SPIKING NEURAL NETWORKS FOR CONTINUOUS CONTROL: NEUROMORPHIC REINFORCEMENT LEARNING IN CONVENTIONAL COMPUTING

Jessica Hunter
Department of Computer Science
New Mexico Institute of Mining and Technology
Socorro, NM 87801, USA
jessica.hunter@student.nmt.edu

Krishna Roy
Department of Electrical Engineering
New Mexico Institute of Mining and Technology
Socorro, NM 87801, USA
krisna.roy@nmt.edu

Md Maruf Hossain Shuvo
Department of Electrical and Computer Engineering
The University of Texas at El Paso
El Paso, TX 79968, USA
mhshuvo@utep.edu

ABSTRACT

Reinforcement learning (RL) algorithms have made strides over the past decade applying them to a wide range of problems and control tasks. However, the deployment of RL on neuromorphic hardware for continuous control tasks remains under-validated. Namely it is unclear whether replacing a conventional actor network with a spiking neural network (SNN) affects the performance of an agent before any hardware-specific benefits manifest. We provide a systematic validation of a minimal, neuromorphically viable spiking actor variant of Soft Actor-Critic (SAC) on conventional hardware, establishing a baseline for future neuromorphic RL research. In this paper, we propose the Spiking Actor Network Soft Actor Critic (SANSAC) to address the use of RL frameworks in continuous environments, designed as a framework that can be implemented on neuromorphic hardware. We compare a traditional Soft Actor Critic (SAC) network to SANSAC in a traditional computer. We demonstrate the near equivalent performance of SANSAC and SAC, while addressing the impact of hidden dimensions. Our results demonstrate the viability of SNN based algorithms in complex continuous environments, as well as competitive performance to traditional neural networks in traditional computers, providing a basis to continue exploring the use of SNNs in continuous RL frameworks. [1]

1 INTRODUCTION

Reinforcement learning (RL) as a field has drawn attention in recent years as breakthroughs continue Mu et al. (2021); Hafner et al. (2019). Despite recent advancements made to the field, solutions for continuous control problems have stalled in respect to off-policy algorithms. The use of neuromorphic hardware including memristors makes this more evident. Neuromorphic hardware poses a problem for the implementation of traditional RL frameworks in continuous control problems. As continuous environments exacerbate the issue of reproducibility, stability, and portability found in RL frameworks Faria et al. (2022). Progress has been slow due to the high complexity of tasks such as those seen in robotics consisting of Partially Observable Markov Decision Processes (POMDP). These tasks lead to an increase in resources needed to train and run an agent for a given task which creates a larger barrier to entry for solutions. Neuromorphic hardware attempts to resolve parts of these issues by decreasing energy cost, training time, and total hardware required for in-situ training on systems that traditionally use embedded hardware Tang et al. (2020a). This is a direct result of the

[1] Implementation code is publicly available at https://github.com/jhunter533/SANSAC.

truly parallel nature of these chips in comparison to traditional traditional computer architectures. However, current research is non extensive for solutions that are high performing on continuous control tasks implemented on neuromorphic hardware. Improvements in this area have been made as the prioritization of spiking neural networks (SNNs) increases, taking cues from biological process to steer cutting edge techniques which advance the ability of problem solving for continuous control problems.

Spiking neural networks, considered the third generation of artificial intelligence Maass (1996), have made large contributions to the implementation of neural networks on neuromorphic hardware. Differing degrees of biological plausibility have given rise to several neuron model types used by SNNs including Leaky-Integrate-Fire (LIF) Abbott (1999), Hodgkin-Huxley Hodgkin & Huxley (1952), Izhikevich Izhikevich (2003), and many more. In addition there exist several learning methods for networks including three-factor learning rules Frémaux & Gerstner (2016), derived from biological cues, and surrogate gradients Neftci et al. (2019), derived from traditional gradient descent. The variety in techniques have allowed for expansion of the use of SNNs in neural network applications. With the acquisition of neuromorphic hardware proving difficult for most at present, it stands to reason that there is a clear gap in comparison of SNN and traditional neural networks on traditional computers. Understandably, this is a result of SNN benefits only being significant in neuromorphic hardware, however, there remains little in terms of RL to broadly assume this to be true for all cases. While it can reasonably assumed that SNNs would not perform significantly better than digital neural networks (DNNs) in this system, it remains an important confirmation to fully understand for continuous tasks. Through the observation of data comparisons on a traditional computer we can see the small differences SNNs and DNNs have that can cascade on individually advantaged hardware.

Implementation of neural networks on neuromorphic hardware, be it memristors Wang et al. (2019) or the Intel Loihi Davies et al. (2018) require all network components to have a physical representation. Thus, models and designs must be constrained to reasonably available neurons. As this constraint poses a crucial block to rapid progress, we extend our research to compare DNN and SNN implementation in the same traditional computer architecture including observations of reduced neuron counts for all hidden layers to function operably.

We propose the implementation of the Spiking Actor Network Soft Actor Critic (SANSAC), a network designed to be a minimal modifcation of SAC where only the actor is replaced by an SNN. Our contribution is not the novelty of this suggestion but the systematic comparison of SANSAC to SAC on identical conventional hardware across multiple network sizes. We analyze the effect of hidden dimension on policy learning and quantify performance differences. This establishes a reproducible baseline against which future neuromorphic implementations can be compared. We aim to understand the core of SNNs in continuous control tasks by observing possible differences in several performance metrics across both algorithms.

2 Related Work

Important to understanding the basis of our methods is the basics of neuromorphic hardware and alternative techniques that influenced the work.

2.1 Population Encoding

Present work in SNNs introduce the biologically plausible mechanism of population encoding Urbanczik & Senn (2008). Described from neuroscience developments of the human brain's mechanisms, it was then applied to reinforcement learning frameworks Knight (1972). While implementations differ from context to context Sun et al. (2025), in continuous control environments it was popularized by PopSAN Tang et al. (2020b;a) which create a general framework for the addition of population encoding and decoding to any traditional Actor-Critic algorithm. While this method for encoding and decoding closely mimics observed behavior of learning in the brain and improves associated SNN performance, we opt to not include it in our algorithm implementations. This stems from the added complexity in design and implementation as well as the possibility of it obscuring the data in our base algorithm comparison results.

2.2 NEUROMORPHIC HARDWARE

Neuromorphic hardware encapsulates systems that aim to emulate biological processes principles through special hardware which is entirely separate from a classical traditional computer architecture. These systems employ a combination of analog, digital, and mixed-signal circuits to implement neurons and synapses. Unlike the traditional computer, these systems are driven by spike events. There has been growing interest in a areas that require limited resources Wang et al. (2022); Zhang et al. (2020) to use this hardware. Whereas traditional computers are defined by their separate CPU and memory units creating bottlenecks for data transmission, neuromorphic machines use neurons and synapses allowing mass parallelization. Often done through the implementation of memristors or semiconductors Schuman (2022). It is currently of growing interest to implement complex algorithms such as RL frameworks on these systems Wu et al. (2020) primarily in an effort to gain nanoscale size and lower power consumption Shi et al. (2021); Wang et al. (2019). The use of such hardware in complex problems including robotics and other embedded hardware systems can increase power and space usage while maintaining similar performance. Highly desired for future implementation of highly efficient sensory and decision making processes Sandamirskaya et al. (2022); Yang et al. (2023); Bhanja et al. (2023). The logical requirement of the system is that they rely on SNNs to cater to biologically plausible characteristics over traditional neurons in DNNs. To understand the impact of the design patterns associated with the hardware we use it to influence our algorithm design.

3 METHODS

3.1 NOTATION

At the core of policy learning environments is the Markov decision processes (MDP) which defines the problem. We define the tuple derived from the MDP for our continuous action space environments as follows: (s_t, a_t, r_t, s_{t+1}). Where s represents the state space at time step, t. $t+1$ is defined as the immediate following time step. We use r to represent the reward of the state transition from the given action, a. We follow the standard notation of π representing the policy with respect to action and state, $\pi(a|s)$.

3.2 SAC

The Soft Actor Critic (SAC) Haarnoja et al. (2018), is designed using stochastic policy and the maximum entropy objective, making it ideal for complex continuous environments. In addition, it has demonstrated high performance in the Farama Foundation Bipedal Walker environment Towers et al. (2024). Environments such as Bipedal Walker are often prone to higher instability due to the complex continuous nature of them. This allows our strategy to be tested in a well established framework with similar problems to that of real world applications. In this work, we use "stability" to refer to the consistency of policy improvement across training episodes and the avoidance of training collapse such as catastrophic forgetting.

Instability of policy learning is in part influenced by the underlying equations of the algorithm. Key to SAC is the transformation required in the actor to acquire action samples, requiring logarithms to aid stochastic exploration. We derive the following more stable form of this transformation from Equation 21 in Haarnoja et al. (2018).

Starting from the original policy log-probability:

$$\begin{aligned}\log \pi(a|s) &= \log \mu(u|s) - \sum_{i=1}^{D} \log(1 - \tanh^2(u_i)) \\ &= \log \mu(u|s) - 2\sum_{i=1}^{D} \log(\text{sech}(u_i)) \end{aligned} \tag{1}$$

where $\text{sech}(x) = 2/(e^x + e^{-x})$ and $\mu(u|s)$ is the policy density over the pre-tanh variable u. This leverages the softplus function for numerical stability during backpropagation. The equation derived reduces learning instability in complex continuous environments that have an increased susceptibility to numerical overflow, particular from $\tanh$ nonlinearity and Gaussian sampling operations. The full

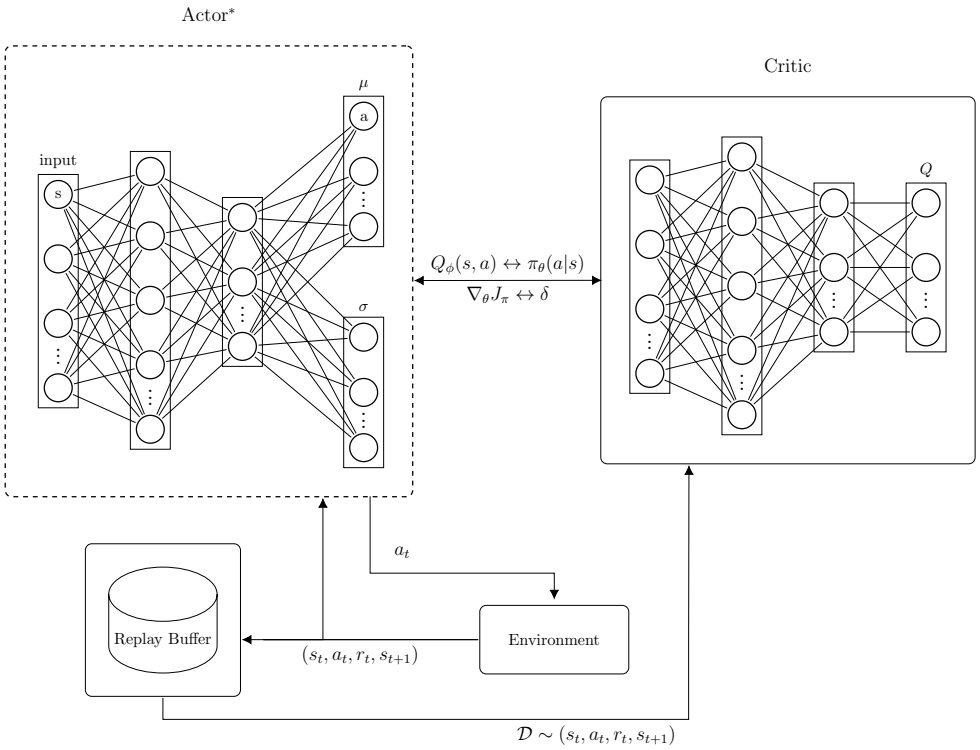


Figure 1: SANSAC/SAC Algorithm Diagram. Dashed border indicates spiking actor in SANSAC. All other components are identical between both algorithms. Critic outputs a state-action value $Q(s, a)$. Actor outputs a policy $\pi(a|s)$.

derivation of Equation 1 appears in Appendix A.2. Implementation of SAC follows the configuration shown in Figure 1, following closely to the presented logic of the original paper, where the algorithm consists of one actor network and two critic networks. Each network consists of three fully connected layers.

3.3 SANSAC

As SNNs have the capacity for physical neuromorphic hardware implementation, an algorithm inspired by the same framework necessitates implementation of spikes. Neuromorphic hardware has the capacity for advantages through the parallelization of data and computations. Through crossbar matrix multiplication simultaneous parallel calculations can be performed which can allow for lower energy costs and increased computation throughput.

Spiking Actor Networks (SAN), inspired by the Actor-Critic framework, utilize an equivalent architecture to the deep learning model used as the basis with the modification that the actor network be made a SNN while the critics remain deep critic networks. Originally proposed in a series of papers by Tang et al, Tang et al. (2020a;b), this methodology became the basis of our algorithm configuration. This hybrid RL framework relies on the final deployable model consisting of the spiking network and discarding other components. The concept then allows for full deployment of a neural network on neuromorphic hardware without the need to overhaul the training process for complex algorithms such as SAC which depends on multiple critics during training time. Thus, the computational burden on a neuromorphic chip can be reduced and allow for easier scaling of networks.

Through the combination of the SAC and SAN frameworks we propose the Spiking Actor Network Soft Actor Critic (SANSAC). The deep neural network is maintained for the two critics while the

actor network is implemented as a spiking neural network. The replay buffer equally critical to this framework remains the same as in a traditional SAC and is discarded after the completion of the training phase. In theory, this implementation would allow for a hybrid training system and fully neuromorphic agent after the training.

Spiked neural networks do not rely upon activation functions but rather rely on neurons with dynamics described by complex equations. As such SANSAC relies on the widely chosen LIF neuron. The LIF model is characterized by the following differential equation to describing the temporal evolution of the membrane potential:

$$\tau_{\mathrm{m}} \frac{\mathrm{d}V}{\mathrm{d}t} = -(V - V_{\mathrm{rest}}) + RI \tag{2}$$

where τ_{m} is the membrane time constant, V_{rest} is the resting potential, R is the membrane resistance, and I is the input current.

Traditional output of the LIF neuron layer is a spike train. This provides a hindrance to the decoding of network outputs into a continuous action space. An alternative proposed Chen et al. (2024),Chen et al. (2022) relies on modifying the output neuron layer form LIF to a nonspiking LIF layer referred to as Leaky-Integrate (LI) neurons. These output a constant voltage over the traditional spike train, which is key for decoding in continuous action spaces. SANSAC makes use of this through the final layer in the actor network which uses a LI neuron configuration while the rest of the layers use LIF neurons.

The implementation of SANSAC is accomplished through the SpikingJelly library for SNNs Fang et al. (2023). SANSAC in alignment with SAC does not utilize any additional encoding or decoding methods.

Artificial neural network agents in RL are fed state variables as described by MDPs where the network is typically trained through a combination of gradient descent and activation functions. However, SNNs are unable to take advantage of gradient descent due to the nonlinear and non-differentiable nature of spike outputs. The discontinous nature leads to the adoption of surrogate gradients Neftci et al. (2019). The surrogate gradient has been pivotal to the work on RL in SNNs, allowing for improved performance in complex and continuous environments. The surrogate gradient enables familiar gradient backpropagation seen in traditional neural networks. SANSAC implements the sigmoid surrogate gradient as described by equation 3.

$$\sigma(x, \alpha) = \frac{1}{1 + \exp(-\alpha x)} \tag{3}$$

The surrogate gradient creates network behavior closer to recurrent neural networks by applying backpropagation through an unrolled network with backpropagation through time (BPTT). Aside from the surrogate gradient, all loss function calculations remain the same as SAC.

4 Experiments

4.1 Experimental Settings

Both the actor and critic networks used in SANSAC and SAC utilized three layers with 2 input hidden dimensions. The same values for a given hidden dimension were used in both each agent's actor and critic networks. Table 1 demonstrates the pairs of hidden dimension variables used in comparison of SANSAC and SAC for quick reference.

	Hidden Dimension 1	Hidden Dimension 2
(1)	256	256
(2)	200	200
(3)	128	128
(4)	64	64

Table 1: Hidden Dimensions

All agents were trained for a maximum of 1200 episodes. Training was terminated early if no improvement in episodic reward was observed for 100 consecutive episodes after a minimum of 50 episodes had been completed. This early stopping procedure was applied identically to both SANSAC and SAC to prevent unnecessary computation when learning had stalled.

4.1.1 SEED GENERATION

Seed generation for all experiments was done randomly to eliminate bias. SAC and SANSAC seeds were kept in sync to each other for reproducibility.

4.2 COMPARISON OF ALGORITHMS

The evaluation of SANSAC and SAC is done through the Bipedal Walker environment from Farama Foundation's Gymnasium Towers et al. (2024) as pictured in Figure 2.

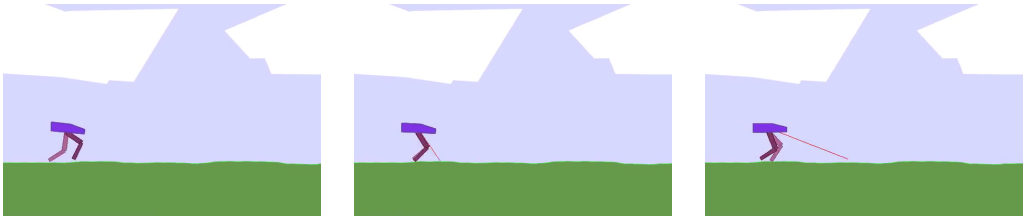


Figure 2: Gymnasium Bipedal Walker: Task is considered solved when 300 points are accumulated in 1600 steps

We consider the standard baseline performance of both algorithms to be the set of highest hidden dimensions presented in Table 1 (1) as it is close to typical hidden dimension settings commonly used and yields the most stable learning out of our tests.

Dimensions	Mean Reward $\pm$ SD		Success Rate		Statistical Tests	
	SANSAC	SAC	SANSAC	SAC	Mann-Whitney P	Cohen's d
(1)	273.9 ± 130.8	224.5 ± 162.9	70%	80%	0.7913	0.080
(2)	197.3 ± 178.2	259.8 ± 146.2	60%	80%	0.6232	-0.140
(3)	210.9 ± 185.9	158.7 ± 213.5	70%	60%	0.6776	0.120
(4)	26.8 ± 149.4	41.8 ± 138.5	10%	10%	0.5205	-0.180

Table 2: Performance comparison between SAC and SANSAC. Success Rate indicates percentage of agents achieving reward ≥ 300 in tests. Statistical tests show no significant differences between algorithms across all dimensions (Mann-Whitney $p > 0.05$). Cohen's d effect sizes: $|d| < 0.2$ indicates negligible difference, $0.2 \leq |d| < 0.5$ small, $0.5 \leq |d| < 0.8$ medium. All configurations show negligible to small effect sizes confirming the practical equivalence.

Comparing the performance of SANSAC and SAC in all hidden dimensions, we observe similar results for both algorithms. Through the use of Mann-Whitney U tests as shown in Table 2 we observe no statistically significant differences between the two algorithms in any configuration.

We demonstrate the reward curves across several hidden dimensions in Figure 4 and the actor loss curves similarly in Figure 5, revealing similar learning dynamics between algorithms. We include both for completeness to provide a complete picture of learning dynamics. The graphs reveal consistent patterns: both algorithms struggle at the lowest dimension (4), improved performance at intermediate dimensions (2, 3), and standard stable performance at higher dimensions (1). The similarity present in the convergence and shapes of the reward curves indicate that SANSAC learns similar behavior to that of SAC. Despite this pattern SANSAC's behavior observed in Figure 4d show an instability as reward drops compared to SAC.

Performance degradation as overall topology of the network decreases is an expected result as the decrease in neurons directly leads to a lowered capacity to learn. The lowest dimension (4), both

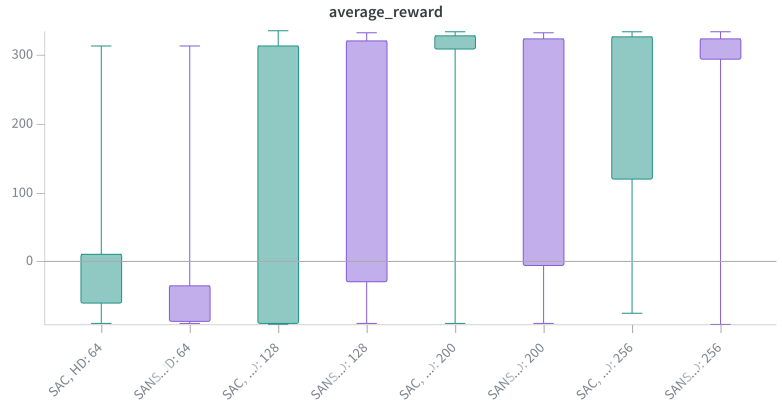
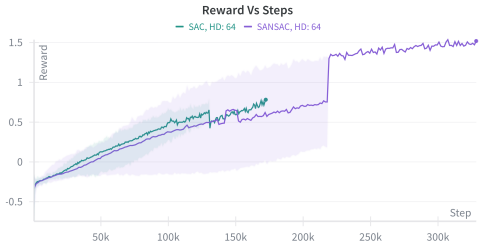
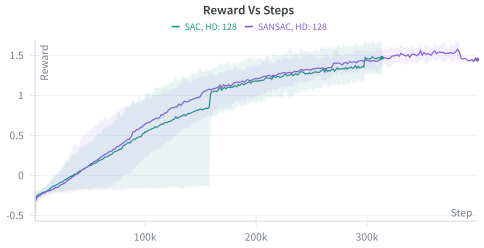


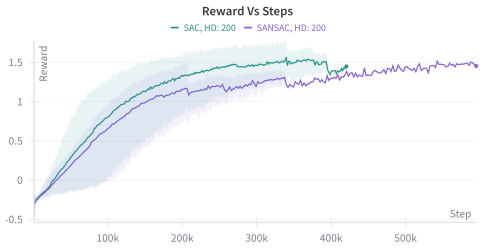
Figure 3: Box plots demonstrating the average reward of agents in testing. Configurations are labeled to separate SANSAC from SAC in each dimension test.



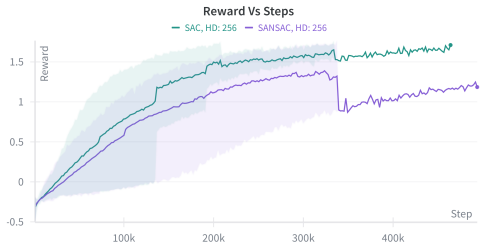
(a) Hidden dimension 64



(b) Hidden dimension 128



(c) Hidden dimension 200



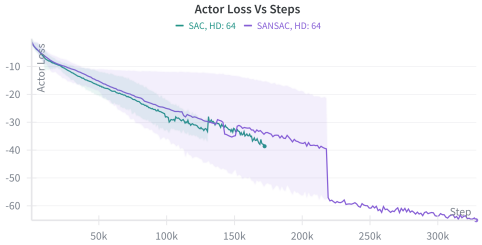
(d) Hidden dimension 256

Figure 4: Reward progression for SANSAC and SAC across different hidden dimensions. Curves show the mean episodic reward across 10 random seeds. Shaded regions represent the ± 1 standard deviation across seeds. Training was terminated after 1200 episodes or no reward improvement in the last 100 consecutive episodes. Truncated curves indicate early termination either through early convergence or early stopping, not a difference in evaluation protocol.

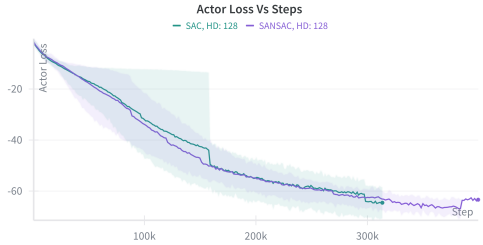
algorithms exhibit this performance drop with only 10% success rates. As the neurons present increases through higher hidden dimensions, both algorithms show substantial improvement with SANSAC showing performance competitive to SAC.

Through statistical analysis it is observed that SANSAC has an overall higher variance in performance compared to SAC in (2) and (3). Despite this, it is clear that SANSAC can perform on par with SAC.

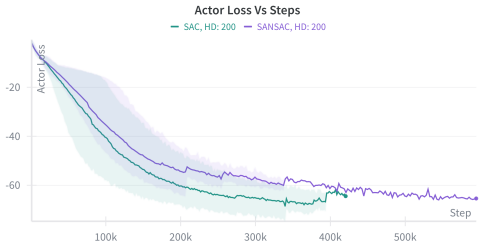
Notably within our experiments SANSAC demonstrates a consistently longer time required to train an agent. SANSAC takes 2x the amount of program time for the same amount of episodes in the tested configurations. This difference in training time is shown in Table 3. This behavior is expected as spiking neuron dynamics require temporal unrolling, creating a bottleneck on conventional hardware



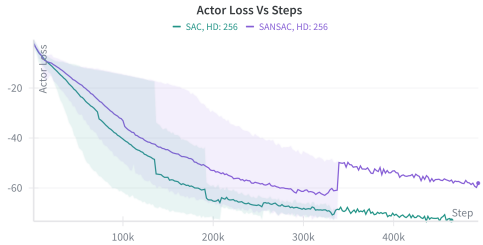
(a) Hidden dimension 64



(b) Hidden dimension 128



(c) Hidden dimension 200



(d) Hidden dimension 256

Figure 5: Actor Loss across steps for agent training averaged across 10 runs of random seeds. Curves show the mean episodic loss across 10 random seeds. Shaded regions indicate ± 1 standard deviation across seeds. The inverse relationship between loss and reward is expected as lower actor loss corresponds to higher expected returns.

which operates sequentially. On neuromorphic hardware, this unrolling would be parallelized creating potential room for speedup. We do not document energy comparison between SANSAC and SAC in conventional hardware as in our experiments reporting was inconsistent, hard to isolate, and overall inconsistent.

	Training Time (minutes)	
Hidden Dimension	**SANSAC**	**SAC**
(4)	44	10
(3)	66	26
(2)	80	23
(1)	54	24
Mean	**61**	**21**

Table 3: Average time taken by each configuration to train the agent across 10 random seeds.

5 LIMITATIONS

Our tests of SANSAC demonstrated competitive performance against SAC in multiple hidden dimension configurations. However, several limitations warrant consideration in interpreting the presented results. The comparison of both algorithms was primarily optimized and tested for the Bipedal Walker environment. While this environment represents a challenging continuous control task, the generalizability of our findings to other continuous environments remains to be validated through future experimentation. Additional findings may be captured through testing different environments which may not be captured in current analysis.

The observed computational time penalty for SANSAC is likely a result of temporal unrolling and bottlenecks inherent to simulating a neuromorphic designed algorithm on conventional architecture.

However, we cannot definitively quantify what portion of the observed overhead is fundamental to the algorithm versus the implementation-specific inefficiencies in the current set of experiments. The performance of SANSAC on neuromorphic hardware remains an open question not answered through this simulation-based study.

The observed variance in success rates of SANSAC compared to SAC suggest higher sensitivity, potentially due to the stochastic nature of spiking neurons or the surrogate gradient approximation. This sensitivity also affected early stopping behavior as a higher variance could delay convergence detection. This creates additional challenges for achieving consistent results and may affect scalability to larger-scale problems.

6 Conclusion

The integration of neuromorphic hardware and its associated design techniques remain underutilized in RL architectures maintained in conventional computing. We introduced SANSAC, a Spiking Actor Network variant of the Soft Actor Critic framework designed to bridge this gap while isolating key behaviors and dynamics necessary to document prior to neuromorphic hardware deployment. Our evaluation demonstrates that SANSAC achieves performance statistically indistinguishable from SAC across multiple hidden dimension configurations. While SANSAC incurs computational overhead during simulation on conventional hardware, this presents a necessary trade-off for maintaining compatibility with neuromorphic hardware architectures where the design would yield energy efficiency benefits.

The success of SANSAC in learning complex continuous control tasks validates the viability of spiking neural networks for reinforcement learning applications. Our findings show that the constraints imposed by neuromorphic design such as surrogate gradients and temporal dynamics do not inherently limit performance compared to traditional deep networks when implemented. These results provide a foundation for future work on neuromorphic reinforcement learning, demonstrating that algorithm design for emerging hardware platforms can maintain competitive performance during the development and training phases conducted on conventional computers. The path toward energy-efficient, neuromorphic RL systems appears viable, with SANSAC serving as a proof of concept for this approach.

7 Future Work

Improvement upon this work would be through rigorous testing against other behavior affecting variables including neuron type and encoding and decoding behavior. The current implementation uses LIF neurons and Sigmoid surrogate gradients, but exploration of alternative neuron models and gradient approximations could yield performance improvements.

To expand our research into more practical implementations that help bridge the gap between neuromorphic computing and reinforcement learning we hope to see the design and implementation of similar algorithms on physical neuromorphic hardware. While most neuromorphic chips remain highly experimental and difficult to acquire we suggest the implementation of these algorithms on FPGAs. While complex in implementation these offer a relatively unexplored path that is more accessible and highly customizable.

Further investigation into the scalability of SANSAC and similar neuromorphic designed algorithms to more complex environments and larger network architectures would provide valuable insights into its practical limitations and advantages. Additionally, exploration of hybrid training strategies that leverage both conventional and neuromorphic hardware during different phases of learning could help mitigate the computational overhead observed in our experiments.

Acknowledgements

We gratefully acknowledge the support of the NMT Research Opportunities Cooperative Program and the Sophomore Research Program by its sponsors: the Air Force Research Laboratory, Sandia National Laboratories, Alumni Donors, and the Offices of Financial Aid and Academic Affairs of New Mexico Tech.

REFERENCES

L.F Abbott. Lapicque's introduction of the integrate-and-fire model neuron (1907). *Brain Research Bulletin*, 50(5-6):303–304, 1999. ISSN 0361-9230. doi: 10.1016/S0361-9230(99)00161-6. URL https://libkey.io/10.1016/S0361-9230(99)00161-6.

Tapanta Bhanja, Jonathan Nußbaum, Khalil Abuibaid, Tatjana Legler, Achim Wagner, and Martin Ruskowski. The need for neuromorphic computing in industrial robotics. In *2023 International Conference on Artificial Intelligence and Power Engineering (AIPE)*, pp. 42–49, 2023. doi: 10.1109/AIPE58786.2023.00015.

Ding Chen, Peixi Peng, Tiejun Huang, and Yonghong Tian. Deep reinforcement learning with spiking q-learning. *CoRR*, abs/2201.09754, 2022. URL https://arxiv.org/abs/2201.09754.

Ding Chen, Peixi Peng, Tiejun Huang, and Yonghong Tian. Fully spiking actor network with intra-layer connections for reinforcement learning, 2024. URL https://arxiv.org/abs/2401.05444.

Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, Yuyun Liao, Chit-Kwan Lin, Andrew Lines, Ruokun Liu, Deepak Mathaikutty, Steven McCoy, Arnab Paul, Jonathan Tse, Guruguhanathan Venkataramanan, Yi-Hsin Weng, Andreas Wild, Yoonseok Yang, and Hong Wang. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018. doi: 10.1109/MM.2018.112130359.

Wei Fang, Yanqi Chen, Jianhao Ding, Zhaofei Yu, Timothée Masquelier, Ding Chen, Liwei Huang, Huihui Zhou, Guoqi Li, and Yonghong Tian. Spikingjelly: An open-source machine learning infrastructure platform for spike-based intelligence. *Science Advances*, 9(40):eadi1480, 2023. doi: 10.1126/sciadv.adi1480. URL https://www.science.org/doi/abs/10.1126/sciadv.adi1480.

Ruan de Rezende Faria, Bruno Didier Olivier Capron, Argimiro Resende Secchi, and Maurício B. de Souza. Where reinforcement learning meets process control: Review and guidelines. *Processes*, 10(11), 2022. ISSN 2227-9717. doi: 10.3390/pr10112311. URL https://www.mdpi.com/2227-9717/10/11/2311.

Nicolas Frémaux and Wulfram Gerstner. Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules. *Frontiers in Neural Circuits*, 9, 2016. ISSN 1662-5110. doi: 10.3389/fncir.2015.00085. URL https://www.frontiersin.org/journals/neural-circuits/articles/10.3389/fncir.2015.00085.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290, 2018. URL http://arxiv.org/abs/1801.01290.

Danijar Hafner, Timothy P. Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *CoRR*, abs/1912.01603, 2019. URL http://arxiv.org/abs/1912.01603.

A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117(4):500–544, 1952. doi: https://doi.org/10.1113/jphysiol.1952.sp004764. URL https://physoc.onlinelibrary.wiley.com/doi/abs/10.1113/jphysiol.1952.sp004764.

E.M. Izhikevich. Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, 14(6):1569–1572, 2003. ISSN 1045-9227. doi: 10.1109/TNN.2003.820440. URL https://libkey.io/10.1109/TNN.2003.820440.

Bruce W. Knight. Dynamics of encoding in a population of neurons. *Journal of General Physiology*, 59(6):734–766, 06 1972. ISSN 0022-1295. doi: 10.1085/jgp.59.6.734. URL https://doi.org/10.1085/jgp.59.6.734.

Wolfgang Maass. Networks of spiking neurons: The third generation of neural network models. *Electron. Colloquium Comput. Complex.*, TR96, 1996. URL https://api.semanticscholar.org/CorpusID:1753085.

Yao Mu, Yuzheng Zhuang, Bin Wang, Guangxiang Zhu, Wulong Liu, Jianyu Chen, Ping Luo, Shengbo Eben Li, Chongjie Zhang, and Jianye Hao. Model-based reinforcement learning via imagination with derived memory. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2021. Curran Associates Inc. ISBN 9781713845393.

Emre O. Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks. *CoRR*, abs/1901.09948, 2019. URL http://arxiv.org/abs/1901.09948.

Yulia Sandamirskaya, Mohsen Kaboli, Jorg Conradt, and Tansu Celikel. Neuromorphic computing hardware and neural architectures for robotics. *Science Robotics*, 7(67):eabl8419, 2022. doi: 10.1126/scirobotics.abl8419. URL https://www.science.org/doi/abs/10.1126/scirobotics.abl8419.

Shruti R.; Parsa Maryam; Mitchell J. Parker; Date Prasanna; Kay Bill Schuman, Catherine D.; Kulkarni. Opportunities for neuromorphic computing algorithms and applications. *Nature Computational Science*, 2(1):10–19, 2022. ISSN 2662-8457. doi: 10.1038/s43588-021-00184-y. URL https://libkey.io/10.1038/s43588-021-00184-y.

Cong Shi, Jing Lu, Ying Wang, Ping Li, and Min Tian. Exploiting memristors for neuromorphic reinforcement learning. In *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pp. 1–4, 2021. doi: 10.1109/AICAS51828.2021.9458542.

Yinqian Sun, Feifei Zhao, Zhuoya Zhao, and Yi Zeng. Multi-compartment neuron and population encoding powered spiking neural network for deep distributional reinforcement learning. *Neural Networks*, 182:106898, 2025. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2024.106898. URL https://www.sciencedirect.com/science/article/pii/S089360802400827X.

Guangzhi Tang, Neelesh Kumar, and Konstantinos P. Michmizos. Reinforcement co-learning of deep and spiking neural networks for energy-efficient mapless navigation with neuromorphic hardware. *CoRR*, abs/2003.01157, 2020a. URL https://arxiv.org/abs/2003.01157.

Guangzhi Tang, Neelesh Kumar, Raymond Yoo, and Konstantinos P. Michmizos. Deep reinforcement learning with population-coded spiking neural network for continuous control. *CoRR*, abs/2010.09635, 2020b. URL https://arxiv.org/abs/2010.09635.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

Robert Urbanczik and Walter Senn. Reinforcement learning in populations of spiking neurons. *Nature Precedings*, 3, 06 2008. doi: 10.1038/npre.2008.1976.1.

Haibing Wang, Zhen He, Tengxiao Wang, Junxian He, Xichuan Zhou, Ying Wang, Liyuan Liu, Nanjian Wu, Min Tian, and Cong Shi. Triplebrain: A compact neuromorphic hardware core with fast on-chip self-organizing and reinforcement spike-timing dependent plasticity. *IEEE Transactions on Biomedical Circuits and Systems*, 16(4):636–650, 2022. doi: 10.1109/TBCAS.2022.3189240.

Zhongrui Wang, Can Li, Wenhao Song, Mingyi Rao, Daniel Belkin, Yunning Li, Peng Yan, Hao Jiang, Peng Lin, Miao Hu, John Paul Strachan, Ning Ge, Mark Barnell, Qing Wu, Andrew G Barto, Qinru Qiu, R Stanley Williams, Qiangfei Xia, and J Joshua Yang. Reinforcement learning with analogue memristor arrays. *Nat. Electron.*, 2(3):115–124, March 2019.

Nan Wu, Adrien F. Vincent, Dmitri B. Strukov, and Yuan Xie. Memristor hardware-friendly reinforcement learning. *CoRR*, abs/2001.06930, 2020. URL https://arxiv.org/abs/2001.06930.

Yi Yang, Chiara Bartolozzi, Haiyan H. Zhang, and Robert A. Nawrocki. Neuromorphic electronics for robotic perception, navigation and control: A survey. *Engineering Applications of Artificial Intelligence*, 126:106838, 2023. ISSN 0952-1976. doi: https://doi.org/10.1016/j.engappai.2023.106838. URL https://www.sciencedirect.com/science/article/pii/S0952197623010229.

Yang Zhang, Zhongrui Wang, Jiadi Zhu, Yuchao Yang, Mingyi Rao, Wenhao Song, Ye Zhuo, Xumeng Zhang, Menglin Cui, Linlin Shen, Ru Huang, and J. Joshua Yang. Brain-inspired computing with memristors: Challenges in devices, circuits, and systems. *Applied Physics Reviews*, 7(1):011308, 01 2020. ISSN 1931-9401. doi: 10.1063/1.5124027. URL https://doi.org/10.1063/1.5124027.

A Appendix

A.1 Hyperparameters

Table A.1: Hyperparameters

Parameter	Value
learning rate	$3 \cdot 10^{-4}$
discount (γ)	0.975
replay buffer size	10^6
number of samples per minibatch	256
target smoothing coefficient (τ)	0.005
Reward Scale	5
Action Dimensions	4
State Dimensions	24
Alpha	0.2
surrogate gradient type	sigmoid
SANSAC timesteps (T)	16
optimizer	Adam

All data gathered was done on a Nvidia GeForce RTX 4070.

A.2 Policy Log-Probability Derivation

Beginning with the original SAC policy log-probability:

$$\begin{aligned}
\log \pi(a|s) &= \log \mu(u|s) - \sum_{i=1}^{D} \log(1 - \tanh^2(u_i)) \\
&= \log \mu(u|s) - 2\sum_{i=1}^{D} \log(\operatorname{sech}(u_i)) \quad \text{(using identity } 1 - \tanh^2(x) = \operatorname{sech}^2(x)\text{)} \\
&= \log \mu(u|s) - 2\sum_{i=1}^{D} \left[\log 2 - u_i - \log(1 + e^{-2u_i})\right] \\
&= \log \mu(u|s) - 2\sum_{i=1}^{D} \left[\log 2 - u_i - \operatorname{softplus}(-2u_i)\right]
\end{aligned}$$

The derived transformation replaces the unstable $\tanh$ operations with the alternative $\operatorname{softplus}$ function ($\operatorname{softplus}(x) \triangleq \log(1 + e^x)$), ensuring gradient stability for continuous control tasks during the policy network backpropagation. This avoids vanishing gradients while maintaining differentiability critical for continuous control environments.