Deep Spiking Q Networks

Masterthesis - Midterm Presentation

Christoph Hahn

October 23, 2019

Outline

1. Theoretical Background
2. Motivation
3. Methods
4. State of the Art
5. Experiments and Preliminary Results
6. Outlook

Outline

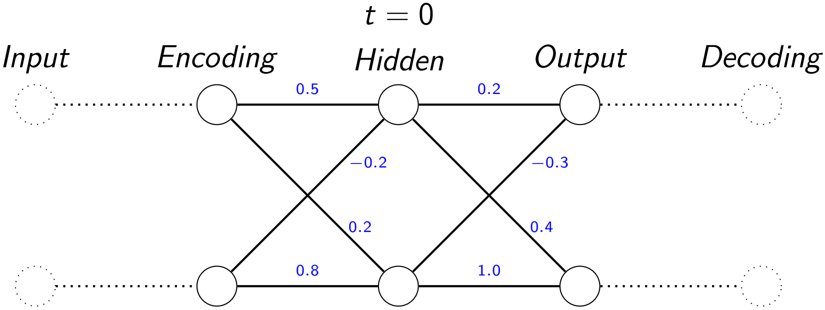
1. **Theoretical Background**
2. Motivation
3. Methods
4. State of the Art
5. Experiments and Preliminary Results
6. Outlook

Reinforcement Learning

- An **agent** learns by moving through an **environment** and receiving **rewards**.
- **Deep Q-Learning**[1]: A neural network (**DQN**) is trained which approximates the **value function** $Q(s, a)$.
- **Policy Gradient**[2]: An agent learns to estimate (with a NN) a **stochastic policy directly** by using gradient ascent.
- More advanced RL algorithms: **Actor-Critic** methods

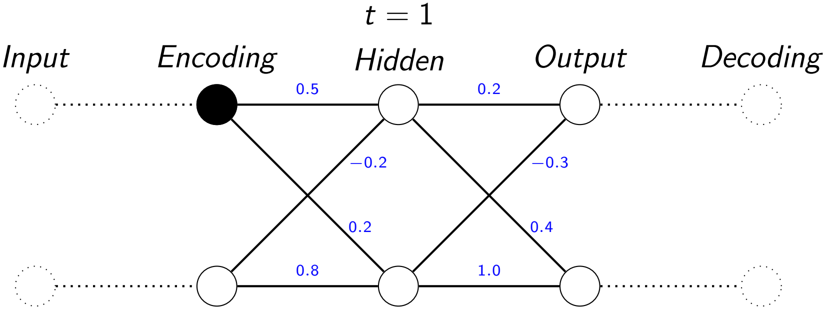
Spiking Neural Networks

- Inspired by neurons in the brain
- **Binary Outputs** (spike or no spike)
- Simulation over a **time period** (discrete or continuous)



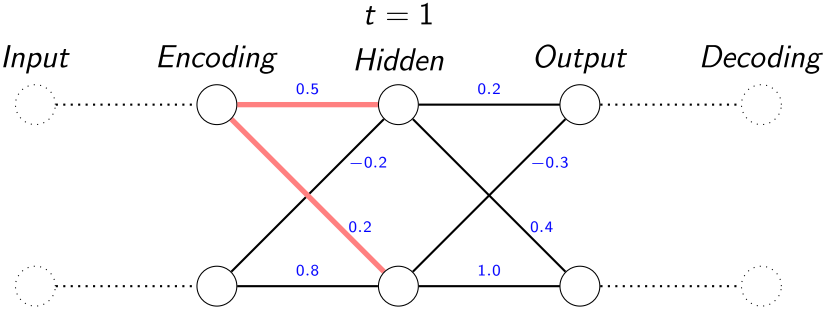
Spiking Neural Networks

- Inspired by neurons in the brain
- **Binary Outputs** (spike or no spike)
- Simulation over a **time period** (discrete or continuous)



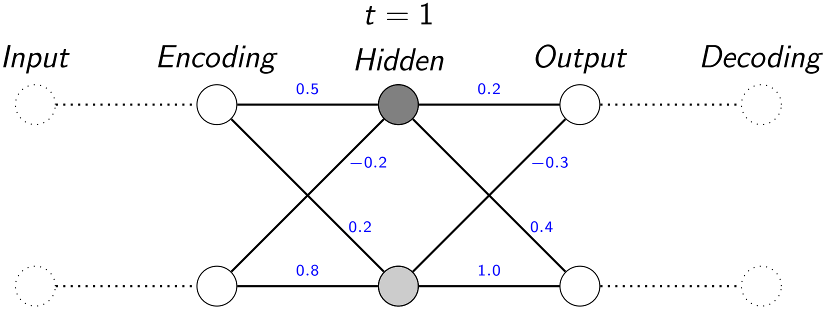
Spiking Neural Networks

- Inspired by neurons in the brain
- **Binary Outputs** (spike or no spike)
- Simulation over a **time period** (discrete or continuous)



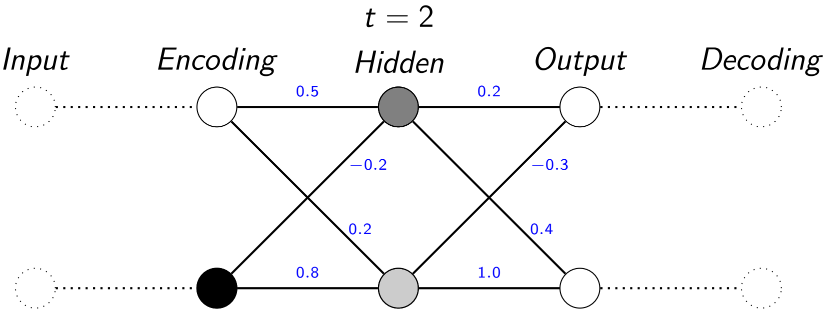
Spiking Neural Networks

- Inspired by neurons in the brain
- **Binary Outputs** (spike or no spike)
- Simulation over a **time period** (discrete or continuous)



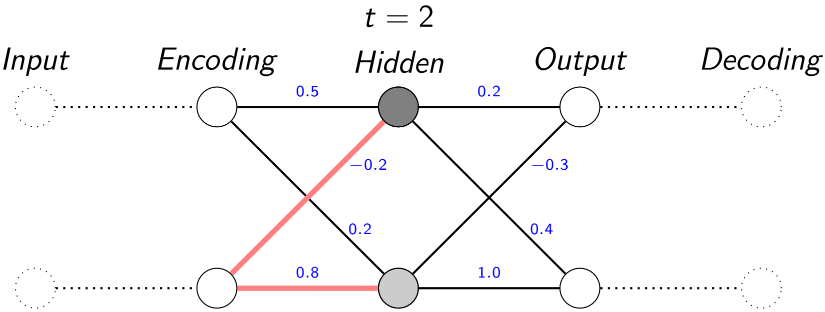
Spiking Neural Networks

- Inspired by neurons in the brain
- **Binary Outputs** (spike or no spike)
- Simulation over a **time period** (discrete or continuous)



Spiking Neural Networks

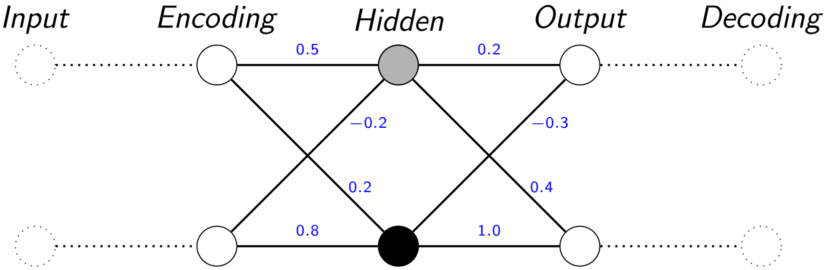
- Inspired by neurons in the brain
- **Binary Outputs** (spike or no spike)
- Simulation over a **time period** (discrete or continuous)



Spiking Neural Networks

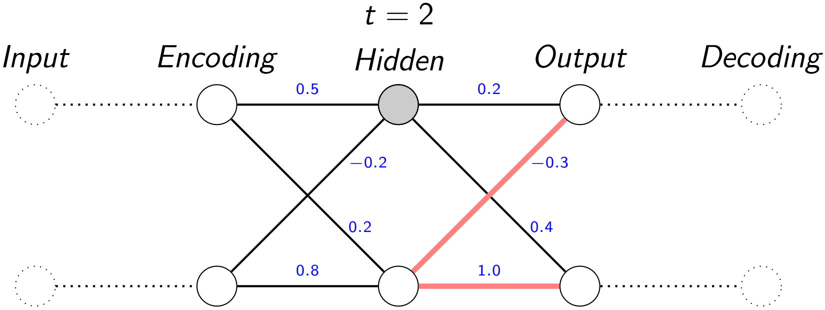
- Inspired by neurons in the brain
- **Binary Outputs** (spike or no spike)
- Simulation over a **time period** (discrete or continuous)

$t = 2$



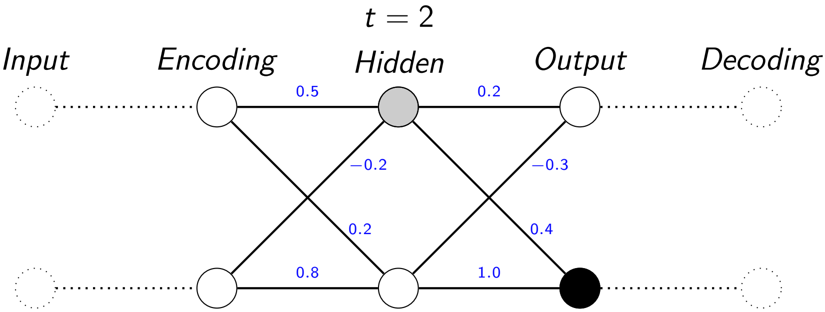
Spiking Neural Networks

- Inspired by neurons in the brain
- **Binary Outputs** (spike or no spike)
- Simulation over a **time period** (discrete or continuous)



Spiking Neural Networks

- Inspired by neurons in the brain
- **Binary Outputs** (spike or no spike)
- Simulation over a **time period** (discrete or continuous)



Neuron models

- Differ in complexity and biological plausibility
- **Leaky Integrate-and-Fire neurons**[3]
- (Non-Leaky) **Integrate-and-Fire** neurons
- Different reset mechanisms: Reset-by-Subtraction, Reset-to-Zero[4]

Outline

1. Theoretical Background
2. **Motivation**
3. Methods
4. State of the Art
5. Experiments and Preliminary Results
6. Outlook

Motivation

- **Energy-Efficiency** and **fast inference** on dedicated hardware (**neuromorphic hardware**)[5]
- **Brain-like** computation
- ? Exploit inherent **temporal** mechanics
- ? Naturally suited for **event-based inputs**

Research Goal

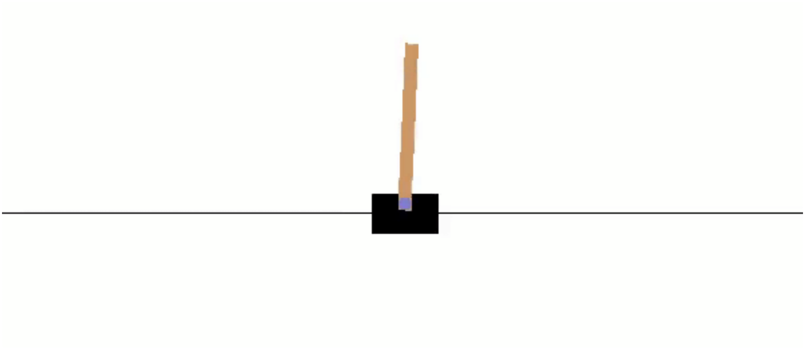
Research goal: **Explore** different methods to obtain a **spiking neural network** that can solve complex **reinforcement learning** tasks and **compare** these methods.

Outline

1. Theoretical Background
2. Motivation
3. **Methods**
4. State of the Art
5. Experiments and Preliminary Results
6. Outlook

Open AI Gym[6]

- Framework for reinforcement learning algorithms
- Defines goal for "solving" an environment
- **Comparison metrics**: Environment solved?, Best 100 episode average, training time, convergence



Training Methods: Overview

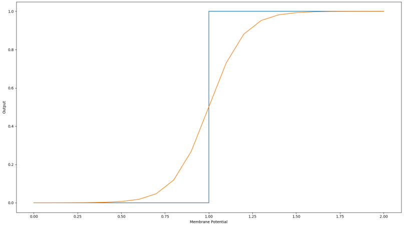
- **Conversion**
- **Surrogate Gradients**
- Local Learning
  - Spike Timing Dependent Plasticity (**STDP**)
  - event-based Random Backpropagation (**eRBP**)

Training Methods: Conversion[7] [4]

- Train a **conventional Neural Network** with certain constraints: ReLu activation in hidden layers
- **Rate-based Encoding**
- **Weight conversion** based on data
- **Rate-based Decoding**

Training Methods: Surrogate Gradients

- SNN is trained on conventional hardware using **Backpropagation**
- **Gradients** of binary function are **replaced** with a steep sigmoid function[8]
- **SpyTorch** Framework[3]



Training Methods: STDP

- **Unsupervised** local learning
- Based on relative spike timings[9]
- Reward-modulated STDP (**R-STDP**): An external reward stimulates the neurons to perform STDP or anti-STDP[10]

Training Methods: STDP

- **Unsupervised** local learning
- Based on relative spike timings[9]
- Reward-modulated STDP (**R-STDP**): An external reward stimulates the neurons to perform STDP or anti-STDP[10]
- **Problem** with STDP: Unclear how to propagate errors backward => Only output layer can be trained with R-STDP, **hidden layers have to be trained unsupervised**

Training Methods: eRBP[11]

- Ideas of eRBP:
  - Use **random feedback weights** instead of symmetric weights
  - **Error neurons** that accumulate the error and emit a **spike** backward once they cross their threshold
  - **Two compartment neurons**, one for the forward pass, one for the backward pass that trigger a weight update when their threshold is crossed

Training Methods: eRBP[11]

- Ideas of eRBP:
  - Use **random feedback weights** instead of symmetric weights
  - **Error neurons** that accumulate the error and emit a **spike** backward once they cross their threshold
  - **Two compartment neurons**, one for the forward pass, one for the backward pass that trigger a weight update when their threshold is crossed
- **Problem** of eRBP in Q-Learning: Approach needs to be adapted that it **remains local**

Outline

1. Theoretical Background
2. Motivation
3. Methods
4. **State of the Art**
5. Experiments and Preliminary Results
6. Outlook

State of the Art

- Conversion
  - Patel et al.[12] **convert a DQN** for BreakOut and report better results after conversion and more robustness

State of the Art

- Conversion
  - Patel et al.[12] **convert a DQN** for BreakOut and report better results after conversion and more robustness
- All methods
  - Good performance on **classification** tasks
  - Few (conversion) to no (all others) papers on deep reinforcement learning

Outline

1. Theoretical Background
2. Motivation
3. Methods
4. State of the Art
5. **Experiments and Preliminary Results**
6. Outlook

Experiments: Direct Conversion DQN

- DQN on CartPole: Average performance 224.11

Simulation Time	**Conversion Accuracy Original DQN**	**Average Performance after Conversion over 100 episodes**
100	64.36%	133.45
1000	84.97%	107.68
10000	94.89%	151.77

Experiments: Indirect Conversion DQN

- Train a classifier to predict the action the DQN would take in state s[13]
- Average performance of the classifier 229.06

Simulation Time	Conversion Accuracy Classifier	Average Performance after Conversion over 100 episodes
100	95.37%	197.34
1000	99.50%	220.82
10000	99.9%	takes too much time

Experiments: Conversion in NEST

- Challenge: **different Neuron models** in NEST compared to the conversion papers
- iaf_psc_delta neurons with reset-to-zero mechanism
  - **classifier** converts with $\approx$ **97%** accuracy (1000 time steps)
  - **direct conversion** does not work ($\approx$**57%**) accuracy for 10000 time steps
  - **Policy Gradient** converts with $\approx$ **98%** (100 timesteps)

Experiments: Conversion in NEST

- Challenge: **different Neuron models** in NEST compared to the conversion papers
- iaf_psc_delta neurons with reset-to-zero mechanism
  - **classifier** converts with $\approx$ **97%** accuracy (1000 time steps)
  - **direct conversion** does not work ($\approx$**57%**) accuracy for 10000 time steps
  - **Policy Gradient** converts with $\approx$ **98%** (100 timesteps)
- pp_psc_delta neurons with adaptive threshold and stochastic firing
  - **classifier** converts with $\approx$**96%** accuracy (1000 time steps)
  - **direct conversion** with $\approx$ **93%** accuracy (10000 time steps)

Experiments: Surrogate Gradients DQN

- Setup: Same hyperparameters for Q-Learning, Non-Leaky Integrate-and-Fire neurons, 20 simulation time steps
- Experiment: Run DQN and DSQN for **1000 episodes 5 times each**
- Result:
  - DQN: Reached gym standard twice, best 100 episode **average** $>$ **190 for all runs**
  - DSQN: Reached gym standard once, two runs with average $>$ 190, **lowest best average 170**

Preliminary Results

- **Conversion** accuracy is **dependent** on the relative difference between the outputs

=> Conversion works better with a softmax **output function** (e.g. Classifier or Policy Gradient)

=> Conversion accuracy for DQNs depends on the specific agent and **environment**

Preliminary Results

- **Conversion** accuracy is **dependent** on the relative difference between the outputs

=> Conversion works better with a softmax **output function** (e.g. Classifier or Policy Gradient)

=> Conversion accuracy for DQNs depends on the specific agent and **environment**

- Training with **surrogate gradients works** for DSQNs, although **worse than** training a standard **DQN**

Preliminary Results

- **Conversion** accuracy is **dependent** on the relative difference between the outputs

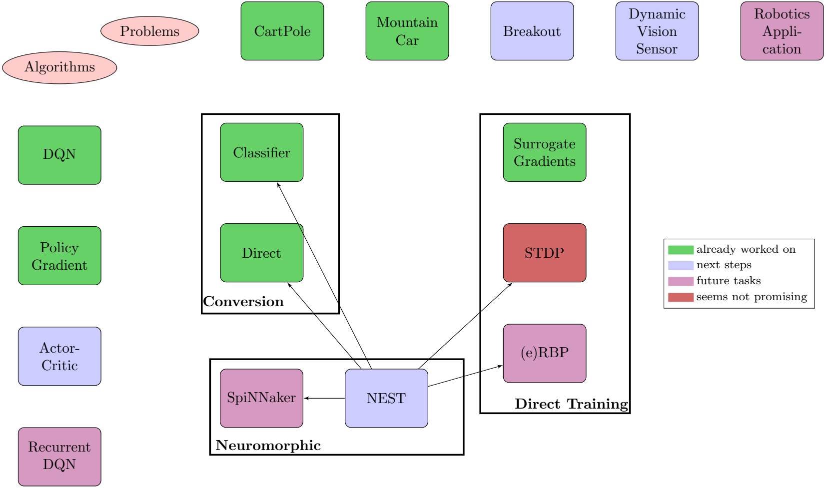
=> Conversion works better with a softmax **output function** (e.g. Classifier or Policy Gradient)

=> Conversion accuracy for DQNs depends on the specific agent and **environment**

- Training with **surrogate gradients works** for DSQNs, although **worse than** training a standard **DQN**
- Direct Training with **STDP** of a deep reinforcement learning algorithm is **difficult**, because only the last layer can be trained with R-STDP and therefore relies on meaningful features calculated in unsupervised STDP layers

Outline

1. Theoretical Background
2. Motivation
3. Methods
4. State of the Art
5. Experiments and Preliminary Results
6. **Outlook**


Problems
Algorithms
CartPole
Mountain Car
Breakout
Dynamic Vision Sensor
Robotics Appli-cation
DQN
Policy Gradient
Actor-Critic
Recurrent DQN
Classifier
Direct
Conversion
Surrogate Gradients
STDP
(e)RBP
Direct Training
SpiNNaker
NEST
Neuromorphic
already worked on
next steps
future tasks
seems not promising

Discussion

- Feedback (Presentation, Experiments, Results, ...)
- ? Which direction to go into
- ? How to compare reinforcement learning algorithms

Backup

Reinforcement Learning

- An **agent** learns by moving through an **environment** and receiving **rewards**.
- **Deep Q-Learning**[1]: A neural network (**DQN**) is trained which approximates the **value function** $Q(s, a)$.
- Loss function for DQN:

$$L = ((r_t + \gamma max_a Q(s_{t+1}, a)) - Q_{old}(s_t, a_t))^2$$

Reinforcement Learning

- **Policy Gradient**[2]: An agent learns to estimate (with a NN) a **stochastic policy directly** by using gradient ascent.
- More advanced RL algorithms: **Actor-Critic** methods

Neuron models

- Differ in complexity and biological plausibility
- **Leaky Integrate-and-Fire neurons**[3]

$$I_i[n+1] = \alpha I_i[n] + \sum_j W_{ij}[n] S_j[n]$$
$$U_i[n+1] = \beta U_i[n] + I_i[n] - S_i[n]$$

- (Non-Leaky) **Integrate-and-Fire** neurons: $\alpha = 0, \beta = 1$
- Reset-by-Subtraction, Reset-to-Zero[4]

Input Encoding/Output Decoding

- **Rate-based** methods
  - Encoding: Poisson spike trains[14], Equidistant spikes, Constant Input Currents[4]
  - Decoding: Number of spikes, Potential of output neurons[4]

Input Encoding/Output Decoding

- **Rate-based** methods
  - Encoding: Poisson spike trains [14], Equidistant spikes, Constant Input Currents [4]
  - Decoding: Number of spikes, Potential of output neurons [4]
  - + Straightforward, **easy to apply** for conversion methods [5]
  - – **No temporal information processing**, lots of spikes needed for good precision [5]

Input Encoding/Output Decoding

- **Temporal** methods
  - Encoding: Rank-based[15], Latency[10]
  - Decoding: Time to first spike[10]

Input Encoding/Output Decoding

- **Temporal** methods
  - Encoding: Rank-based[15], Latency[10]
  - Decoding: Time to first spike[10]
  - \+ Sparse spikes => **Energy efficiency and fast inference**[5]
  - − **Difficult to apply**

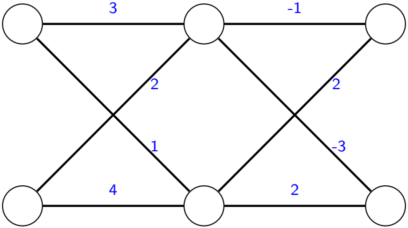
Training Methods: Overview

- Conversion methods
  - Generic conversion
  - Training of a constraint network
- Backpropagation based methods
  - Surrogate Gradients
  - Shadow network for training
  - Binary ANNs
- Local Learning
  - Spike Timing Dependent Plasticity (STDP)
  - event-based Random Backpropagation (eRBP)

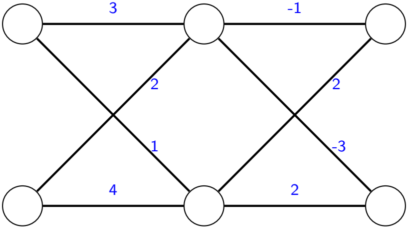
Training Methods: Conversion[7] [4]

- Train a **conventional Neural Network** with certain constraints: ReLu activation in hidden layers
- **Encoding** methods: Poisson, Equidistant Spikes, Constant Input current
- **Weight conversion**/normalization: Data-based, Model-based, Robust, Parameter search
- Output **Decoding**: Function of Potentials or Spikes

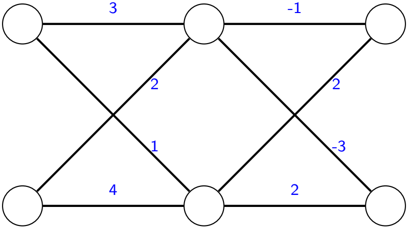
Conversion Example



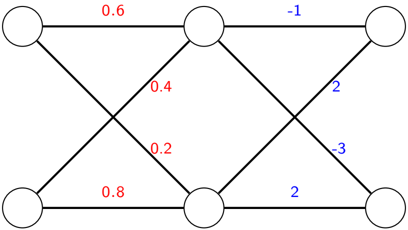
Data-based normalization



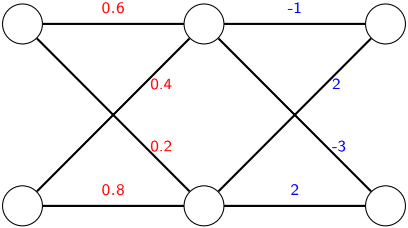
Layer 1: Max-activation is 5 => Rescale by 0.2



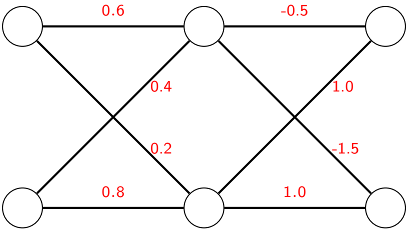
Layer 1: Max-activation is 5 => Rescale by 0.2



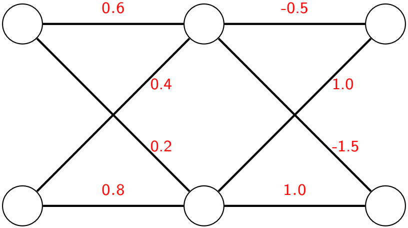
Layer 2: Max activation is 2 => Rescale by 0.5



Layer 2: Max activation is 2 => Rescale by 0.5



Problem: Underactivation



Training Methods: STDP

- **Unsupervised** local learning
- Connections where a pre-synaptic spike occurs just before a post-synaptic spike are strengthened
- Connections where a post-synaptic spike occurs just before a pre-synaptic spike are weakened[9]
- Reward-modulated STDP (**R-STDP**): An external reward stimulates the neurons to perform STDP or anti-STDP[10]
- **Problem** with STDP: Unclear how to propagate errors backward => Only output layer can be trained with R-STDP, **hidden layers have to be trained unsupervised**

Training Methods: eRBP[11]

- Problems of Conventional Backpropagation:
  - Keep symmetric forward and backward weights (**weight transport problem**)
  - backpropagate error signals
- Ideas of eRBP:
  - use **random feedback weights** instead of symmetric weights
  - **error neurons** that accumulate the error and emit a **spike** backward once they cross their threshold
  - **two compartment neurons**, one for the forward pass, one for the backward pass that trigger a weight update when their threshold is crossed
- **Problem** of eRBP in Q-Learning: Approach needs to be adapted that it **remains local**

State of the Art

- Conversion
  - Patel et al.[12] **convert a DQN** for BreakOut and report better results after conversion and more robustness
  - Many papers (e.g.[7,4])that **convert classification networks** with high accuracy
  - Converted SNNs perform best in terms of accuracy compared to other training methods[5]
- Surrogate Gradients
  - State-of-the-art results for **classification tasks** such as MNIST[5]

State of the Art

- STDP
  - **Deep classification networks**, where the hidden layers are trained unsupervised and the output layer is trained with R-STDP[16] [17]
  - Competitive performance in **unsupervised** networks[5]
- (e)RBP
  - Performance of RBP almost as good as of standard BP on **classification** tasks[18]
  - State-of-the-art performance for classification with eRBP on **DVS**-**gesture** data set[19]

Experiments: En-/Decoding Methods DQN

- A classifier with normalized inputs (between 0 and 1) is converted and simulated for 100 time steps per episode
- Conversion Accuracy for 1000 time steps is reported

Encoding Method \ Decoding Method	Spikes	Potential
Poisson spike train	74.1%	74.1%
Equidistant Spikes	77.6%	82.2%
Constant input currents	93.7%	92.4%

Open research questions

- **Comparison** of **conversion methods** for DQNs
- Running a DSQN on **neuromorphic hardware**
- **Influence of the environment** on DQN conversion
- Conversion and Direct Training of **more advanced Reinforcement Learning algorithms**
- Adapt **eRBP for Q-Learning**
- **Exploit temporal capabilities** of SNN for reinforcement learning tasks
- **Compare conversion and direct training** methods for reinforcement learning

Sources

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, and Shane Legg & Demis Hassabis. Human-level control through deep reinforcement learning. *Nature, Volume 518*, 26 February 2015.

[2] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, (April 2015):1057–1063, 2000. ISSN 10495258.

[3] Emre O. Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks. *IEEE SPM WHITE PAPER FOR THE SPECIAL ISSUE ON NEUROMORPHIC COMPUTING*, 28 January 2019.

[4] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, and Michael Pfeiffer. Theory and tools for the conversion of analog to spiking convolutional neural networks. December 2016.

[5] Michael Pfeiffer and Thomas Pfeil. Deep learning with spiking neurons: Opportunities and challenges. *Frontiers in NeuroScience*, 25 October 2018.

[6] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John

Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016. URL http://arxiv.org/abs/1606.01540.

[7] Peter U. Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih chii Liu, and Michael Pfeiffer. Fastclassifying, high-accuracy spiking deep networks through weight and threshold balancing. In *inNeural Networks (IJCNN), 2015 International Joint Conference on (Killarney: IEEE)*, 2015.

[8] Friedemann Zenke and Surya Ganguli. Superspike: Supervised learning in multi-layer spiking neural networks. October 14, 2017.

[9] Henry Markram, Joachim Lübke, Michael Frotscher, and Bert Sakmann. Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. *Science*, 275(5297):213–215, 1997. ISSN 00368075. doi: 10.1126/science.275.5297.213.

[10] Milad Mozafari, Mohammad Ganjtabesh, Abbas Nowzari-Dalini, and Timothée Masquelier. Spyketorch: Efficient simulation of convolutional spiking neural networks with at most one spike per neuron. 6 March 2019.

[11] Emre Neftci, Charles Augustine, Somnath Paul, and Georgios Detorakis. Neuromorphic Deep Learning Machines. pages 1–25, 2016. doi: 10.3389/fnins.2017.00324. URL http://arxiv.org/abs/1612.05596{%}0Ahttp://dx.doi.org/10.3389/fnins.2017.00324.

[12] Devdhar Patel, Hananel Hazan, Daniel J. Saunders, Hava T. Siegelmann, and Robert Kozma. Improved robustness of reinforcement learning

policies upon conversion to spiking neuronal network platforms applied to Atari Breakout game. *Neural Networks*, (xxxx), 2019. ISSN 08936080. doi: 10.1016/j.neunet.2019.08.009. URL https://doi.org/10.1016/j.neunet.2019.08.009.

[13] Claus Meschede. Training neural networks for event-based end-to-end robot control. *Master thesis at the Technical University of Munich*, 28 July 2017.

[14] Professor David Heeger. Poisson model of spike generation. September 5, 2000.

[15] Simon Jonathan Thorpe. Rate coding versus temporal order coding: What the retinal ganglion cells tell the visual cortex. *Neural Computation*, July 2001.

[16] Saeed Reza Kheradpisheh, Mohammad Ganjtabesh, Simon J. Thorpe, and Timothée Masquelier. STDP-based spiking deep convolutional neural networks for object recognition. *Neural Networks*, 99:56–67, 2018. ISSN 18792782. doi: 10.1016/j.neunet.2017.12.005.

[17] Milad Mozafari, Mohammad Ganjtabesh, Abbas Nowzari-Dalini, Simon J. Thorpe, and Timothée Masquelier. Bio-inspired digit recognition using reward-modulated spike-timing-dependent plasticity in deep convolutional networks. *Pattern Recognition*, 94:87–95, 2019. ISSN 00313203. doi: 10.1016/j.patcog.2019.05.015.

[18] Pierre Baldi, Peter Sadowski, and Zhiqin Lu. Learning in the machine: Random backpropagation and the deep learning channel. *Artificial Intelligence*, 260:1–35, 2018. ISSN 00043702. doi: 10.1016/j.artint.2018.03.003.

[19] Jacques Kaiser, Alexander Friedrich, J. Camilo Vasquez Tieck, Daniel Reichard, Arne Roennau, Emre Neftci, and Rüdiger Dillmann. Embodied Neuromorphic Vision with Event-Driven Random Backpropagation. pages 1–8, 2019. URL http://arxiv.org/abs/1904.04805.