3C-CNN was developed for faster training and efficient utilization of GPUs [1]. We also take advantage of mini-batch training on A3C-LSTM for better stability and apply synchronous descents [1], where backpropagation waits for all agents to finish their actions so as to avoid stale gradients [8]. Some also refer to similar algorithms as A2C [42].

A3C-LSTM (Figure 1) consists of a CNN to extract features from raw observations, an LSTM cell to compress history, and a value and policy networks. We denote the features extracted by the LSTM as $h_t{=}f_{\text{LSTM}}(\text{CNN}(o_t), h_{t-1})$. In order to be consistent with models introduced later on, we further add two sets of Fully-Connected (FC) layers on top of h_t, obtaining their concatenation $[f_{\text{FC1}}(h_t), f_{\text{FC2}}(h_t)]$ as inputs to the value and policy networks. This structure allows us to later make either or both of these pathways stochastic. The objective for the value network is to estimate the state value V_t by regressing the estimated $t_{\max}$-step discounted returns with discount rate $\gamma \in (0, 1)$ (Equation 1); the policy network proposes a policy π_t and is guided by advantage-based policy gradients using the generalized advantage estimation $\hat{A}$ (details see [36]), regularized by an entropy term to encourage exploration (Equation 2).

$$\text{Value estimation objectvie: } \mathcal{L}_{V_t} = \mathbb{E}_{s_t,o_t,a_t}(\Sigma_{t'=t}^{t_{\max}}\gamma^{t'-t}r_{t'} - V_t)^2, \tag{1}$$

$$\text{Policy gradient with entropy regularization: } \nabla_\theta \log \pi_t(\hat{A}_t) + \beta\nabla H(\pi_t). \tag{2}$$

Finally, we also compare our proposed method with a sthochastic weight variant of A3C-LSTM, NoisyNet A3C (NN-A3C, Figure 1). The construction mostly follows [10] and more details are illustrated in the Appendix.

Our architecture and training protocol produce a state-of-the-art level A3C-LSTM, which is an essential component in our work since we aim at developing a technique that is highly competitive, even surpassing the performance of a very powerful baseline. We compare the baseline A3C implementation replicated by us with another mainstream version as well as human players in Table 1.

4 Actor Critic Methods with Stochastic Activation

This section first illustrates how to integrate stochastic activations into A3C-LSTM, arriving at the stochastic activation A3C family of models. We then describe the primary stochastic activation A3C used in our work, along with two additional variants. Finally, we extend the technique of stochastic units to actor-critic methods with off-policy training (DDPG and ACER).

4.1 Stochastic Activation A3C

Inspired by the SNN design from [39] whose intermediate units are half deterministic and half stochastic in order to encode information of different uncertainty levels, we craft an initial version of stochastic activation A3C in a similar manner, termed stochastic A3C (SA3C). Following the output of the LSTM hidden state h_t, the next layer is split into a deterministic channel and a stochastic channel. The deterministic channel $k_t{=}f_{\text{det}}(h_t)$ is parameterized by a FC layer. The stochastic units follow factored Gaussian distributions. The variance is, for now, set to a fixed value and treated as a hyperparameter, but note that subsequent layers can learn to rely on the deterministic or the stochastic units in any proportion to manage the amount of noise in the value and policy functions. The mean $\mu_t{=}f_{\text{mean}}(h_t)$ is also parameterized by a FC layer. The pseudocode for SA3C is in Algorithm 1. Fully-Stochastic A3C (FSA3C) is an interesting control setup that replaces the deterministic channel with a stochastic one and attains a fully-stochastic intermediate representation. Hierarchical Prior stochastic activation A3C (HPA3C) is inspired by BNNs that craft their priors to the model parameters in order to achieve certain effects, such as inducing sparsity [21]. Analogously, HPA3C adds a KL-divergence between the stochastic activation distribution and a prior to the objective function. Specifically, the prior for the variance is fixed to a value σ^2, treated as a hyperparameter, and the prior for μ_t is derived from the previous step stochastic latent sample.[1] Our design is also similar in spirit to latent forward modeling [40] where the history predicts and guides the future, but in a more implicit form of prior regularization:

$$\text{Derivation of } \mu_t^p : z_{t-1}{\sim}\mathcal{N}(\mu_{t-1}, \sigma^2), \mu_t^p{=}f^p(z_{t-1}),$$

$$\text{Prior regularization: KL}\left[\mathcal{N}(\mu_t, \sigma_t^2) \middle\| \mathcal{N}(\mu_t^p, \sigma^2\right] = \log\frac{\sigma}{\sigma_t} + \frac{\sigma_t^2 + (\mu_t - \mu_t^p)^2}{2(\sigma)^2} - \frac{1}{2}.$$

We found that a proper prior choice is critical — omitting either the prior on the mean or the variance significantly deteriorates the model performance.The pseudocode for HPA3C is in the Appendix.

Forward propagation through stochastic activation A3C is identical to A3C-LSTM, except that the stochastic activations z_t are sampled from $\mathcal{N}(\mu_t, \sigma_t)$ and then concatenated with the deterministic counterpart k_t as the inputs for the policy and value networks. Backpropagation via the stochastic units is done by the reparametrization trick [18].

[1] All operations are element-wise because of the factored Gaussian assumption.

Initialize network parameters θ;
Fix variance σ^2;
for $k = 0, 1, 2, \cdots$ **do**
  Clear gradients $d\theta \leftarrow 0$;
  Simulate under current policy π_{t-1} until $t_{\max}$ steps are obtained, where,
  $h_t = f_{\text{LSTM}}(\text{CNN}(o_t), h_{t-1})$, $\mu_t = f_{\text{mean}}(h_t), k_t = f_{\text{d}}(h_t)$,
  $z_t \sim \mathcal{N}(\mu_t, \sigma_t^2 = \sigma^2)$, $V_t = f_v(z_t, k_t)$, $\pi_t = f_p(z_t, k_t)$, $t = 1, \cdots t_{\max}$;
  $R = \begin{cases} 0, \text{ if terminal} \\ V_{t_{\max}+1}, \text{otherwise} \end{cases}$;
  for $t = t_{\max}, \cdots 1$ **do**
    $R \leftarrow r_t + \gamma R$;
    $A_t \leftarrow R - V_t$;
    Accumulate gradients from value loss: $d\theta \leftarrow d\theta + \lambda \frac{\partial A_t^2}{\partial \theta}$;
    $\delta_t \leftarrow r_t + \gamma V_{t+1} - V_t$;
    $\hat{A}_t \leftarrow \gamma \tau \hat{A}_{t-1} + \delta_i$;
    Accumulate policy gradients with entropy regularization:
    $d\theta \leftarrow d\theta + \nabla \log \pi_t(a_t) \hat{A}_t + \beta \nabla H(\pi_t)$;
  end
end

Algorithm 1: SA3C

Lastly, it is worth noting that while our models employ Gaussian units thanks to their flexibility and ease to train, the proposed framework can adopt other stochastic units as well. We conduct preliminary experiments with dropout stochastic units in the Appendix and leave further investigation along this direction to future works.

4.2 DDPG and ACER

Deep Deterministic Policy Gradients (DDPG) [20] is an off-policy actor critic method. It explores via injecting action space noise, commonly from the Ornstein-Uhlenbeck process. We equip DDPG with parametric noise [32] (PG-DDPG) or stochastic activation (SDDPG). We do not incorporate an LSTM module to DDPG and its variants. The baseline algorithm thus follows exactly as in [20] and its parametric noise version, PN-DDPG, exactly as in [32] but without randomizing the convolutional layers. Unlike A3C-LSTM, DDPG keeps separate encoders for actor and critic. We only use stochastic activations to the behavior actor network and not to off-policy training.

Actor Critic with Experience Replay (ACER) [43] is a sample-efficient actor-critic algorithm with a hybrid of on/off-policy gradients. We compare amongst ACER and its variants with stochastic units or noisy layer. Augmenting ACER with stochastic activation (SACER) follows the same protocol as augmenting A3C-LSTM with stochastic activation and we also use stochastic activations for off-policy training. As an additional comparison, we construct a NoisyNet version of ACER, NN-ACER by similarly randomizing the value and policy networks as

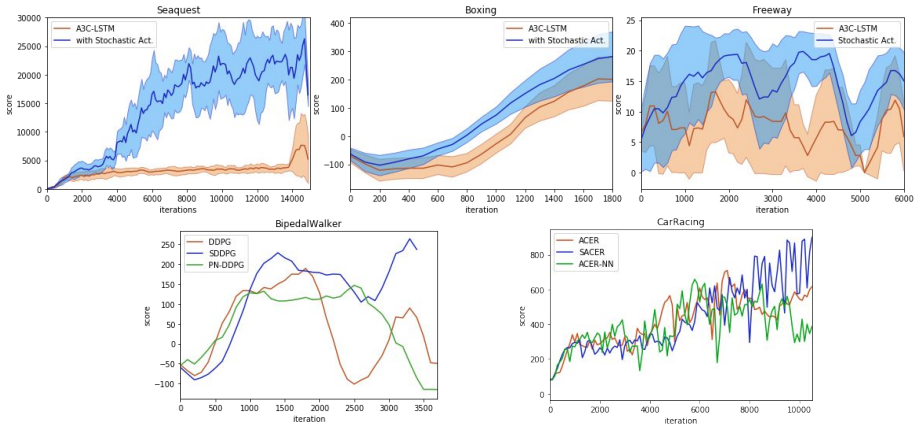

Fig. 2: Training curves over 3 runs (median); vertical: rewards; horizontal: iterations. For Atari games, we plot the curves for the baseline and the best stochastic activation model along with the interquartile range. For the rest, we compare among the baseline, SA3C and then stochastic weights.

in NN-A3C. The pseudocode of our ACER and its stochastic variants are in the Appendix.

5 Experimental Setup and Results

This section first introduces the environments used in our experiments. Extensive ablation studies are done on the Atari games. We then discuss the empirical advantages of stochastic activation A3C over its deterministic baseline and how its design flexibility can adapt well to a variety of environments and tasks. Finally, we present additional results generalizing SA3C to off-policy methods, namely DDPG and ACER, on BipedalWalker2D and CarRacing respectively.

5.1 Environments

Our experiments are primarily done in an on-policy manner on 7 selected classic Atari 2600 games contained in the Arcade Learning Environment [5] and interfaced via OpenAI Gym [7] to cover a diverse range of tasks and exploration types [3]. Full descriptions of these games are in the Appendix. To avoid memorization and impose more randomness, we use the stochastic frame-skipping: each action is repeated for a number, uniformly sampled between 2 and 4, of consecutive frames. Exploration type is categorized by the taxonomy from [3]. The stochasticity of Atari games originates from multiple sources, including frame-skipping, partial observation of some environments, non-stationary policy during training, approximation errors, et cetera. For preprocessing, we crop Atari

Game	A3C-LSTM		SA3C		NoisyNet		Optimal Model		
	Best	Avg.	Best	Avg.	Best	Avg.		Best	Avg.
Seaquest	13922 ±4920	6785 ±5050	**28876** ±4270	23411 ±4783	849 ±313	1332 ±367	HPA3C	**29656** ±5317	24992 ±3356
BeamRider	9214 ±608	8723 ±627	**9994** ±3717	8966 ±1013	6117 ±1808	5838 ±287	FSA3C	**13779** ±3605	10551 ±2341
MsPacman	4670 ±1864	3973 ±543	**4960** ±1639	4743 ±220	4096 ±1351	3705 ±297	FSA3C	**5590** ±1521	5382 ±268
Boxing	**100** ±0.0	99.7 ±0.2	**100** ±0.0	99.9 ±0.0	94 ±4.4	11.6 ±59.6	HPA3C	**100.0** ±0.0	99.6 ±0.23
Breakout	588 ±180	560 ±22	**621** ±194	556 ±45	570 ±252	551 ±25	HPA3C	596 ±197	569 ±22
Qbert	15333 ±2462	14732 ±482	**15560** ±184	15365 ±150	14352 ±1335	11231 ±3348	HPA3C	**16045** ±556	15365 ±150
Freeway	**23.3** ±1.2	22.8 ±0.7	22.4 ±1.1	21.6 ±0.5	22.4 ±0.8	21.5 ±0.6	HPA3C	**23.9** ±1.3	23.2 ±0.5

Table 2: We report Atari results following the evaluation protocol in Sec 5. SA3C outperforms the baselines most of the time. The last column displays the results from the optimal stochastic activation variants for each game which can further boost the testing scores.

games to display only the game playing region, subtract estimated mean and divide standard deviation, and rescale to 80×80.

DDPG models are tested on a continuous task, BipedalWalker2D, where a robot needs to reach the end of a path within a time limit and positive reward is given for moving forward, totaling ≥ 300 for reaching the end, while a negative reward of -100 is given for falling. No preprocessing is done for this environment. ACER models are tested with CarRacing, a simple driving simulator whose observations consist of RGB top-view of a race car and a black bar containing other driving information. We only receive the pixel-valued observations and also discretize its action space. More details on the preprocessing of CarRacing is provided in the Appendix.

5.2 Stochastic Activation A3C Results

Hyperparameters and Model Architecture For Atari, hyperparameters are tuned on Seaquest A3C-LSTM and then transferred to other games. We inherit all common hyperparameters from A3C-LSTM to stochastic activation A3Cs and only tune the additional ones, namely σ^2 for SA3C and FSA3C, HPA3C and the KL term weight for HPA3C. In particular, we would like to emphasize the coefficient for entropy regularization is tuned to perform optimally on the baseline–a higher value in fact deteriorates its performance; in other words, any performance gain via stochastic activations cannot be replaced by increasing the entropy term. For other environments, hyperparameters are tuned on the baseline and then transferred to stochastic weight/activation models.

Fig. 3: For SA3C, stochastic activations can result in stochastic policies. The less ambiguous the environment is, the more certain the policies become (left to right). Arrows indicate the direction of movement, followed by the number of times this action being selected (out of 5 samples).

Most games share a common model architecture but we use a slightly slimmer CNN for Boxing, Breakout and Freeway. Since HPA3C needs to learn σ_t, more variance is introduced to the gradients and the resulting stochastic activations require further normalization. After trial-and-error with several techniques such as Batch [15] and Layer [2] Normalization, we pick the most effective option–concatenated ReLU [37]. Full details are given in the Appendix.

Evaluation Protocol We report Atari results on A3C-LSTM, NN-A3C, SA3C and the best performing stochastic activation A3C variant in Table 2 following the protocol:

1. Train 3 independent runs—a standard DRL practice [31, 44].
2. For each run, validate the current model on a non-training random seed, select the best (validated) one after training.
3. Test the selected model for each run on 10 other random seeds not included in training or validation, obtaining $\mu_1 \pm \sigma_1$, $\mu_2 \pm \sigma_2$, $\mu_3 \pm \sigma_3$.
4. Report the best $\mu_i \pm \sigma_i$ under the column "Best".
5. Average across the 3 models, i.e. $\mu \pm \sigma$ over μ_1, μ_2, μ_3, reported under "Avg.".

The proceeding protocol not only showcases how good a policy the algorithm can attain if optimized well but also indicates variances in performance due to policy gradient training. We also plot the training curves composed of average validation scores with the standard deviation bars for Seaquest, Boxing and Freeway in Figure 2, other games in the Appendix.

Inference and Stochastic Policies Based on the protocol from [10], NoisyNet is tested by setting the stochastic weights equal to the learned mean. For stochastic activation A3Cs, there are multiple possibilities during evaluation time. One is to only use the mean from the stochastic units, referred to as Maximum A Posteriori (MAP)–borrowing the Bayesian terminology.

	DDPG	SDDPG	PN-DDPG		ACER	SACER	NN-ACER
BipedalWalker	5900	**3500**	6300	CarRacing	891	**900**+	859

Table 4: Compare various actor-critic baselines with their stochastic-activation/weight variants, tested on BipedalWalker and CarRacing. We report the median iteration of solving the environmen for BipedalWalker and the median best score in 10K iteration for CarRacing, where 300+ and 900+ are considered solved for each task respectively.

Alternatively, we sample the stochastic activations and vote the majority decision, leading to stochastic policies. Figure 3 shows the decisions out of 5 sampled policies for selected states: when there is no clear immediate goal, e.g. no enemy around, decisions tend to diverge, but otherwise they agree. Videos of Seaquest with deterministic polices from A3C-LSTM versus stochastic policies from SA3C are included in the Supplementary Materials. Table 3 compares different evaluation schemes from Seaquest — for stochastic policies we attempt 1, 5, and 50 samples — and 5 samples give the optimal results for most models. If not mentioned otherwise, all stochastic activation A3C results are obtained by voting among policies from 5 sampled activations.[2]

Seaquest	SA3C	FSA3C	HPA3C
MAP	27695±9096	4387±171	25474±8067
1	27081±6817	**5090**±1099	24475±4765
5	**28876**±4270	4453±592	**29656**±5317
50	28341±9839	4794±523	28341±9839

Table 3: Compare evaluating using μ only (MAP), using a single sampled stochastic activation and averging over 5 or 50 stochastic activation outcomes. Sampling and averaging 5 activations tend to be optimal.

5.3 Actor-Critic Models with Experience Replay

We further integrate this technique to actor-critic methods with experience replay, namely ACER and DDPG. Complete hyperparameter details are in the Appendix. For DDPG, we plot the median training curves out of 3 independent runs in Figure 2. We found that DDPG is much less stable in training comparing with A3C-LSTM. Adding stochasticity to DDPG does not improve its training stability, which remains an open question. Nonetheless, SDDPG tends to converge significantly faster than DDPG and DDPG-PN, at iteration 4000, 5900 and 6300 respectively (Table 4). For ACER, we plot the median training curves out of 3 independent runs in 10K iterations in Figure 2. However, note that we stop the training once the environment is solved, i.e. average validation score over 10 random runs ≥ 900 . Out of the 3 runs (with maximum 10K iterations), only SACER manage to solve the environment. The median best scores attained by ACER is 891 and NN-ACER 859 (Table 4).

[2]We use 1 sample for Freeway. As it only has 2 actions, voting would strongly diminish policy stochasticity.

Stochastic activations boost the performances of the baselines and outperform parametric space noise. These results confirm the effectiveness of our proposed method when coupled with experience replay.

5.4 Further Analyses of Stochastic Activations

We performed further analyses of stochastic activation networks to investigate the mechanisms behind the observed performance increase. We first inspected the value learning process of SA3C. These results are given below. We investigated several other aspects, including the significance of a shared intermediate stochastic learning, the policy learnign process, and a detailed comparison of to stochastic weights. The results of these experiments are given in the appendix.

Motivated by recent works using dropout units to estimate uncertainty [11, 16], we obtain and analyze uncertainty estimations by calculating the sample variance of multiple approximated values over sampled stochastic activations. Concretely, for SA3C models at different training stages, we sample stochastic activations 5 times for each time step, calculate the variances of those resulting values and plot these in Figure 4 for the first 700 time steps, that is

$$\hat{\sigma_v}^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(f_V(k_t,z_t)_i - \bar{f_V}\right)^2, \bar{f_V} = \frac{1}{N}\sum_{i=1}^{N} f_V(k_t,z_t)_i, N{=}5, t{=}1\cdots 700.$$

The variances tend to go down over training (Figure 4). This behavior is reminiscent of (Bayesian) models employing distributions over the weights, where these distributions reflect parametric uncertainty. Indeed, there is a connection between stochastic weights and activations that we will discuss more later. Despite the fixed variance of the noise in the stochastic units, the value network is clearly capable of gradually adapting the variance through learning. This can be achieved by shifting focus from stochastic units to deterministic units in downstream computations.

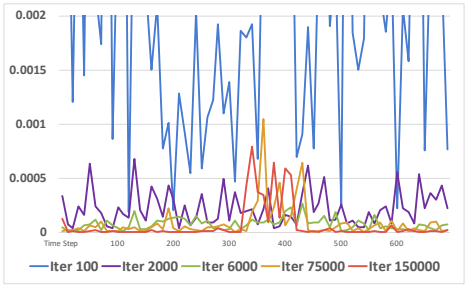

Fig. 4: At different training iterations for SA3C, we sample 5 V_ts via stochastic activations for $t = 1..700$, plot their variances and observe the general trend of variances descends over training.

At convergence time, the value network usually approximates with little variance, reflecting low uncertainty, except for a surge period around step 340 (Figure 5). This in fact corresponds to a special event where different actions can lead to varying amounts of rewards: the submarine has reached maximum capacity of rescued divers and it can either shoot the upcoming enemy to gain some points or surface to collect a large amount of rewards by releasing the divers—eventually the submarine chooses the latter. Ambiguous situation can

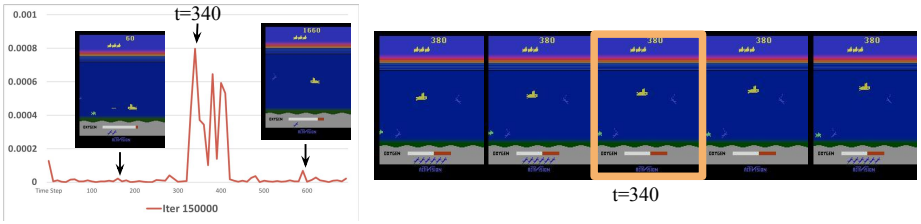

Fig. 5: A zoom-in of the variance plot at convergence (iter 150K). The variance is generally low except for periods with high unpredictability. Around step 340, the submarine has rescued divers and should surface, but could also shoot the enemy. More details see Section 5.4.

increase the variance comparing to those with a clear target, comparing amongst step 340, 600 and 160. Therefore, we argue that SA3C's adaption to stochastic activations is not merely reducing the influence of the stochastic units, but rather carefully balancing between the deterministic and stochastic signals to implicitly establish a distributional perspective over returns and values, a beneficial trait for RL [23] and a proper reflection of the environment unpredictability.

6 Practical Advice and Algorithm Limitations

Stochastic activation is a general approach to improve A3C but not the panacea to every environment and task. Fully observable environments, such as Breakout, benefit less from stochastic activations. Environments with sparse rewards like Freeway also receive a more limited performance boost. RL problems with sparse rewards and/or more complex logic will require more specialized systems in combination with stochastic units, such as curiosity driven exploration.

The flexibility of stochastic activation A3C allows effective application to a diversity of tasks, but model selection can appear labor-demanding at first glance. From our experiences, SA3C is the go-to model as an initial attempt; if more aggressive exploration seems appropriate, FSA3C is a good candidate; if forecasting the upcoming states is essential in solving the task or rewards are sparse, HPA3C will likely perform better and more stable. One can thus always easily customize the stochastic activation A3C to meet the need of the task.

7 Conclusion

We proposed a flexible and simple to implement technique to improve DRL-based actor-critic methods by adding stochastic activations. The proposed method outperforms existing state-of-the-art baselines on a variety of benchmarks. In future work, we hope to integrate the proposed technique with curiosity-driven exploration to address problems with sparse rewards and experiment with other types of stochastic units such as binary units for feature level count-based exploration [31].

Bibliography

[1] Adamski, I., Adamski, R., Grel, T., Jedrych, A., Kaczmarek, K., Michalewski, H.: Distributed deep reinforcement learning: learn how to play Atari games in 21 minutes. arXiv (2016)

[2] Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv (2016)

[3] Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., Munos, R.: Unifying count-based exploration and intrinsic motivation. In: NIPS (2016)

[4] Bellemare, M.G., Dabney, W., Munos, R.: A distributional perspective on reinforcement learning. ICML (2017)

[5] Bellemare, M.G., Naddaf, Y., Veness, J., Bowling, M.: The arcade learning environment: An evaluation platform for general agents. In: IJCAI (2013)

[6] Blundell, C., Cornebise, J., Kavukcuoglu, K., Wierstra, D.: Weight uncertainty in neural networks. ICML (2015)

[7] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: OpenAI Gym (2016)

[8] Chen, J., Monga, R., Bengio, S., Jozefowicz, R.: Revisiting distributed synchronous SGD. arXiv (2016)

[9] Florensa, C., Duan, Y., Abbeel, P.: Stochastic neural networks for hierarchical reinforcement learning. ICLR (2017)

[10] Fortunato, M., Azar, M.G., Piot, B., Menick, J., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., et al.: Noisy networks for exploration. ICLR (2018)

[11] Gal, Y., Ghahramani, Z.: Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In: ICML (2016)

[12] Gal, Y., McAllister, R., Rasmussen, C.E.: Improving PILCO with bayesian neural network dynamics models. In: ICML Workshop (2016)

[13] Ghavamzadeh, M., Mannor, S., Pineau, J., Tamar, A., et al.: Bayesian reinforcement learning: A survey. Foundations and Trends® in Machine Learning (2015)

[14] Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. arXiv preprint arXiv:1801.01290 (2018)

[15] Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML (2015)

[16] Kahn, G., Villaflor, A., Pang, V., Abbeel, P., Levine, S.: Uncertainty-aware reinforcement learning for collision avoidance. In: arxiv (2016)

[17] Kingma, D.P., Salimans, T., Welling, M.: Variational dropout and the local reparameterization trick. In: NIPS (2015)

[18] Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: ICLR (2013)

[19] Kuss, M., Rasmussen, C.E.: Gaussian processes in reinforcement learning. In: NIPS (2004)

[20] Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. ICLR (2016)
[21] Louizos, C., Ullrich, K., Welling, M.: Bayesian compression for deep learning. In: NIPS (2017)
[22] Machado, M.C., Bellemare, M.G., Talvitie, E., Veness, J., Hausknecht, M., Bowling, M.: Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. arXiv (2017)
[23] Barth-Maron, G., Hoffman, M., Budden, D., Dabney, W., Horgan, D., TB, D., Muldal, A., Heess, N., Lillicrap, T.: Distributional policy gradients. In: ICLR (2018)
[24] Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. In: ICML (2016)
[25] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature (2015)
[26] Murphy, K.P.: A survey of POMDP solution techniques. Tech. rep. (2000)
[27] Neal, R.M.: Learning stochastic feedforward networks. Technical Report (1990)
[28] O'Donoghue, B., Munos, R., Kavukcuoglu, K., Mnih, V.: Combining policy gradient and q-learning. arXiv preprint arXiv:1611.01626 (2016)
[29] Osband, I., Russo, D., Wen, Z., Van Roy, B.: Deep exploration via randomized value functions. arXiv (2017)
[30] Osband, I., Van Roy, B., Wen, Z.: Generalization and exploration via randomized value functions. ICML (2016)
[31] Ostrovski, G., Bellemare, M.G., Oord, A.v.d., Munos, R.: Count-based exploration with neural density models. ICML (2017)
[32] Plappert, M., Houthooft, R., Dhariwal, P., Sidor, S., Chen, R.Y., Chen, X., Asfour, T., Abbeel, P., Andrychowicz, M.: Parameter space noise for exploration. ICLR (2018)
[33] Prentice, S., Roy, N.: The belief roadmap: Efficient planning in linear pomdps by factoring the covariance. (2007)
[34] Pritzel, A., Uria, B., Srinivasan, S., Puigdomenech, A., Vinyals, O., Hassabis, D., Wierstra, D., Blundell, C.: Neural episodic control. ICML (2017)
[35] Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P.: Trust region policy optimization. In: ICML (2015)
[36] Schulman, J., Moritz, P., Levine, S., Jordan, M., Abbeel, P.: High-dimensional continuous control using generalized advantage estimation. ICLR (2016)
[37] Shang, W., Sohn, K., Almeida, D., Lee, H.: Understanding and improving convolutional neural networks via concatenated rectified linear units. In: ICML (2016)
[38] Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. artificial intelligence (1998)

[39] Tang, Y., Salakhutdinov, R.R.: Learning stochastic feedforward neural networks. In: NIPS (2013)
[40] Tian, Y., Gong, Q.: Latent forward model for real-time strategy game planning with incomplete information. In: NIPS Symposium (2017)
[41] Todorov, E.: General duality between optimal control and estimation. In: CDC (2008)
[42] Wang, J.X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J.Z., Munos, R., Blundell, C., Kumaran, D., Botvinick, M.: Learning to reinforcement learn. arXiv (2016)
[43] Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., de Freitas, N.: Sample efficient actor-critic with experience replay. ICLR (2016)
[44] Wu, Y., Mansimov, E., Grosse, R.B., Liao, S., Ba, J.: Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In: NIPS (2017)
[45] Ziebart, B.D., Maas, A.L., Bagnell, J.A., Dey, A.K.: Maximum entropy inverse reinforcement learning. (2008)