Improved robustness of reinforcement learning policies upon conversion to spiking neuronal network platforms applied to ATARI games

Anonymous Authors[1]

Abstract

Various implementations of Deep Reinforcement Learning (RL) demonstrated excellent performance on tasks that can be solved by trained policy, but they are not without drawbacks. Deep RL suffers from high sensitivity to noisy and missing input and adversarial attacks. To mitigate these deficiencies of deep RL solutions, we suggest involving spiking neural networks (SNNs). Previous work has shown that standard Neural Networks trained using supervised learning for image classification can be converted to SNNs with negligible deterioration in performance. In this paper, we convert Q-Learning ReLU-Networks (ReLU-N) trained using reinforcement learning into SNN. We provide a proof of concept for the conversion of ReLU-N to SNN demonstrating improved robustness to occlusion and better generalization than the original ReLU-N. Moreover, we show promising initial results with converting full-scale Deep Q-networks to SNNs, paving the way for future research.

1. Introduction

Recent advancements in deep reinforcement-learning (RL) have achieved astonishing results surpassing human performance on various ATARI games (Mnih et al., 2015; Hasselt et al., 2016; Wang et al., 2016). However, deep RL is susceptible to adversarial attacks similarly to deep learning (Huang et al., 2017). The vulnerability to adversarial attacks is due to the fact that deep RL uses gradient descent to train the agent. Another consequence of the gradient descent algorithm is that the trained agent learns to focus on a few sensitive areas and when these areas are occluded or perturbed, the performance of the RL agent deteriorates. Moreover, there is evidence that the policies learned by the networks in deep RL algorithms do not generalize well and the performance of the agent deteriorates when it encounters a state that it has not seen before even if it is similar to other states (Witty et al., 2018).

Biological systems tend to be very noisy by nature (Richardson & Gerstner, 2006; Stein et al., 2005), but they can still operate well even under harsh conditions that affect their internal state and input. Spiking Neural Networks (SNNs) are considered to be closer to biological neurons due to their event-based nature; they are often termed the third generation of neural networks (Maass, 1996). A spike is the quantification of the internal and external process of the neuron and is always equal to other spikes. Therefore, the individual neuron can serve as a small bottleneck that gives the ability to sustain low intermittent noise and not propagate the noise further. Moreover, spiking neurons as a group in a network can damp the noise even further due to their collective effect and their architectural connectivity (Hazan & Manevitz, 2012). However, SNNs are typically harder to train using backpropagation due to the non-differentiable nature of the spikes (Pfeiffer & Pfeil, 2018).

Much of the recent work with SNNs has focused on implementing methods similar to backpropagation (Huh & Sejnowski, 2018; Wu et al., 2018) or using biologically inspired learning rules like spike-timing-dependent plasticity (STDP) to train the network (Bengio et al., 2015; Diehl & Cook, 2015; Gilra & Gerstner, 2018; Ferr et al., 2018). One of the benefits of using SNNs is their potential to be more energy efficient and faster than rectified linear unit networks (ReLU-N), particularly so on dedicated neuromorphic hardware (Martí et al., 2016).

Using SNNs in a RL environment seems almost natural since many animals learn to perform certain tasks using a variation of semi-supervised and reinforcement learning. Moreover, there is evidence that biological neurons also learn using evaluative feedback from neurotransmitters such as dopamine (Wang et al., 2018) (e.g., in the postulated dopamine reward prediction-error signal (Schultz, 2016)). However, since spiking neurons are fundamentally different from artificial neurons, it is not clear if SNNs are as capable as ReLU-Ns in machine learning domains. This raises the questions: Do SNNs have the capability to represent the same functions as ReLU-N? To be more specific, can SNNs represent complex policies that can successfully play Atari games? If so, do they have any advantages in handling noisy inputs?

We answer these questions by demonstrating that ReLU-networks trained using existing reinforcement learning algorithms can be converted to SNN with similar performance on the reinforcement learning task when playing Atari Breakout game. Furthermore, we show that such converted SNNs are more robust than the original ReLU-Ns. Finally, we demonstrate that full-sized deep Q-networks (DQN) (Mnih et al., 2015) can also be converted to SNNs and maintain its better than human performance, paving the way for future research in robustness and RL with SNNs.

2. Background

2.1. Arcade learning environment

The Arcade learning environment (ALE) (Bellemare et al., 2013) is a platform that enables researchers to test their algorithms on over 50 Atari 2600 games. The agent sees the environment through image frames of the game, interacts with the environment with 18 possible actions, and receives feedback in the form of the change in the game score. The games were designed for humans and thus are free from experimenter bias. The games span many different genres that require the agent/algorithm to generalize well over various tasks, difficulty levels, and timescales. ALE thus has become a popular test-bed for reinforcement learning.

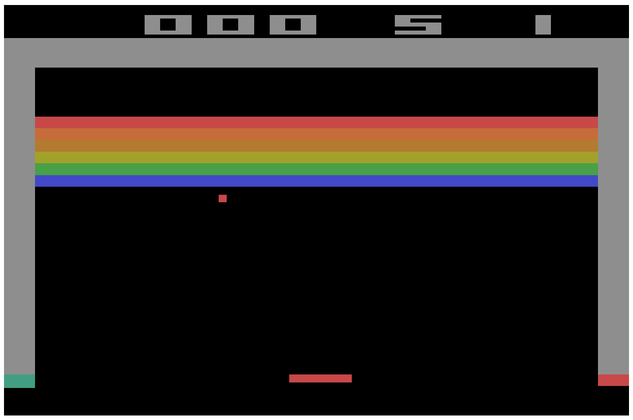


Figure 1. Screenshot of Atari 2600 Breakout game

Breakout: We demonstrate our results on the game of Breakout. Breakout is a game similar to the popular game Pong. The player controls a paddle at the bottom of the screen. There are rows of colored bricks on the upper part of the screen. A ball bounces in between the bricks and the player controlled paddle. If the ball hits a brick, the brick breaks and the score of the game is increased. However, if the ball falls below the paddle, the player loses a life. The game starts with five lives, and the player/agent is supposed to break all the bricks before they run of lives. Figure 1 shows a frame of the game.

2.2. Deep Q-Networks

Reinforcement learning algorithms train a policy π to maximize the expected cumulative reward received over time. Formally, this process is modelled as a Markov decision process (MDP). Given a state-space $\mathcal{S}$ and an action-space $\mathcal{A}$, the agent starts in an initial state $s_0 \in \mathcal{S}$, from a set of possible start states $S_0 \in \mathcal{S}$. At each time-step t, starting from $t = 0$, the agent takes an action a_t to transition from s_t to s_{t+1}. The probability of transitioning from state s to state s' by taking action a is given by the transition function $P(s, a, s')$. The reward function $R(s, a)$ defines the expected reward received by the agent after taking action a on state s.

A policy π is defined as the conditional distribution of actions given the state $\pi(s, a) = Pr(A_t = a|S_t = s)$. The Q-value or action-value of a state-action pair for a given policy, $q^\pi(s, a)$, is the expected return following policy π after taking the action a from state s.

$$q^\pi(s, a) = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k R_{t+k} | S_t = s, A_t = a, \pi] \tag{1}$$

where γ is the discount factor. The action-value function follows a Bellman equation that can be written as:

$$q^\pi(s_t, a_t) = r_t + \gamma \max_{a_{t+1}} q^\pi(s_{t+1}, a_{t+1}) \tag{2}$$

Many widely used reinforcement learning algorithms first approximate the Q-value and then select the policy that maximizes the Q-value at each step to maximize returns (Sutton & Barto, 2018). Deep Q-networks (DQN) (Mnih et al., 2015) are one such algorithm that uses deep artificial neural networks to approximate the Q-value. The neural network can learn policies from only the pixels of the screen and the game score and has been shown to surpass human performance on many of the Atari 2600 games.

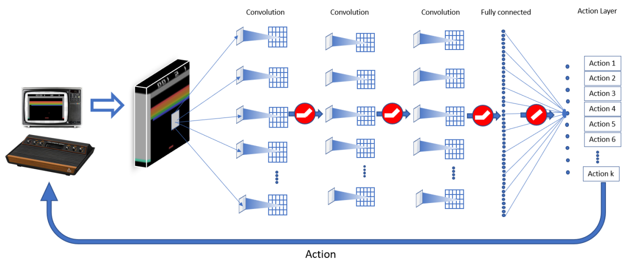


Figure 2. Architecture of Deep Q-networks; following Mnih et al. (2015); ReLu nonlinear units are emphasized by red circles.

2.3. Spiking neurons

SNNs may use any of the various neuron models (W. Gerstner, 2002; Tuckwell, 1988). For our experiments, we use four different variations of spiking neuron. We use the notation below used to describe the variance neurons: τ is the time constant. V is the membrane potential voltage. v_{rest} is the resting membrane potential. v_{thresh} is the neuron threshold.

1. Integrate-and-fire (IF) neuron: The IF neuron is the simplest form of spiking neuron models. The neuron simply integrates input until the membrane potential V exceeds the voltage threshold $V_{threshold}$ and a spike is generated. Once the spike is generated, the membrane potential is reset to V_{reset}.

$$\tau \frac{dv}{dt} = \sum_{i=1}^{n} W_i * Input_i \tag{3}$$

2. Subtractive Integrate-and-fire (SubIF) neuron: The SubLIF (Cassidy et al., 2013; Diehl et al., 2016; Rueckauer et al., 2017) behaves similar to the IF neuron with one small change, when the membrane potential voltage exceeds threshold value, the neuron emits a spike and resets its membrane voltage to $V_{reset} + (V - V_{threshold})$. By adding the overshoot voltage the neuron "remember" the excessive voltage from the last spike and will be more prone to be excited with the next incoming inputs. This reduces the information lost when spiking in SNNs converted from ReLU-N.

3. Leaky integrate-and-fire (LIF) neuron: The LIF neuron behaves similarly to the IF neuron. However, for every time-step that its membrane potential is above the resting potential, the neuron leaks a constant amount of current.

$$\tau \frac{dv}{dt} = (v_{rest} - V_{t-1}) + \sum_{i=1}^{n} W_i * Input_i \tag{4}$$

4. Stochastic leaky integrate-and-fire neuron: The stochastic leaky integrate-and-fire neuron is based on the LIF neuron. However, the neuron may spike if its membrane potential is below the threshold with probability proportional to its membrane potential (escape noise). The escape noise (σ) is described here:

$$\sigma = \begin{cases} 1/\tau_\sigma \exp(\beta_\sigma (V_t - V_{threshold})) & \text{if less than 1} \\ 1 & \text{otherwise} \end{cases} \tag{5}$$

where τ_σ and β_σ are constant positive parameters. For this paper, we set both τ_σ and β_σ to 1.

For all the spiking models listed above, after every spike, the neuron enters a refractory period during which they are unable to spike or integrate input. For this paper, we ignore the refractory period for simplicity in the conversion from artificial neurons. For a complete list of the parameters used for LIF and stochastic LIF see supplementary materials Table 2.

3. Related work

Much of the recent work has focused on the ReLU-N to SNN conversion. Prez-Carrasco et al. (2013) first introduced the idea of converting CNN to spiking neurons with the aim of processing inputs from event-based sensors. Cao et al. (2015) suggested that frequency of spikes of the spiking neuron is closely related to the activations of a rectified linear unit (ReLU) and reported good performance on computer vision benchmarks. Diehl et al. (2015) proposed a method of weight normalization that re-scales the weights of the SNN to reduce the errors due to excessive or too little fringing of neurons. They also showed near lossless conversion of ReLU-Ns for the MNIST classification task. Rueckauer et al.Rueckauer et al. (2016; 2017)) demonstrated spiking equivalents of a variety of common operations used in deep convolutional networks like max-pooling, softmax, batch-normalization and inception modules. This allowed them to convert popular CNN architectures like VGG-16 (Simonyan & Zisserman, 2014), Inception-V3(Szegedy et al., 2016), BinaryNet(Courbariaux et al., 2016), etc. They achieved near lossless conversion of these networks. There has been no previous work on conversion of Deep Q-networks into SNN to our knowledge. Figure 3 shows the network in Figure 2 converted to a spiking neural network (SNN).

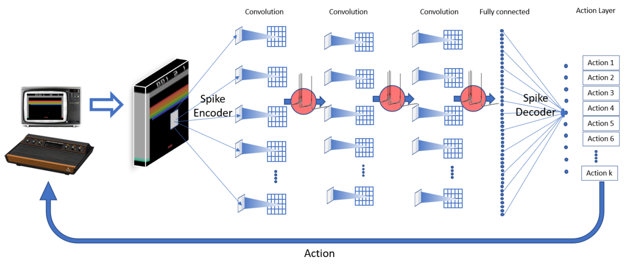


Figure 3. Network architecture following (Mnih et al., 2015), after converting ReLu nonlinearity to spiking network.

4. Methods

We trained each network using the DQN algorithm (Mnih et al., 2015). We started by testing our methods on shallow ReLU-Ns with one hidden layer and then move on to full-sized deep Q-networks with the same architecture as Mnih et al. (2015). We trained the smaller networks using a replay memory size of 200000 and initial replay memory size of 50000. We trained the network over 30000 episodes.

The rest of the hyper-parameters we used are same as in Mnih et al. (2015)'s work. For a complete list of the hyper-parameters used see supplementary materials Table 1.

The trained ReLU-Ns are then converted to SNN. For the converted SNN, The firing frequency of the spiking neurons in the output layer is proportional to the Q-value of the corresponding action.

We simulate spiking neurons using the PyTorch based open source library BindsNET (Hazan et al., 2018). Testing SNN based agents in the ALE is a computationally heavy task. We use BindsNET as it allows users to leverage GPUs to simulate the SNN and speed up testing.

4.1. Network architecture

Typically, the network used to train on Atari games using the DQN algorithm consists of multiple convolutional layers followed by fully connected layers (Mnih et al., 2015). However, to reduce the complexity of the network and reduce the number of parameters, we choose a shallow fully connected network with one hidden layer for our initial experiments.

The ReLU-N consists of 80x80 pixel input followed by a fully connected hidden layer with 1000 ReLU neurons. The output layer is a fully connected layer with 4 neurons that give the estimate of the optimal action-value of each of the 4 possible actions in the Breakout game.

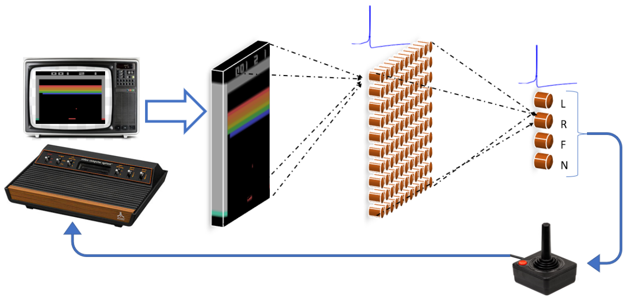


Figure 4. Network architecture: The input to the network consists of an 80x80 image produced by preprocessing the frames of the game. The hidden layer consists of 1000 neurons followed by the output layer. The size of the output layer is equal to the number of possible actions for the game.

Figure 4 shows the network architecture of the shallow SNN. The network architecture of the shallow SNN is similar to the shallow ReLU-N except that the neurons are replaced by spiking neurons and the ReLU non-linearity is removed.

4.2. Experiments

The ReLU-N can be converted to SNN by replacing the ReLU neurons with spiking neurons. However, the result of this straight forward conversion usually causes to a very little spiking activity in the network. Therefore, the network needs to run for a large number of time steps on given input to generate enough meaningful activity for a good estimate of the Q-values. In order to expedite the process and increase the spiking activity, the weights need to be scaled up. Generally, the weights of deeper layers need to be scaled higher than weights of the shallower layers. We can treat the scale of the weights as parameters that need to be adjusted with a constant run-time for each input. All the weights of the same layer are scaled by the same factor thus preserving the learned filters. To search for the weight scale parameters, we can use many different methods. While Rueckauer et al. (2017) showed one way of normalizing the weights, we also employed other methods such as grid search and particle swarm optimization (Clerc, 2012) to search for the optimal parameters. For our experiments, we run the SNN for 500 time-steps on each input.

Binary input

The first part of our experiments uses binary pixel inputs. Each state consists of an 80x80 image of binary pixels. The frames from the Gym environment are pre-processed to create the state.

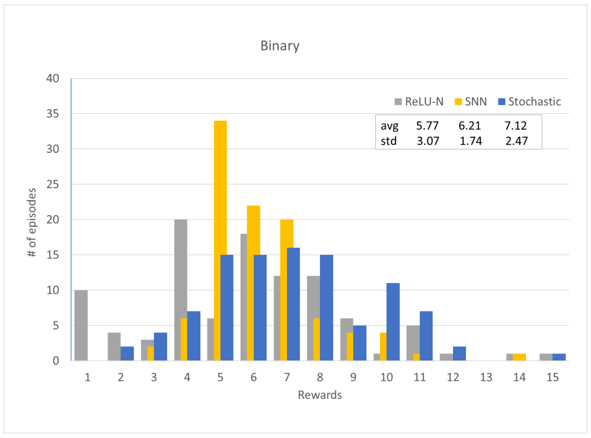
Each frame from the gym environment is cropped to remove the text above the screen displaying the score and the number of lives left. The image is then re-sized to an 80x80 image and converted to a binary image. The previous frame is then subtracted from the current frame while clamping all the negative values to 0. We then add the most recent four such difference frames to create a state for the RL environment. Thus, a state is an 80x80 binary image containing the movement information of the last four states. From this image, however, it is not possible to detect the direction of the movement from the image. This, we believe, restricts the performance of the agent on the game.

Grayscale input

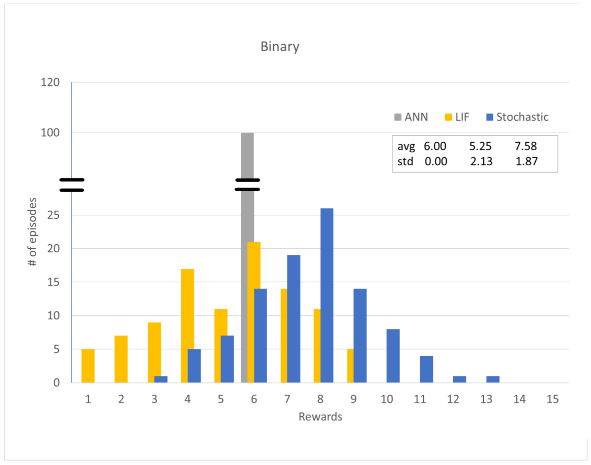
The binary input does not contain information about the direction of the ball movement which we believe can confuse the agent. To alleviate this problem, we weighted each frame according to time and added them to create the state. The most recent frame has the highest weight, and the least recent frame has the least weights. At time t the state is made up of the sum of the most recent 4 frames as follows:

$$S_t = F_t * 1 + F_{t-1} * 0.75 + F_{t-2} * 0.5 + F_{t-3} * 0.25 \quad (6)$$

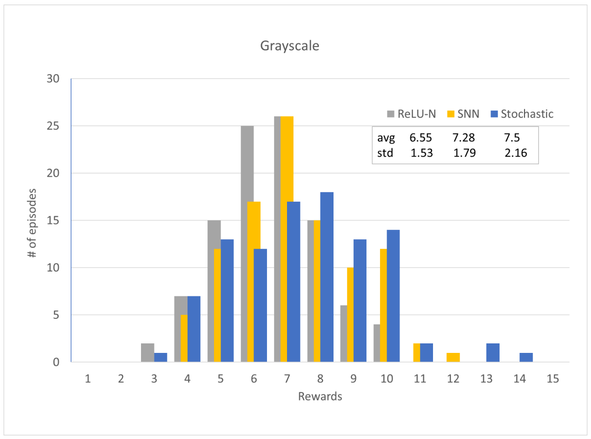
Where S_t and F_t are the state and the frame at time t respectively.



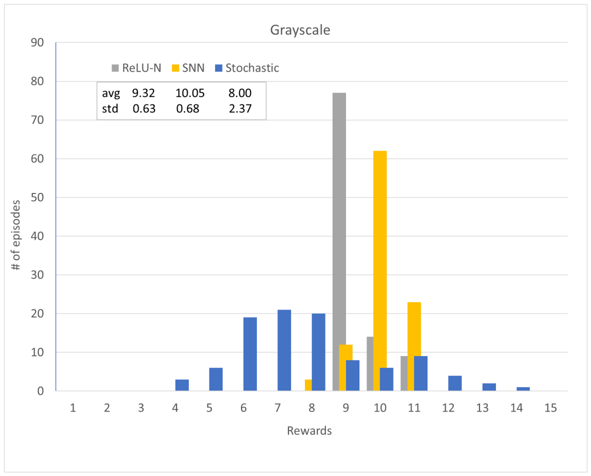
(a) 0.05 epsilon greedy binary input



(b) Greedy binary input



(c) 0.05 epsilon greedy grayscale input



(d) Greedy grayscale input

Figure 5. Performance of the networks for Binary and Grayscale inputs. Each plot shows the reward distribution over 100 episodes using 0.05 epsilon greedy policy.

Robustness

Recent work has shown that deep Q-networks are vulnerable to white-box and black-box adversarial attacks (Huang et al., 2017). Witty et al. (2018) also showed that the policies learned by the DQN algorithm generalize poorly for the states of the game that the agent has not seen during training. To test the robustness of the SNN against the ReLU-N, we test the performance of each network when a 3-pixel thick horizontal bar of pixels spanning the entire width of the input is occluded. The thickness of the occlusion bar corresponds to the thickness of the paddle on the screen after preprocessing. We tested the performance for every possible position of the bar on the screen. The position of the occlusion bar does not change during the episode. This is a challenging task since the bar sometimes partially or wholly occludes the position of the ball or the paddle; however, it tests the robustness and generalization of the policies represented by both the networks.

5. Results

The experiments below show the results of 100 episodes on two different inputs (Binary or Grayscale) using two policies (greedy and 0.05 epsilon-greedy). We tested the shallow SNN using LIF neurons and stochastic LIF neurons. We refer to the SNN using LIF neurons as SNN and the SNN using stochastic LIF neurons as stochastic SNN.

Binary input

The results demonstrate that SNNs are capable of representing policies that perform even better than the ReLU-N they are converted from. Figure 5 shows the performance of the ReLU-N against the performance of SNN and the stochastic SNN for binary input. We can see that the stochastic SNN performs better on average than the ReLU-N it is converted from. The optimal parameters for this binary input spiking neural networks were found using grid search.

Grayscale input

The results show that the performance for the grayscale input is higher then the binary input for both networks (SNN and ReLU-N) as shown in figure 5. Table 1 summarizes the best performance for each method of input for each network. We also see that the standard deviation of the rewards gained by the SNN is lower and the behavior is less random than for the binary input, due to proper weight normalization. We employed particle swarm optimization (Clerc, 2012) to search for the optimal weight scaling parameters. The scale of each of the two layers was treated as a parameter; thus the dimension of the search space is 2. The swarm size was set to 13. The stochastic LIF network has a smoother surface of performance over the parameter space than the LIF network. This suggests that the stochastic LIF network is more robust to change in the scaling of its weights. The escape noise of the stochastic LIF neuron can be tuned to improve the performance further however we leave that to future work.

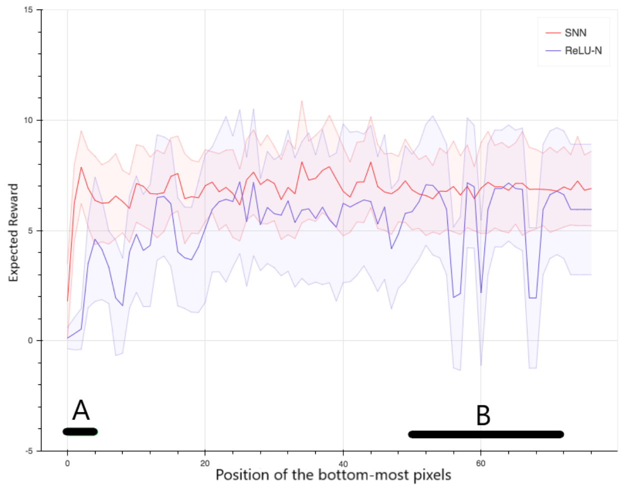
Input	ReLU-N	SNN	Stochastic
0.05 Epsilon greedy			
Binary	5.77 ± 3.07	6.21 ± 1.74	7.12 ± 2.47
Grayscale	6.55 ± 1.53	7.28 ± 1.79	7.5 ± 2.16
greedy			
Binary	6.0 ± 0	5.25 ± 2.13	7.58 ± 1.87
Grayscale	9.32 ± 0.63	10.05 ± 0.68	8.0 ± 2.37

Table 1. Best performance achieved for different inputs and networks. Each value represents an average of 100 episodes.

Robustness

Figure 6 shows the performance of the ReLU-N and SNN for the robustness task. The x-axis represents the vertical position of the bottom-most occluded pixels. Thus as we move from left to right on the plot, the 3-pixel thick occlusion bar moves from bottom to the top of the screen. Figure 6 shows the result of 77 experiments, one for each possible position of the horizontal occlusion bar. Each experiment was run for 100 episodes using 0.05 epsilon greedy policy.

Our experiments on robustness show that SNNs are much more robust to occlusions on input as compared to ReLU-N even though they share the same weights. The ReLU-N trained using backpropagation is very sensitive to occlusions and perturbations at a few places in the input. When these areas are occluded, the ReLU-N performs poorly. One such area is near the bottom of the screen (marked in Figure 6 by A). Occlusion in this area results in drastic decrease in the performance of the ReLU-N. This is understandable as this area contains the paddle and also shows the position of the ball just before it hits the paddle or falls below the screen. Surprisingly, occluding the neighborhood of area A has much less negative impact on the performance of the SNN as compared to ReLU-N. Once the paddle is visible, we see that the SNN has no significant loss in performance.



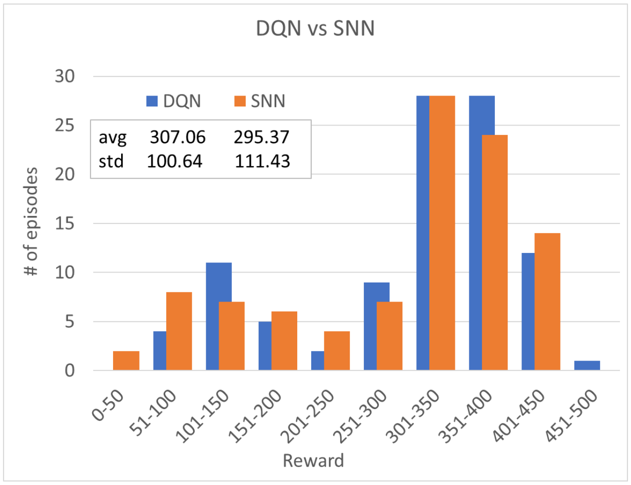
(a) Pixel-wise robustness ReLU-N vs SNN

Figure 6. Performance of ReLU-N and SNN for the robustness test. The x-axis represents the position of the bottom most occluded pixels of the 3-pixel thick horizontal occlusion bar. The y-axis represents the average reward. The standard distribution for the reward distribution is shown using the shaded region. The two critical areas are marked by the black bars **A** and **B** at the bottom of the plot. **A** shows the area near the paddle, while **B** marks the region of the screen occupied by the brick wall.

Another sensitive area for the ReLU-N corresponds to the position of the brick wall, marked by B in figure 6. We see that occluding some of the positions in this area results in a sharp drop in performance for ReLU-N. This can be explained by the nature of the gradient descent updates. Since the score changes when the ball hits the bricks and the backpropagation loss calculated using the TD-error is highest when the score changes, the filters of the network learn to discriminate these areas. Thus, when these areas are occluded, the performance drops. Interestingly, these sudden drops in performance are not observed in the SNN. This suggests that the SNN is more robust to occlusions in the input than the ReLU-N it is converted from. We also see that the SNN performs better than the ReLU-N in most of the experiments and has a lower standard deviation in the reward distribution.

For detailed list of results for positions of the occlusion see supplementary materials, Table 3.

6. Deep Q-networks



(a) DQN vs SNN

Figure 7. Performance of Deep Q-network vs. Deep Spiking Network. Each plot shows the reward distribution over 100 episodes using 0.05 epsilon greedy policy.

To demonstrate that our approach is applicable for state-of-the-art, large-scale networks, we trained the Deep Q-network (Mnih et al., 2015) and converted the weights to SNN with similar network architecture (see Figure 3). Since converting the DQN to SNN requires parameter search for a larger number of parameters, we used the parameter normalization method (Rueckauer et al., 2017). This approach shows reasonable performance, although its performance can be clearly improved using a systematic parameter optimization method. The deep Q-SNN was tested using the subtractive-IF neurons. We used the OpenAI baseline implementation of DQN to train the network (Dhariwal et al., 2017). We show that the DQN can be converted to spiking Q-network without significant loss in performance; see Figure 7 for full distribution of rewards using the two networks. At the present stage of the work, we did not conduct robustness test for the trained networks. We leave a systematic robustness study and comparison to future work.

7. Conclusion

In this paper, we demonstrate that ReLU-Ns trained on the game breakout can be converted to SNNs without degradation of performance. Moreover, we show that SNNs are more robust to occlusion attack and can outperform traditional ReLU networks on reinforcement learning tasks. In some cases, SNNs perform better than ReLU-N on previously unseen states. These results, combined with other benefits of SNNs, such as energy efficiency on neuromorphic hardware, make SNNs an ideal framework for reinforcement learning tasks when resources are limited and the environment is noisy.

In summary:

1. SNNs can perform reinforcement learning tasks like playing Atari games.
2. SNNs can be trained on reinforcement learning tasks by conversion from trained ReLU-Ns.
3. SNNs can outperform the ReLU-Ns from which they have been converted on reinforcement learning tasks, like playing Atari games.
4. SNNs are robust to attacks and perturbations in the input. They have improved generalization on states, which they have not encountered before.

References

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, jun 2013.

Bengio, Y., Fischer, A., Mesnard, T., Zhang, S., and Wu, Y. From stdp towards biologically plausible deep learning. In *Deep Learning Workshop, International Conference on Machine Learning (ICML)*, 2015.

Cao, Y., Chen, Y., and Khosla, D. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113 (1):54–66, May 2015. ISSN 1573-1405. doi: 10.1007/s11263-014-0788-3. URL https://doi.org/10.1007/s11263-014-0788-3.

Cassidy, A. S., Merolla, P., Arthur, J. V., Esser, S. K., Jackson, B., Alvarez-Icaza, R., Datta, P., Sawada, J., Wong, T. M., Feldman, V., Amir, A., Rubin, D. B., Akopyan, F., McQuinn, E., Risk, W. P., and Modha, D. S. Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–10, Aug 2013. doi: 10.1109/IJCNN.2013.6707077.

Clerc, M. Standard Particle Swarm Optimisation. 15 pages, September 2012. URL https://hal.archives-ouvertes.fr/hal-00764996.

Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., and Bengio, Y. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., Wu, Y., and Zhokhov, P. Openai baselines. https://github.com/openai/baselines, 2017.

Diehl, P. and Cook, M. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in Computational Neuroscience*, 9:99, 2015. ISSN 1662-5188. doi: 10.3389/fncom.2015.00099. URL `https://www.frontiersin.org/article/10.3389/fncom.2015.00099`.

Diehl, P. U., Neil, D., Binas, J., Cook, M., Liu, S., and Pfeiffer, M. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, July 2015. doi: 10.1109/IJCNN.2015.7280696.

Diehl, P. U., Pedroni, B. U., Cassidy, A., Merolla, P., Neftci, E., and Zarrella, G. Truehappiness: Neuromorphic emotion recognition on truenorth. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 4278–4285, July 2016. doi: 10.1109/IJCNN.2016.7727758.

Ferr, P., Mamalet, F., and Thorpe, S. J. Unsupervised feature learning with winner-takes-all based stdp. *Frontiers in Computational Neuroscience*, 12:24, 2018. ISSN 1662-5188. doi: 10.3389/fncom.2018.00024. URL `https://www.frontiersin.org/article/10.3389/fncom.2018.00024`.

Gilra, A. and Gerstner, W. Non-linear motor control by local learning in spiking neural networks. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1773–1782, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL `http://proceedings.mlr.press/v80/gilra18a.html`.

Hasselt, H. v., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pp. 2094–2100. AAAI Press, 2016. URL `http://dl.acm.org/citation.cfm?id=3016100.3016191`.

Hazan, H. and Manevitz, L. M. Topological constraints and robustness in liquid state machines. *Expert Systems with Applications*, 39(2):1597 – 1606, 2012. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2011.06.052. URL `http://www.sciencedirect.com/science/article/pii/S0957417411009523`.

Hazan, H., Saunders, D. J., Khan, H., Patel, D., Sanghavi, D. T., Siegelmann, H. T., and Kozma, R. Bindsnet: A machine learning-oriented spiking neural networks library in python. *Frontiers in Neuroinformatics*, 12:89, 2018. ISSN 1662-5196. doi: 10.3389/fninf.2018.00089. URL `https://www.frontiersin.org/article/10.3389/fninf.2018.00089`.

Huang, S. H., Papernot, N., Goodfellow, I. J., Duan, Y., and Abbeel, P. Adversarial attacks on neural network policies. *CoRR*, abs/1702.02284, 2017.

Huh, D. and Sejnowski, T. J. Gradient descent for spiking neural networks. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 1440–1450. Curran Associates, Inc., 2018.

Maass, W. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10:1659–1671, 1996.

Martí, D., Rigotti, M., Seok, M., and Fusi, S. Energy-efficient neuromorphic classifiers. *Neural Computation*, 28:2011–2044, 2016.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518 (7540):529–533, February 2015. ISSN 00280836. URL `http://dx.doi.org/10.1038/nature14236`.

Pfeiffer, M. and Pfeil, T. Deep learning with spiking neurons: Opportunities and challenges. *Frontiers in Neuroscience*, 12:774, 2018. ISSN 1662-453X. doi: 10.3389/fnins.2018.00774. URL `https://www.frontiersin.org/article/10.3389/fnins.2018.00774`.

Prez-Carrasco, J. A., Zhao, B., Serrano, C., Acha, B., Serrano-Gotarredona, T., Chen, S., and Linares-Barranco, B. Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing–application to feedforward convnets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2706–2719, Nov 2013. ISSN 0162-8828. doi: 10.1109/TPAMI.2013.71.

Richardson, M. J. E. and Gerstner, W. Statistics of subthreshold neuronal voltage fluctuations due to conductance-based synaptic shot noise. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 16(2):026106, 2006. doi: 10.1063/1.2203409. URL `https://doi.org/10.1063/1.2203409`.

Rueckauer, B., Lungu, I.-A., Hu, Y., and Pfeiffer, M. Theory and tools for the conversion of analog to spiking convolutional neural networks. *arXiv preprint arXiv:1612.04052*, 2016.

Rueckauer, B., Lungu, I.-A., Hu, Y., Pfeiffer, M., and Liu, S.-C. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in Neuroscience*, 11:682, 2017. ISSN 1662-453X. doi: 10.3389/fnins.2017.00682. URL https://www.frontiersin.org/article/10.3389/fnins.2017.00682.

Schultz, W. Dopamine reward prediction-error signalling: a two-component response. *Nature Reviews Neuroscience*, 17:183–195, 2016.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Stein, R. B., Gossen, E. R., and Jones, K. E. Neuronal variability: noise or part of the signal? *Nature Reviews Neuroscience*, 6:389 EP –, May 2005. URL https://doi.org/10.1038/nrn1668. Review Article.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT Press, Cambridge, MA, USA, 2st edition, 2018. ISBN 9780262039246.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Tuckwell, H. C. *Introduction to Theoretical Neurobiology*, volume 2 of *Cambridge Studies in Mathematical Biology*. Cambridge University Press, 1988. doi: 10.1017/CBO9780511623202.

W. Gerstner, W. M. K. *Spiking Neuron Models. Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002.

Wang, J. X., Kurth-Nelson, Z., Kumaran, D., Tirumala, D., Soyer, H., Leibo, J. Z., Hassabis, D., and Botvinick, M. M. Prefrontal cortex as a meta-reinforcement learning system. *Nature Neuroscience*, 21:1–9, 2018.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. Dueling network architectures for deep reinforcement learning. In Balcan, M. F. and Weinberger, K. Q. (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1995–2003, New York, New York, USA, 20–22 Jun 2016. PMLR. URL http://proceedings.mlr.press/v48/wangf16.html.

Witty, S., Lee, J. K., Tosch, E., Atrey, A., Littman, M., and Jensen, D. Measuring and characterizing generalization in deep reinforcement learning. *arXiv preprint arXiv:1812.02868*, 2018.

Wu, Y., Deng, L., Li, G., Zhu, J., and Shi, L. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in Neuroscience*, 12:331, 2018. ISSN 1662-453X. doi: 10.3389/fnins.2018.00331. URL https://www.frontiersin.org/article/10.3389/fnins.2018.00331.