Acknowledgement

I would like to thank Prof. Harold Soh for his tireless guidance, and my peers who have given me feedback over the course of the project. This work was partially supported by the Science and Engineering Research Council, Agency of Science, Technology and Research, Singapore, through the National Robotics Program under Grant No.172 25 00063.

B.Comp. Dissertation

Event-Driven Visual-Tactile Learning for Robots

By

Kuan Sheng Yuan, Jethro

Department of Computer Science

School of Computing

National University of Singapore

2019/20

B.Comp. Dissertation

Event-Driven Visual-Tactile Learning for Robots

By

Kuan Sheng Yuan, Jethro

Department of Computer Science

School of Computing

National University of Singapore

2019/20

Project No: H226080
Advisor: Dr. Harold Soh
Deliverables:
Report: 1 Volume

Abstract

In this work, we contribute an event-driven visual-tactile perception system, built on spiking neural networks. Our perception system is trained and tested end-to-end on novel multi-modal event-based data for two sensor modalities: (1) Tactile data, from our biologically-inspired fingertip tactile sensor, NeuTouch and (2) visual data, from the Prophesee camera. We evaluate our system on two robotic tasks: container classification, and rotational slip detection. On both tasks, we observe good classification accuracies relative to standard deep learning methods. Our system can be run on neuromorphic hardware, making this a crucial first step towards enabling power-efficient, intelligent robots.

Keywords:
Spiking Neural Networks, Multi-modal Machine Learning

Implementation Software and Hardware:
Python 3, PyTorch

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Human beings are blessed with innate abilities to integrate sensory information from different stimuli. Our sense of smell, hearing, touch, vision, and taste each contribute to how we perceive and act in the world. Consider the scenario of fetching a carton of soy-milk from the fridge; humans use vision to locate the carton, and are also able to infer from grasping the object, the amount of soy-milk the carton contains. These actions (and inferences) are performed robustly using a power-efficient neural substrate — compared to popular deep learning approaches for using multiple sensor modalities in artificial systems, human brains require far less energy [1], by communicating with sparse, discrete spikes.

In this work, we take crucial steps towards efficient visual-tactile perception for robotic systems. We take full advantage of this sensor data using spiking neural networks. These neural networks use neuronal units that mimic the biological neuron, and possess desirable characteristics such as low latencies, and the ability to handle event-based data at high temporal resolution. Traditional neural networks deployed on currently commonplace hardware like GPUs face the Von Neumann bottleneck — the throughput of the computer system is limited by the memory model.

We gain inspiration from biological systems, which are *asynchronous* and *event-driven*. Consider using a video camera to capture the flight of a bird for further analyses. The camera operates at a fixed frame rate (say, 30 frames per second), and will fail to capture crucial information about its wing movement. In addition, the video camera captures unimportant information such

as the skyline. Event-based sensors such as the camera sensors overcome this limitation by instead capturing changes in the scene — events. In contrast to resource-hungry deep learning methods, event-driven perception offers an alternative approach that promises power-efficiency and low-latency — features that are ideal for real-time mobile robots. However, event-driven systems remain under-developed relative to standard synchronous perception methods [2].

We make multiple contributions that advance event-driven visual-tactile perception. First, to enable richer tactile sensing, we use the NeuTouch fingertip sensor. Compared to existing commercially-available tactile sensors, NeuTouch's neuromorphic design enables scaling to larger number of taxels while retaining low latencies.

Next, we investigate multi-modal (visual-tactile) learning using the NeuTouch and the Prophesee event-based camera. Specifically, we develop a *visual-tactile spiking neural network* (VT-SNN) that incorporates both sensory modalities for supervised-learning tasks. Different from conventional deep artificial neural network (ANN) models, SNNs process discrete spikes asynchronously, and thus, are arguably better suited to the event data generated by our neuromorphic sensors. In addition, SNNs can be used on efficient low-power neuromorphic chips such as the Intel Loihi [3].

Our experiments center on two robot tasks: object classification and (rotational) slip detection. In the former, we tasked the robot to determine the type of container being handled and amount of liquid held within. The containers were opaque with differing stiffness, and hence, both visual and tactile sensing are relevant for accurate classification. We show that relatively small differences in weight (≈ 30g across 20 object-weight classes) can be distinguished by our prototype sensors and spiking models. Likewise, the slip detection experiment indicates rotational slip can be accurately detected within 0.08s (visual-tactile spikes processed every ≈ 1ms). In both experiments, SNNs achieved competitive (and sometimes superior) performance relative to ANNs with similar architecture.

The work presents an exciting opportunity to enable power-efficient intelligent robots. Presented with labeled data, an event-driven perception network can be trained end-to-end, and used on neuromorphic chips as robotic controllers.

1.1 Research Outline

In chapter 2, we first provide a comprehensive study on spiking neural networks. We discuss why and when we should use SNNs, and how to train them. We then discuss the prior art for visual-tactile systems and event-based sensors in chapter 3.

Next, we introduce our event-driven visual-tactile spiking-neural network — VT-SNN — which enables fast perception on two event-based sensors: the NeuTouch [4], and the Prophesee event camera chapter 5.

In chapter 4, we detail the robot hardware setup used across our experiments. Next, we discuss our experimental methods and results for two robot tasks: container classification (chapter 6), and slippage detection (chapter 7). We discuss our experimental results, and demonstrate that spiking neural networks achieve competitive (and sometimes superior) performance over deep learning methods, and in addition having the unique property of early classification. We then run the trained models on the Intel Loihi in chapter 8 to quantify the gains in power-efficiency and inference speeds. Finally, we conclude our work (chapter 9), and provide future research directions.

In the appendix, we include additional work, evaluating the feasibility of spiking neural networks as agents in a reinforcement learning setting Appendix B. This research direction was explored in the first semester, but later abandoned.

1.2 Individual Contributions

The event-driven visual-tactile perception system is a joint work between our group, the Collaborative Learning & Adaptive Robots (CLeAR) group, and the TEE research group. This work has been submitted to ROS 2020.

The TEE research group contributed the novel NeuTouch tactile sensor, which was used to collect data for the tactile modality. The CLeAR group contributed: (1) a visual-tactile spiking neural network (VT-SNN) that leverages multiple event sensor modalities, (2) systematic experiments demonstrating the effectiveness of our event-driven perception system on object classifi-

cation and slip detection, with comparisons to conventional ANN methods, and (3) visual-tactile event sensor datasets comprising more than 50 different object classes across the experiments, including RGB images and proprioceptive data from the robot.

Among the contributions of the CLeAR group, my individual contributions include the experimentation (training and evaluation) of the VT-SNN models, as well as some guidance on the ANN models. During the research process, I also maintained the code repository at high-quality. Model training was designed to be reproducible, and parameter searches could be done efficiently across multiple GPUs. Different experimental runs could also be plotted, and their results aggregated for later analyses.

The benchmarking work in chapter 8, and the work on spiking neural agents with reinforcement learning Appendix B are my own.

Chapter 2

Background

This chapter provides background knowledge about spiking neural networks. First, we review the differences between deep neural networks, and spiking neural networks (2.1). Next, we introduce the basics of spiking neural networks (2.3), and their benefits (2.2). Finally, we discuss how to train them (2.6).

2.1 The Generations of Neural Networks

Neural network models can be classified into three generations, according to their computational units: perceptrons, non-linear units, and spiking neurons [5]. Perceptrons can be composed to produce a variety of models, including Boltzmann machines and Hopfield networks. One characteristic feature of perceptrons is that only give digital output.

Non-linear units apply an activation function with a continuous set of possible output values to a weighted sum of the inputs. Classic examples of networks of this second generation include feed-forward and recurrent sigmoidal neural networks. These networks are able to compute functions with analog input and output. Non-linear units are currently the most widely used computational unit, and is responsible for the explosion of progress in machine learning research, in particular, the success of deep learning. This is largely because networks built with these units are trainable with well-researched gradient-based methods, such as backpropagation.

Name	Input	Output	Examples
(1) Perceptrons	Digital	Digital	Hopfield Networks, Boltzmann Machines
(2) Non-linear Units	Digital/Analog	Digital/Analog	Feed-forward & Recurrent ANNs
(3) Spiking Neurons	Analog	Analog	Feed-forward SNNs

Table 2.1: The three generations of neural networks.

Despite being biologically inspired, second-generation neural networks (ANNs) bear little resemblance to the human cortex. In contrast to the digital computation in ANNs, networks of neurons perform fast analog computations. This inspired a the third generation of neural networks use computational units called *spiking neurons*, which communicate via discrete spikes. Much like our biological neurons, spiking neurons are connected to each other at synapses, receiving incoming signals at the dendrites, and sending spikes to other downstream neurons via the axon. Each computational unit stores its membrane potential, which fluctuates over time based on well-defined neuronal dynamics. Rather than firing at each propagation cycle, these computational units fire only when their individual membrane potentials crosses its firing threshold.

Henceforth, we shall term second-generation neural networks artificial neural networks (ANNs), and third-generation neural networks spiking neural networks (SNNs).

2.2 Motivating Spiking Neural Networks

Since ANNs have excellent performance, and can handle both digital and analog input and output, why should we bother with spiking neural networks? In this section, we motivate SNNs from various perspectives.

2.2.1 Information Encoding via Spikes

To directly compare ANNs and SNNs, one can consider the real-valued outputs of ANNs to be the firing rate of a spiking neuron in steady state. In fact, this scheme, termed *rate coding*, has

been used to explain computational processes in the brain [2].

However, experiments have shown that different actions are taken based on single spikes [6]. In addition, the humans are capable of performing tasks such as visual pattern analysis and pattern classification in 100ms [7]. Rate coding strategies are far too inefficient for the rapid information transmission required for sensory processing. In addition, the firing distribution of biological neurons is heavily skewed towards lower firing rates. To obtain a good estimate of the firing rate, many spikes would be required.

Different from ANNs, spiking neuron models are able to encode information beyond the average firing rate: these models also utilize the relative timing between spikes [8], or spike phases (in-phase or out-of-phase). These time-dependent codes are termed *temporal codes*, and play an important role in biology. It has also been successfully demonstrated that temporal coding achieves competitive empirical performance on classification tasks for both generated datasets, as well as image datasets like MNIST and CIFAR [9].

There is additional benefit to encoding information directly in the temporal domain. Architectures for atemporal artificial neural networks that use a memory mechanism, such as the LSTM [10], require every neuron wait on the activation of all neurons in previous layers before producing an answer. In spiking neural networks, the output layers spike over time, and predictions can be made at any time-step. Hence, SNNs are able to operate in two regimes. The first regime is a highly accurate but slow regime, where predictions are made at later time-steps. This is the regime that atemporal ANNs support, where predictions are typically made only after a fixed time-step. The second regime is a low accuracy but fast regime, where the network fast predictions, at a much lower accuracy. This behaviour mirrors the speed-accuracy tradeoff observed in human decision-making [9], and we also observe and exploit this property in our work described in chapter 5.

By observing that biological neurons use spikes to communicate information, a wide range of coding schemes have been developed. Each of these codes have different information capacities, and their pros and cons. We defer this discussion to [7], but summarize the coding schemes in Table 2.2. Spiking neural networks are able to take advantage of these various

Coding Scheme	Encoding of Information
Rate Coding	Firing rate of neurons
Count Code	Count of number of neurons that spike during in time window
Binary Code	Binary pattern of length N from N neurons (e.g. 0010, where the third neuron spiked)
Temporal Code	Time of each spike in the spike train
Rank Code	Order in which neurons fire
Synchrony Code	Patterns from a population of neurons

Table 2.2: Coding schemes in Spiking Neural Networks

coding schemes.

2.2.2 Practical Benefits

The ability to communicate information in a small number of spikes has immense practical benefits.

Real-time Performance and Parallelism

The analog communication modeled in spiking neurons allow for naturally parallel and high-speed computation. The promise of real-time performance is a key motivator behind the development of neuromorphic hardware. Typical hardware such as the everyday computer uses the von Neumann architecture, which separates memory and processing. This model of operation has severe implications on the throughput of the computer system, and is commonly known as the von Neumann bottleneck [11]. Neuromorphic hardware collocates memory and processing, overcoming the von Neumann bottleneck.

Power Efficiency

Next, the primary motivation cited in present-day literature is the power-efficiency of spiking neural networks. First, less energy is used in propagating fewer spikes. Second, communication via spikes is much more computationally efficient than the floating point arithmetic performed

in artificial neural networks. This low energy footprint is a highly desirable trait in robotics applications.

Online Learning and Fault Tolerance

Biological neurons brain evolve and adapt as time passes, creating new connections, and destroying old ones. While the phenomenon of *neuroplasticity* is not yet well understood, it is hoped that the learning mechanisms behind human intelligence will inspire a new generation of online learning algorithms.

Neurons also frequently perish, and the human brain has the ability to self-heal. In the same way, spiking neural network architectures can be designed to possess the same self-healing and fault tolerance characteristics. This is a desirable trait for systems where uptime is critical.

2.2.3 Biological Plausibility

In addition to having practical benefits, spiking neural networks more closely model the biological brain. A faction of the machine learning and neurobiology community strives for emulation of the biological brain. There are several incompatibilities between ANNs and the current state of neurobiology that are difficult to reconcile.

First, neurons in ANNs communicate via continuous-valued activations. In contrast, it is well established that biological neurons communicate by broadcasting spike trains — trains of action potentials — to downstream neurons. The spikes are to a first-order approximation of uniform amplitude, unlike the continuous-valued activations of ANNs.

Second, backpropagation as a learning procedure also presents incompatibilities with the biological brain [12], explained in further detail in section 2.6. This result motivates the search for alternative learning mechanisms for spiking neural networks.

2.2.4 Summary

In conclusion, the main motivations for spiking neural networks are the following:

- Neuromorphic hardware can overcome the von Neumann bottleneck

- When combined with neuromorphic hardware, SNNs can:
  - achieve real-time performance via asynchronous spiking communication
  - achieve low power consumption from sparse spiking computation
  - implement plasticity-inspired online learning mechanisms
  - Fault tolerance beyond what von Neumann architectures can provide
- Some SNNs have biologically-plausible neuron models, which can help answer open questions in neuroscience

2.3 Neuron Models

Neuron models are mathematical models that describe the underlying neuron dynamics: how electrical signals propagate from inputs to outputs, and the changes in the neurons internal state, such as the membrane potential.

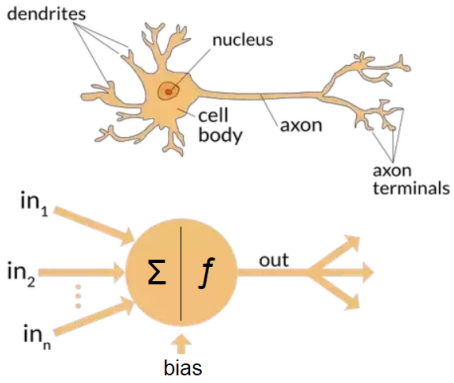
Many of these neuron models are modeled after the biological neuron, which has a few main components:

Dendrites Neurons receive input signals at the dendrites, which transmit incoming electrical signals to the cell body.

Cell Body The cell body is responsible for "processing" the input from the dendrites. The typical behaviour is to accumulate the electrical input from the dendrites. Once the cell's membrane potential crosses a certain threshold, it spikes, and sends an output signal through the axon.

Axons The axon receives electrical signals from the cell body. Electrical signals propagate down to axon terminals, which control the release of neurotransmitters.

The perceptron was biologically-inspired, and has an analogous representation that may be helpful for understanding, which we present in Figure 2.1.



Biological Neuron	Perceptron
Dendrites	Inputs
Cell Body	Processing input: $f(\sum \mathbf{w} \cdot \mathbf{i})$
Axon (Terminals)	Output

Figure 2.1: The biological neuron [13], in contrast with a perceptron.

In this section, we present a taxonomy of neuron models. These neuron models vary in computational and implementation complexity. We discuss some of them in brief, and conduct a thorough review of the Spike Response Model (SRM). For a more in-depth treatment of spiking neuron models, we refer you to [14].

2.3.1 McCulloch-Pitts

Activation functions such as the sigmoid function, and hyperbolic tangent function can be implemented at the hardware level in the McCulloch-Pitts neuron model. However, these models use digital input and output, which means they have a higher footprint than spiking neuron models which perform analog computation.

2.3.2 Biologically-plausible Models

The Hodgkin-Huxley [15] model is a mathematical model that uses four-dimensional non-linear differential equations to model the neuron dynamics, using the transfer of ions as the medium. This model is popular among the neuroscientific community, for accurately modelling the biological process. The Morris-Lecar [16] model is a simplification of the Hodgkin-Huxley model, which is more easily implementable in hardware.

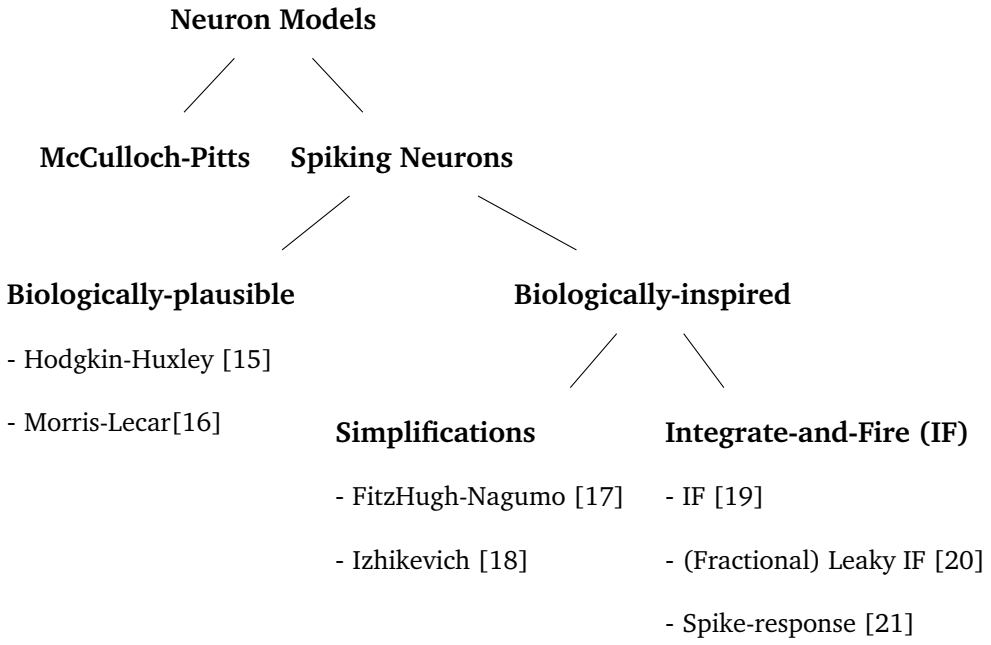


Figure 2.2: A taxonomy of popular spiking neuron models.

2.3.3 Biologically-inspired Models

Some biologically-inspired models often sacrifice biological-plausibility for ease of implementation. These models have a smaller number of parameters, resulting in models that are not only easier to train, but also have a smaller energy footprint. These models include the FitzHugh-Nagumo [17], and the Izhikevich [18] model.

A particularly notable family of biologically-inspired models in the integrate-and-fire family. These models maintain as state, the membrane potential of the neuron, accumulating charge from input neurons. The leaky integrate-and-fire model introduces an additional "leak" term, which controls the decay of the neuron's membrane potential over time.

2.4 The Spike-response Model

The spike-response model [21] is a generalization of the leaky integrate-and-fire model. We review this model because this neuron model is used in SLAYER [22], the framework we use in our experiments.

Rather than modelling the neuronal dynamics using differential equations, the SRM model uses convolutional filters, a property that makes the simulation and training of these spiking models easy. In contrast to the leaky IF model, the refractory behaviour of the neuron is also captured, which has allowed it to better fit spiking data [23].

In this model, a neuron is characterized by a single variable μ, the membrane potential of the cell. Suppose that the neuron has fired its last spike at time $\hat{t}$. Then, in the most general formulation, at each time $t > \hat{t}$, the state of the cell is expressed as an integral over the previous timesteps:

$$\mu(t) = \eta(t - \hat{t}) + \int_{-\infty}^{+\infty} \kappa(t - \hat{t}, s) I(t - s) ds \tag{2.1}$$

where η describes the form of the action potential, κ is the linear response to an input pulse, and $I(t)$ is the stimulating current. In most spiking neural network architectures, the imposed current $I(t)$ is instead described as a weighted sum of presynaptic inputs, and the equation can be rewritten as:

$$\mu(t) = \eta(t - \hat{t}) + \sum_j \sum_f w_j \epsilon(t - \hat{t}, t - t_j^f) \tag{2.2}$$

where ϵ, the spike response kernel, is obtained by convolving the kernel κ with the time course of the presynaptic input.

A common form for the refractory kernel η is:

$$\eta(t) = -\eta_0 \exp\left(-\frac{t}{\tau}\right) \mathcal{H}(t) \tag{2.3}$$

where η_0 is the amplitude of the relative refractorines, τ is a decay time constant, and $\mathcal{H}$ is the Heaviside step function.

SLAYER uses the SRM model, and formulates it this way. Suppose that the neuron receives an input spike train, $s_i(t) = \sum_f \delta\left(t - t_i^{(f)}\right)$. The spike train is represented as a sum of Dirac delta functions with $\delta(x) = 0$ for $x \neq 0$ and $\int_{-\infty}^{\infty} \delta(x) dx = 1$. In the spike response model, incoming spikes are converted into a *spike response signal*, $a_i(t)$, by convolving $s_i(t)$ with a spike response kernel $\epsilon(\cdot)$: $a_i(t) = (\epsilon * s_i)(t)$. The refractory response of the neuron can also

be represented with a refractory kernel $\eta(\cdot)$, giving the formula for the refractory response: $(\eta * s)(t)$ for the neuron's spike train $s(t)$.

Each spike response signal is scaled by a synaptic weight w_i, to generate a post synaptic potential (PSP). The neuron's state is then given by the equation:

$$\mu(t) = (\eta * s)(t) + \sum_i w_i(\epsilon * s_i)(t) = (\eta * s)(t) + \boldsymbol{w}^T \boldsymbol{a}(t) \tag{2.4}$$

The neuron produces an output spike when membrane threshold θ is reached. The spike function $f_s(\cdot)$ is defined as:

$$f_s(\mu) : \mu \to s, s(t) := s(t) + \delta(t - t^{f+1}) \text{ where } t^{f+1} = \min\left\{t : \mu(t) = \theta, t > t^f\right\} \tag{2.5}$$

2.5 Neuromorphic Hardware

To obtain the full benefits of spiking neural networks, they need to be deployed on specialized neuromorphic hardware. In a traditional Von Neumann architecture, the logic core operates on data fetched sequentially from memory. In neuromorphic chips, both computation and memory are instead distributed across computational units that are connected via synapses. Hence, the neuronal architecture and parameters play a key role in information representation and define the computations that are performed.

It has also been observed that spike-trains in the mammalian brain are often sparse in time, suggesting that timing and relative timings of spikes encode large amounts of information. Neuromorphic chips implement this same sparse, low-precision communication protocol between neurons on the chip, and by offering the same asynchronous, event-based parallelism paradigm that the brain uses, are able to perform certain workloads with much less power than Von Neumann chips.

Examples of such chips include IBM's TrueNorth [24] and Intel's Loihi [3]. Loihi features 128 neuromorphic cores, each of which implementing 1024 spiking neural units. The mesh protocol further supports scaling up to 4096 on-chip cores. The large number of computational

units and their interconnectivity is a common feature in neuromorphic chips.

Blouw, Choo, Hunsberger, *et al.* quantified the trade-offs between network size, inference speed, and energy cost between different hardware. The comparison showed that the Intel Loihi was 100 times more efficient than a GPU, with inference speeds also being three times faster [25].

Because spiking neural networks have not yet been successfully trained on many tasks, neuromorphic chips has seen little practical use. These chips have only recently been successfully used in robotic navigation [26], and solving graph problems by manual construction of the network graph [27].

2.6 Training Spiking Neural Networks

It is desirable to train spiking neural networks to perform arbitrary tasks, utilizing power-efficient neuromorphic chips that break the Von Neumann bottleneck. However, spiking neural networks have yet to achieve superior performance relative to their ANN counterparts. In this section, we first discuss the difficulties in training SNNs. Then, we provide a taxonomy of training strategies.

Difficulties in Training

Because backpropagation has been effective in training artificial neural networks, the natural approach is to attempt to apply it to spiking neural networks. However, consider the chain rule in backpropagation:

$$\delta_j^\mu = g'\left(a_j^\mu\right) \sum_k w_{kj} \delta_k^\mu \tag{2.6}$$

δ_j^μ and δ_k^μ denote the partial derivatives of the cost function for input pattern μ with respect to the net input to some arbitrary unit j or k. Unit j projects feed-forward connections to the set of units indexed by k. $g(\cdot)$ is the activation function applied to the net input of unit j, denoted a_j^μ, w_{kj} are the feedforward weights projecting from unit j to the set of units indexed by k.

The chain rule formulation presents two key challenges. First, the gradients $g'(\cdot)$ requires derivatives, but $g(\cdot)$ in spiking neurons is represented by sum of Dirac delta functions, for which derivatives do not exist. The non-differentiability.

Second, the expression $\sum_k w_{kj}\delta_k^\mu$ uses feedforward weights in a feedback fashion. This mean that backpropagation is only possible in the presence of symmetric feedback weights, but these do not exist in the brain. This is also known as the "weight-transport" problem [28]. A more practical problem the usage of non-local information during error assignment. This makes the backpropagation incompatible with many neuromorphic hardware implementations, such as the Loihi: the learning rules of the Loihi are constrained to be of a sum-of-products form.

In the following sections, we present training methods that work around these difficulties, according to the following classification:

Usage of Gradients Recent advancements in the field have opened up the possibilities of using gradient-based methods for training spiking neural networks. We classify the learning algorithms based on whether they utilize gradient information for learning.

Offline/Online Since many neuromorphic hardware implementations impose strict restrictions on the neuron update rules, many learning algorithms use simulations on Von Neumann hardware to learn the SNN weights before deploying them on hardware. We classify the learning algorithms based on whether online learning on neuromorphic chips is possible.

	Non-gradient based	**Gradient-based**
Offline-only	Evolutionary Algorithms	ANN $\rightarrow$ SNN Smooth models Surrogate Gradients
Online	Local Learning Rules	Surrogate Gradients + Random BP

Table 2.3: A taxonomy of learning algorithms for spiking neuron models.

Local Learning Rules

Local learning rules are simple fixed rules, that tells a neuron how to update its weights for each training example. These rules often only use information from neighbouring neurons, and can be easily implemented on neuromorphic hardware. The earliest of such rules is Hebbian learning [29], which can be summarized as "neurons which fire together wire together".

A recent, and much more popular learning rule is the spike-timing-dependent plasticity rule (STDP). Central to these learning rules is the theme that neuron spike ordering and their relative timings encode information. STDP adjusts the strength of connections between neurons using the relative timing of a neuron's output and its input potentials.

In machine learning terminology, the weights of the synapses are adjusted according to fixed rules for each training example. Each synapse is given a weight $0 \leq w \leq w_{\max}$, characterizing its strength, and its change depends on the exact moments t_{pre} of pre-synaptic spikes and t_{post} of post-synaptic spikes [30]:

$$\Delta w = \begin{cases} -\alpha\lambda \cdot \exp\left(-\frac{t_{\text{pre}}-t_{\text{post}}}{\tau_-}\right), \text{if } t_{\text{pre}} - t_{\text{post}} > 0 \\ \lambda \cdot \exp\left(-\frac{t_{\text{post}}-t_{\text{pre}}}{\tau_+}\right), \text{if } t_{\text{pre}} - t_{\text{post}} < 0 \end{cases} \tag{2.7}$$

where τ_+ and τ_- are time constants. $\tau_+ = 16.8ms$ and $\tau_- = 33.7ms$ are reasonable approximations obtained experimentally.

Because the rules are local, these local learning rules fall within the unsupervised learning paradigm of machine learning. However, more recently, variants of STDP such as reward-modulated STDP [31] have been proposed to allow for neurons to learn to update synaptic weights to maximize the global reward of a system.

Libraries such as BindsNET [32] that simulate SNNs on Von Neumann computers implement these rules. Recent attempts have been made to combine Reinforcement Learning and STDP: both in solving reinforcement learning problems [32], and using the reinforcement learning framework to train SNN [33], [34]. However, SNNs trained using the STDP learning rule have yet to achieve comparable performance compared to ANNs on relatively simple datasets like MNIST [12].

Evolutionary Algorithms

In the absence of gradients, neuroevolution is a popular option. Evolutionary algorithms were once thought to be ineffective for training neural networks, which have large parameter spaces. However, advancements in the algorithmic front have made them feasible for solving challenging reinforcement learning problems [35].

One such evolutionary algorithm that can be easily applied to spiking neural networks is differential evolution (DE). In differential evolution, a population of candidate solutions are maintained. At each timestep, the parameters of these agents are tweaked, and the candidates is evaluated. In some variants of DE, a reproduction step involving crossing-over of weights between candidate solutions is also introduced. The agent that has the best performance is returned after some stopping criterion (e.g. time-step) is reached. Evolutionary algorithms tend to be computationally expensive, but parallelizable. For example, in DE the evaluation of candidates can be done in parallel. Hence, it requires offline training. DE has been successfully applied to achieve comparable results with ANNs on simple datasets like IRIS [36].

2.6.1 Gradient-based methods

The class of gradient-based methods use several techniques to overcome the difficulties in training presented in section 2.6. Gradient-based methods seek to optimize a loss function. When the target label is a spike train, a spike train distance metric such as the van-Rossum distance is commonly used. Alternative distance metrics are well covered in [37]. In classification tasks, a common approach is to instead use a rate-based scheme, using the output spike counts and the cross-entropy loss.

Converting ANNs into SNNs

One approach is converting trained ANN models into SNNs [38], [39]. One can achieve the same input-output mapping as a deep ANN with a converted SNN. Common ANN layers such as softmax, batch normalization and max-pooling layers have their corresponding spiking counterparts. This method allows for using the full toolkit of deep-learning methods in training

the ANNs prior to conversion. However, only a subset of ANN architectures can be converted into SNNs. For example, the activations need to be always non-negative, to match the firing rates of SNNs. These converted ANNs are also limited to using rate-codes, which as discussed in section 2.2 is an inefficient spike encoding.

Smooth models

All gradient-based methods applied directly to SNNs require overcoming the discontinuous, binary nature of spike trains. One simple solution is to use smooth models.

First, neuron models with a smooth spike generating process like the Hodgkin-Huxley, Morris-Lecar and FitzHugh-Nagumo models need no further approximation techniques. Next, stochastic models also allow the definition of gradients on expectations, allowing the log-likelihood of a spike train to be optimized using gradient descent [40].

Surrogate Gradients

Rather than using a smooth model, surrogate gradient methods allow for backpropagation without using a smooth model. Surrogate gradient methods replace the spiking non-linearity by the derivative of a smooth function. SpikeProp approximates the membrane threshold function at a local area with a linear function, introducing gradients and computing the exact formulae for error backpropagation for synaptic weights and spike times [41]. Others have modified the threshold function with a gate function [42], used the alpha transfer function to derive gradient update rules [9], and approximate the Dirac delta spikes with a probability density function [22].

Online Gradient-based Learning Rules

While surrogate gradient methods enable backpropagation, these methods use non-local information and cannot be implemented on neuromorphic hardware. Most of these methods cannot train neurons in the hidden layers: they can only train neurons at the final layer, that receive the desired target output pattern [43], [44].

One can relax the requirements of gradient backpropagation [45], to achieve local learning rules. Random backpropagation algorithms sidestep the problem of non-locality, by replacing the weights in the backpropagation rule with random weights. On some benchmark tasks, it was empirically shown to result in little loss in performance. SuperSpike uses random feedback weights, and surrogate gradients to derive a biologically-plausible, online learning rule suitable for hardware implementations [46]. DECOLLE also uses surrogate gradients in a linear IF neuron model, using a local and rate-based cost function to derive local update rules from partial first principles [47].

Equilibrium Propagation was recently proposed to solve the neurobiological incompatibilities of backpropagation [48]. Because the gradients are defined only in terms of local perturbations, the synaptic updates correspond to the standard form of STDP. The propagated signal encodes the gradients of a well-defined objective function on energy-based models, where the goal is to minimize the energy of the model. To resolve the issue of communication using binary-valued signals, step-size annealing was used to train spiking neural networks with Equilibrium Propagation [49].

2.6.2 Conclusion

In this chapter, we have discussed the principles and motivations behind spiking neural networks. We also presented a taxonomy of different spiking neuron models, and their tradeoffs. We focused in particular on the integrate-and-fire family, specifically the spike response model, because of its widespread adoption in the neuromorphic community. We then discussed the challenges in training spiking neuron models, and techniques to side-step them.

The gold standard is an online learning rule that uses only local information, that can be implemented on neuromorphic hardware. We have seen methods such as SuperSpike and DECOLLE, which combine a variety of approximation techniques to derive gradient-based local update rules. These methods do not yet scale well to deep SNNs. How to perform credit assignment using local information in deep networks is still an open research question.

2.6.3 Future Work

In this review, we have spoken mainly about the techniques and the theory, but have little mention of the experiments performed. There are unsurprisingly few publicly available event-based datasets. Many research papers take an existing non-spiking dataset such as MNIST, and convert them into spike trains using a Poisson-encoding process. Others use the N-MNIST and N-Caltech101 dataset [50], or the DvsGesture [51] dataset. The spike trains in these dataset are much less sparse than the naturally occurring spikes in biological neurons. In addition, both SNNs and ANNs demonstrate strong experimental results on these datasets, which suggest that they may be too easy. We would like to see more diverse and challenging publicly-available spiking datasets, and strong and reliable benchmarking for the various SNN architectures and learning methods.

There has also been a resurgence of interest in combining differential equation solvers with neural networks [52]. All the learning methods described in this chapter require the discretization in time. Since the neuronal dynamics of many spiking neurons can be represented as differential equations, continuous-time SNNs may be worth looking into.

Finally, a future direction of research may be the interpretation of spiking neural networks. In deep learning, we have come to learn that convolutional neural networks learn low-level features in lower layers, and higher level features in deeper layers by visualizing the learned filters. Understanding what the spiking neural network is learning at each layer, and how the different layers interact may provide answers to open neuroscience questions, as well as open up new research directions in learning algorithms.

Chapter 3

Related Work

In this section, more relevant to the presented work, we give an overview of related work on visual-tactile perception for robotics.

3.1 Visual-Tactile Perception for Robots

Humans process visual and tactile sensory information in similar ways [53]. This motivates the search for systems that can efficiently exploit both modalities. Early works on visual-tactile sensing date back to 1984, where vision and tactile data was used to create a surface description of primitive objects [54]. In this work, tactile information plays a supporting role for vision, in part due to the low resolution of tactile sensors at that time. Tactile sensing was also used to enhance a robot's ability to categorize objects through behavioural exploration [55], as well as surface classification using deep neural networks [56]. Li, Dong, and Adelson also used sensor information from both a tactile sensor, and a camera to achieve high-accuracy slip detection on unseen objects with a trained deep neural network [57]. Visual-tactile data has also been successfully used to learn compact multimodal representations using self-supervised learning for use in policy learning [58].

Recent advancements in tactile technology [59] have encouraged the use of tactile sensing for more complex tasks, including object exploration [60], shape completion [61], and slip detection [62], [63]. One popular sensor is the BioTac; similar to a human finger, it uses textured

skin, allowing vibration signatures to be used for high accuracy material and object identification and slip detection [64]. Other recent works include the Gelsight [65] — an optical-based tactile sensor — for visual-tactile slip detection, grasp stability and texture recognition.

Different from the prior work, rather than leveraging conventional deep learning methods using traditional synchronous sensors, our sensors and learning method are both event-driven. The novel NeuTouch sensor we used addresses limitations in existing tactile sensing technology, making it suitable for use in robot end-effectors. Our work contributes a system that is able to handle the high temporal resolution of this sensor.

3.2 Event-based Perception: Sensors

Work on event-based perception has focused primarily on vision (see [66] for a comprehensive survey). The emphasis on vision can be attributed both to its applicability across many tasks, as well as the recent availability of event cameras such as the DVS and Prophesee Onboard. Unlike conventional optical sensors, event cameras capture pixel changes asynchronously. SNNs are often evaluated with the DvsGesture dataset [51], a publicly-available dataset of event data captured from the DVS camera, with the task of hand gesture recognition.

Event-based sensors have been successfully used in conjunction with deep learning techniques [66]. The binary events are first converted into real-valued tensors, which are processed downstream by deep ANNs. This approach yields good models (e.g. for motion segmentation [67], optical flow estimation [68], and car steering prediction [69]), but at high compute cost. Spiking neural networks have desirable characteristics such as low latency, high temporal resolution and low power consumption (see section 2.2), that make more suitable for learning with event data.

We have discussed in detail the difficulties and training methods for spiking neural networks in section 2.6. SNNs do not yet outperform their deep ANN cousins on pseudo-event image datasets, and the research community is exploring better training methods for real event data.

In this work, we develop a visual-tactile SNN (VT-SNN) trained using SLAYER [22], an approximate backpropagation method for SNNs. There is little work involving multi-modal SNNs,

and prior work has focused on audio-video data [70], [71] (e.g., for emotion detection [72]). To the best of our knowledge, we are the first to explore a fully event-based visual-tactile multimodal system.

Chapter 4

Robot and Sensors Setup

In this chapter, we describe the robot hardware setup used across our experiments (Fig. 4.1). I once again emphasize that the experimental setup and data collection were done by my collaborators, but I include it here to give a basic understanding of the experiments performed.

We used a 7-DoF Franka Emika Panda arm with a Robotiq 2F-140 gripper and collected data from two primary sensors: the NeuTouch for the tactile modality, and the Prophesee Onboard camera for the visual modality. Both sensors are event-based.

4.1 NeuTouch Tactile Sensor

We use the novel NeuTouch sensor, pictured in Figure 4.2. The NeuTouch sensor resembles a human fingertip, comprising of "skin" and "bone". Tactile sensing is achieved using a layer of electrodes with 39 taxels and a graphene-based piezoresistive thin film. The sensor output is transformed into spiking input in real-time, using the ACES decoder.

We mounted two NeuTouch sensors to the Robotiq 2F-140 gripper and the ACES decoder on the Panda arm (Fig. 4.1, left). To ensure consistent data, we performed a sensor warm-up before each data collection session and obtained baseline results to check for sensor drift.

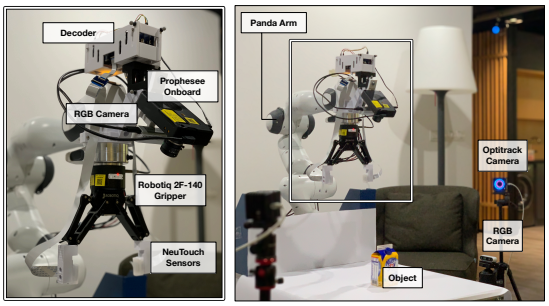


Figure 4.1: Robot Experiment Setup. (**Left**) Close-up of the Franka Emika Panda arm and sensors; a NeuTouch sensor was attached on each Robotiq gripper finger. The Prophesee event and Realsense cameras were mounted on the arm and pointed towards the center of the gripper's grasp area. (**Right**) A view of the object classification experiment showing and the object (soy milk carton) to be grasped and lifted.

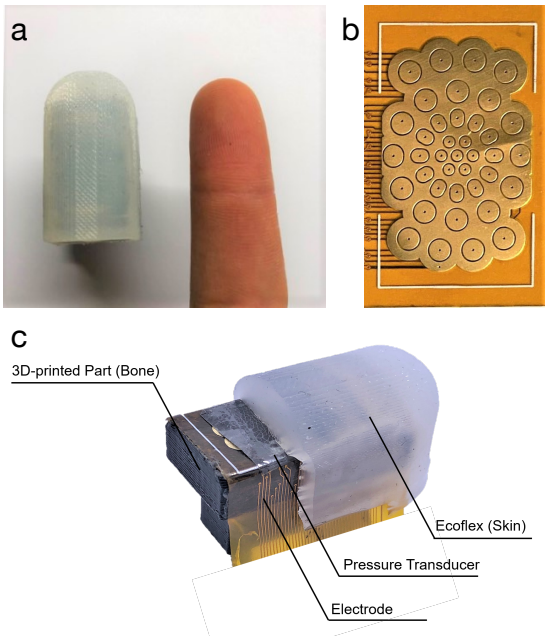


Figure 4.2: (a) NeuTouch compared to a human finger. (b) Spatial distribution of the 39 taxels on NeuTouch. (c) Cross-sectional view of NeuTouch and constituent components. NeuTouch performs tactile sensing using an electrode layer with 39 taxels and a graphene-based piezoresistive thin film that is embedded beneath the protective Ecoflex "skin".

4.2 Prophesee Event Camera

Event-based vision data was captured using the Prophesee Onboard[1]. Similar to the tactile sensor, each camera pixel fires asynchronously and a positive (negative) spike is obtained when there is an increase (decrease) in luminosity. We mount the Onboard on the arm, and point it towards the gripper to obtain information about the object of interest (Fig. 4.1). Although the camera has a maximum resolution of 640×480, we captured spikes from a cropped 200×250 rectangular window to minimize noise from irrelevant regions. The event camera bias parameters were tuned following recommended guidelines[2] and we use the same parameters throughout all experiments.

4.3 Secondary Sensors

In addition to the event-based sensor, we collected sensor data from RGB cameras and the Optitrack motion capture system. These two sensors are not event-based, and their data streams were not used in our visual-tactile perception system.

Two Intel RealSense D435s were used to collect non-event image data, purely for visualization and debugging purposes. The OptiTrack motion capture system was used to collect object movement data for the slip detection experiment. Reflective markers were placed on the rigid parts of the end-effector, and an array of 11 OptiTrack Prime 13 cameras were used to perform marker tracking.

[1] Details available at `https://www.prophesee.ai`.

[2] `https://support.prophesee.ai/portal/kb/articles/bias-tuning`

Chapter 5

Visual-Tactile Spiking Neural Network (VT-SNN)

The Visual-Tactile Spiking Neural Network (VT-SNN) enables learning and perception by using both visual and tactile modalities, and can be easily extended to incorporate other event sensors.

5.1 Model Input

The NeuTouch tactile sensor we use has 39 taxels, with a positive and negative polarity channel per taxel (pictured at Figure 4.2). With 2 fingers, that gives an flattened input size of 156.

For the vision modality, we use a cropped window of resolution 200×250 from the Prophesee event camera. The event-based video data also has 2 channels: for the positive and negative polarities. Hence, the input to our model is of dimensions $2 \times 200 \times 250$.

Including the time dimension (T timesteps), this results in input dimensions of $156 \times 1 \times 1 \times T$, and $2 \times 200 \times 250 \times T$ for the tactile and visual modalities respectively.

5.2 Model Architecture

In our experiments, we use a simple, fully-connected architectures, summarized in Figure 5.1.

For our experiments where our model is only provided with tactile data, the Tactile SNN em-

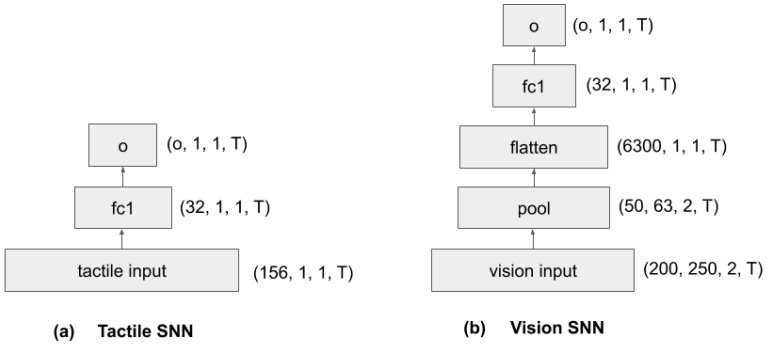


Figure 5.1: The model architectures for (a) Tactile SNN and (b) Vision SNN. The models have a time dimension of length T, and hidden layer size of 32. o is equal to the number of output classes.

ploys a fully-connected (FC) network consisting of 2 dense spiking layers. The layers consist of spiking neurons: SLAYER uses the Spike Response Model (SRM), which we discuss extensively in section 2.4.

The Vision SNN uses 3 layers; the first layer is a pooling layer with kernel size and stride length of 4. The pooling layer is implemented as a 3D convolution, with a fixed kernel of equal weights. This pooling layer serves to reduce the size of the spiking input. The pooled spike train is flattened, and then passed as input to a 2-layer FC architecture identical to the Tactile SNN.

For experiments where both modalities are available to the model, we used a multi-modal architecture (Figure 5.2) very much inspired by existing multi-modal work in the field of multi-modal fusion. We use two layer FC architectures to encode the spiking input from both modalities respectively. The tactile and vision encoders have output sizes o of 50 and 10, respectively. The encoded spike trains are then concatenated, and passed into a dense spiking layer, generating output spikes.

Note that the output dimensionality is dependent on the task: 20 for container & weight classification, and 2 for rotational slip classification. The model architectures are agnostic to the size of the input time dimension (number of discrete timesteps in the input spike train), and

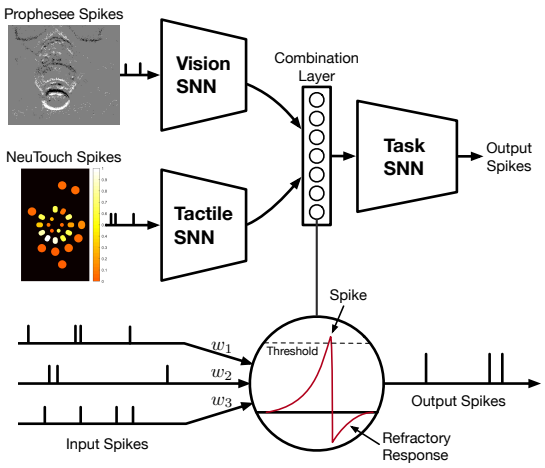


Figure 5.2: The Visual-Tactile Spiking Neural Network (VT-SNN) comprises of two "spiking encoders" for each modality. The encoded input for both modalities are concatenated at the combination layer, and is densely connected to the output layer, whose size is task-specific.

the same model architectures are used in both classification tasks.

5.3 Model Training

We optimize our spiking neural networks using SLAYER [22]. As discussed in section 2.6, back-propagation cannot be directly applied to spiking neural networks because of the binary, non-continuous nature of spiking input. SLAYER overcomes this problem via a stochastic spiking neuron approximation to derive an approximate gradient, and temporal credit assignment policy to distribute errors.

While SLAYER requires conventional GPU hardware to train models, the learned weights can be transferred onto neuromorphic hardware such as Loihi [3]. This work is presented later in chapter 8. Both Loihi and the GPU models operate on an algorithmic discrete time-step model, and the spiking data needs to be binned into fixed-width intervals. We used a straightforward binning process where the (binary) value for each bin window V_w was 1 whenever the

total spike count in that window $\sum_w S$ exceeded a threshold value S_{min}:

$$V_w = \begin{cases} 1 & \sum_w S \geq S_{\min} \\ 0 & \text{otherwise.} \end{cases} \tag{5.1}$$

The SNN accumulates spikes over time, and a prediction is made after T timesteps, where T is the length of the time dimension (number of bins). Following [22], class prediction is determined by the number of spikes in the output layer. Each output neuron is associated with an output class and the neuron that generates the most spikes represents the winning class.

We trained the model by minimizing the loss:

$$\mathcal{L} = \frac{1}{2} \sum_{t=0}^{T} \left(\sum \mathbf{s}^o(t) - \sum \tilde{\mathbf{s}}^o(t) \right)^2 \tag{5.2}$$

which captures the difference between the observed output spike count $\sum \mathbf{s}^o(t)$ and the desired spike count $\sum \tilde{\mathbf{s}}^o(t)$ for output neuron o.

For example, in the container classification task there are 20 output classes: a permutation of 4 objects and 5 weight classes. For this task, the spiking output layer has 20 output neurons, one for each output class. Spiking input is passed through the network, and the spike counts at the output neurons are collected. The predicted output is the class with the most number of spikes.

Appropriate counts have to be specified for the correct and incorrect class and are application-specific hyperparameters, which we tune manually for each task. The hyperparameters and some discussion around it are provided in Appendix A.

From initial trials, we observed that training solely with the classification loss above on our data led to rapid over-fitting and poor test accuracy, possibly due to the small size of our dataset. We explored several different techniques to mitigate this issue (e.g., ℓ_1 regularizers and dropout), none of which showing significant improvement. We eventually found that simple ℓ_2 regularization on the weights of the network was sufficient:

$$\hat{\mathcal{L}} = \mathcal{L} + \gamma_2 \|\theta\|_2^2 \tag{5.3}$$

where θ comprise the network weights. For all our SNN models, we optimized $\hat{\mathcal{L}}$, with a tuned $\gamma = 0.05$.

Chapter 6

Container & Weight Classification

In this chapter, we task VT-SNN to classify containers, containing differing amounts of liquid. We designed this experiment to determine the performance of our visual-tactile perception system, on a task which would be difficult from a single modality. Because the objects are opaque, it would be difficult to tell solely from vision how much liquid it contained.

6.1 Data Collection

We used 4 different containers: an aluminium coffee can, a plastic pepsi bottle, a cardboard soy milk carton, and a tuna can.

We chose these four containers because of their varying hardness. We collected data for each container, containing {0%, 25%, 50%, 75%, 100%} of the maximum amount of liquid they could contain[1]. The robot would grasp and lift each object class fifteen times, yielding 15 samples per class (see Figure 6.2 for a breakdown). Hence, we obtain a dataset of 20 object classes, for four containers and 5 weight classes each.

[1]The can did not have a cover and we filled it with a packet of rice to avoid spills and possible liquid damage. The tuna can was placed with the open side facing downwards so, the rice was not visible.

Figure 6.1: Containers used for container classification task: coffee can, plastic soda bottle, soy milk carton, and metal tuna can.

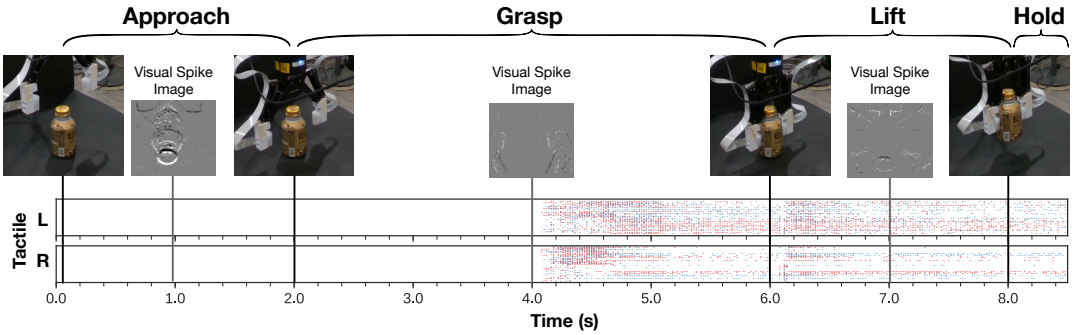


Figure 6.2: Data collection procedure for the container & weight classification task. The robot end-effector starts at 10 cm above the grasp point. It then *approaches* the coffee can, stopping at the 2-second mark. The gripper then starts to close, *grasping* it at around the 4-second mark, resulting in tactile spikes. At the 6-second mark, the robot *lifts* it by 5cm above the table. The robot then holds it in the same position till the 8.5-second mark. For container and weight classification, we only use data starting from the 2-second mark.

6.2 Data Pre-processing

For both modalities, we selected data from the grasping, lifting and holding phases. This corresponds to the 2.0s to 8.5s window in Figure 6.2.

We set a bin duration of 0.02s (325 bins) and a binning threshold value $S_{\text{min}} = 2$. We used stratified K-folds to create 5 splits; each split contained 240 training and 60 test examples with equal class distribution.

6.3 Model Comparison

We compared VT-SNN with traditional deep learning models, which are fed the same binary spiking input. We use Multi-layer Perceptrons (MLPs) with Gated Recurrent Units (GRUs) [73] to handle the time-series input.

We trained each model using (i) the tactile data only, (ii) the visual data only, and (iii) the combined visual-tactile data. We implemented all the models using PyTorch. The SNNs were trained with SLAYER to minimize spike count differences [22] and the ANNs were trained to minimize the cross-entropy loss. All models were trained using RMSProp, for a total of 500 epochs.

Because the dataset is small, we are unable to create a validation split, without significantly reducing the amount of training or test data. Instead, we use the parameters of the model at the 500th epoch to compute their accuracies.

6.4 Results and Analysis

We summarize the test accuracies of the models in Table 6.1. We found that the tactile-only model gave a 12% higher accuracy than the vision-only model. We had initially expected the performance of the vision-only model to be poor, because we had hypothesized that vision would be unable to differentiate between the weight classes for each object, since they were opaque. However, by examining the data we found that the Pepsi bottle was not fully opaque, and the water level was observable by the Onboard vision camera on some trails. In addition,

Table 6.1: Container & Weight Classification: Average Accuracy with Standard Deviation in Brackets

Model	Tactile	Vision	Combined
SNN	0.700 (0.045)	0.579 (0.064)	0.811 (0.048)
ANN (MLP-GRU)	0.497 (0.059)	0.429 (0.054)	0.437 (0.062)

on softer objects, the vision camera was able to see deformations of the object as the gripper closed, revealing how much liquid was in the container.

As expected for the task, the multi-modal SNN model outperforms both unimodal models, achieving the highest accuracy of 81%. This suggests that both modalities provide essential information for the task, and that our model is able to successfully exploit both modalities. Figure 6.3 shows the spiking output for the three modality combinations, for a coffee can with 100% weight. The trained tactile-only and vision-only models are uncertain of the container and the weight category respectively: the tactile model is confuses the tuna can and coffee can, while vision model correctly predicts the container but is unsure about the weight category. The combined visual-tactile model incorporates information from both the modalities and is able to predict the correct class with high certainty.

The SNN models performed far better than the ANN models. We had expected comparable performance since MLP-GRU models are known to perform well on a variety of tasks. The poor performance of the ANN models were possibly due to the relatively long sample durations (325 time-steps). This resulted in the large number of parameters in the ANN models, which are difficult to learn for a small dataset. On the other hand, the SNN models have less than 10,000 parameters.

6.4.1 Early Classification

One feature of spiking neural networks lacking in conventional deep models is the ability to perform early classification: instead of waiting for all the output spikes to accumulate, we can

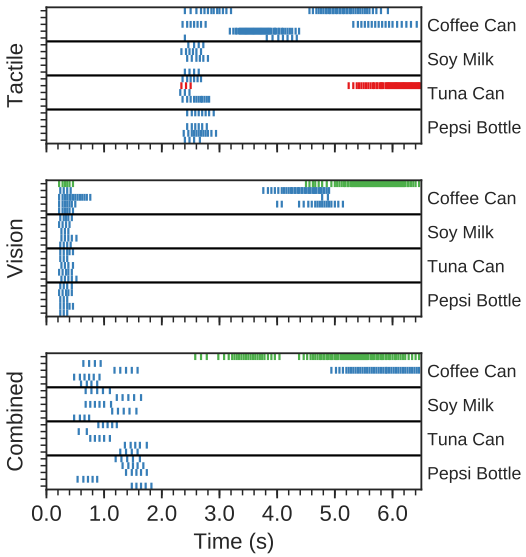


Figure 6.3: Output spikes for the models trained with different modalities with correct and incorrect predictions in green and red, respectively. The weight categories are arranged from 0% to 100% (bottom to top) for each container. The tactile model is unable to distinguish between a coffee can and a tuna can while the vision model is uncertain about the weight. The combined visual-tactile model predicts the correct class with high certainty.

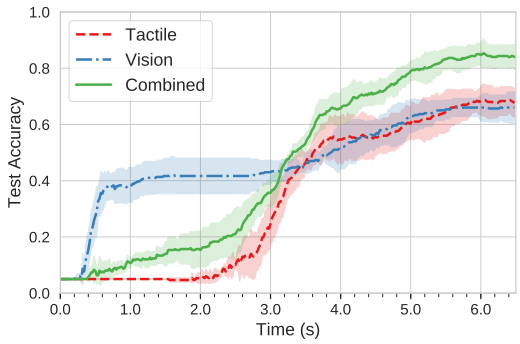


Figure 6.4: Container and weight classification accuracy over time. Lines show average test accuracy and shaded regions represent the standard deviations. Vision-only classification results in higher early accuracy as visual spikes are obtained as the gripper is closing, and tactile events arise only upon contact with the object. Combining both vision and tactile event data via our VT-SNN results in significantly higher accuracy, compared to using each modality separately.

perform classification based on the number of spikes seen up to time t. Fig. 6.4 shows the accuracies of the different models over time.

The combined visual-tactile model achieves the highest accuracies overall, but between $0.5-3.0$s, the vision model was already able to distinguish between objects. This is possibly a result of small movements (of the mounted camera) as the gripper closed, which resulted in minute but noticeable changes perceived by the Onboard. As expected, tactile spikes do not emerge until contact is made with the object at ≈ 2s.

Chapter 7

Rotational Slip Classification

In this chapter we discuss the second task: rotational slip classification. This is an important task for stable grasping: stable grasp points are often incorrectly predicted for objects with a center-of-mass that are not obvious from sight, such as a hammer, or objects with irregular shapes. Detecting rotational slip allows the controller to re-grasp the object upon slip, or change grasp locations. For re-grasping to be effective, the rotational slip detection needs to be quick and accurate. By using the additional tactile modality, we hoped that the robot would be able sense rotational slip more accurately.

7.1 Data Collection

We constructed a test object using Lego Duplo blocks (Fig. 7.2) with a hidden 10g mass in each leg. The "control" object was designed to be balanced at the grasp point. To induce rotational slip, the object was modified by transferring the hidden mass from the right leg to the left. As such, the stable and unstable objects were visually identical and had the same overall weight.

The robot was set to grasp and lift both object variants 50 times, yielding 50 samples per class. Similar to the previous experiment, the robot was instructed to close upon the object, lift the object by 10cm off the table (in 0.75 seconds) and hold the position for another 4.25 seconds. The gripper's grasping force was tuned to enable the object lift, and yet allow for rotational slip for the off-center object (Fig. 7.2, right).

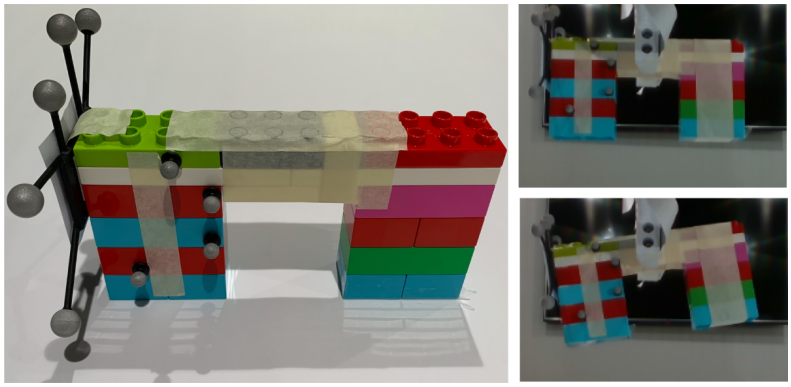


Figure 7.1: (**Left**) Object for the Slip Classification Task with attached OptiTrack markers. (**Right**) Object during a stable grasp (top) and unstable grasp with rotational slip (bottom); the bottom object has additional mass transferred to the left leg (from the right leg) that causes it to rotate during lifting. Note that both objects had equal mass.

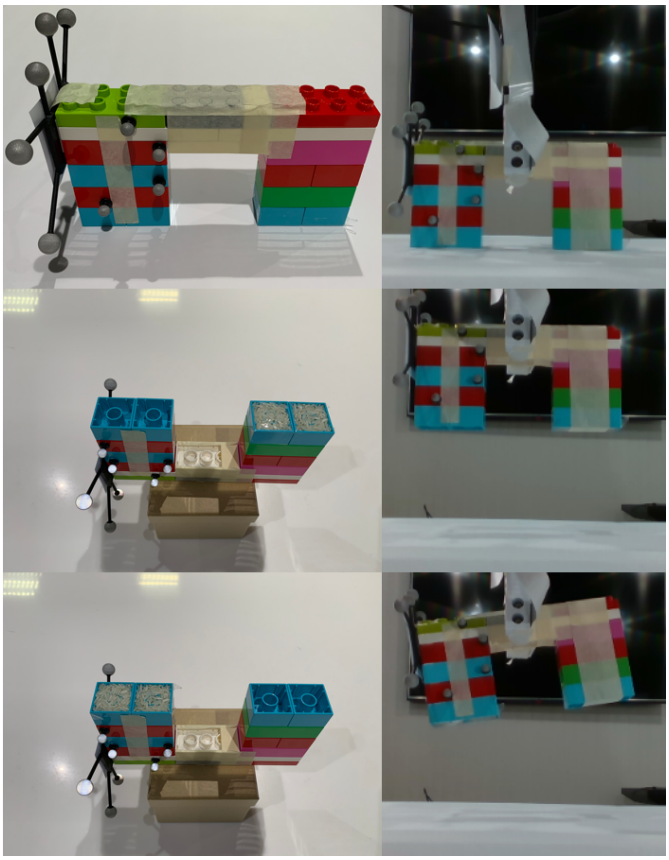


Figure 7.2: Top: Object and the fixed grasp point. Middle: Center of mass aligns with grasp point, and no rotational slip happens upon lifting the object. Bottom: Center of mass is shifted away from grasp point, rotational slip happens upon lift.

Table 7.1: Rotational Slip Detection: Average Accuracy with Standard Deviation in Brackets

Model	**Tactile**	**Vision**	**Combined**
SNN	0.837 (0.044)	1.0 (0.0)	1.0 (0.0)
ANN (MLP-GRU)	0.860 (0.069)	1.0 (0.0)	1.0 (0.0)

The OptiTrack data was used to annotate rotational slip: specifically, we obtained the baseline orientation distribution (for 1 second or 120 frames) and defined rotational slip as an orientation larger (or smaller) than 98% of the baseline frames lasting more than four consecutive OptiTrack frames. From our collected data, we found that the rotational slip occurred almost immediately during lifting.

7.2 Data Preprocessing

We extracted a 0.15s window around the start of the lift, and set a bin duration of 0.001s (150 bins) with binning threshold $S_{\text{min}} = 1$, to obtain spiking input of 150 time-steps. Again, we used stratified K-folds to obtain 5 splits, where each split contained 80 training examples and 20 testing examples.

As with the last experiment, we train the models on the three modality combinations, and compare the classification accuracies between SNNs and ANNs. Note that the task is now a binary classification task, so the output layers have size 2.

7.3 Results and Analysis

The test accuracy of the models are summarized in Table 7.1. For both the SNN and ANN, both the vision and multi-modal models achieve 100% accuracy. This is unsurprising, as rotational slip would produce a distinctive slipping signature. On the other hand, the tactile-only model achieves only a 84% and 86% accuracy. This could be due to the deformation of the Ecoflex skin in the direction of the rotational slip, resulting in a less distinctive signature. We would

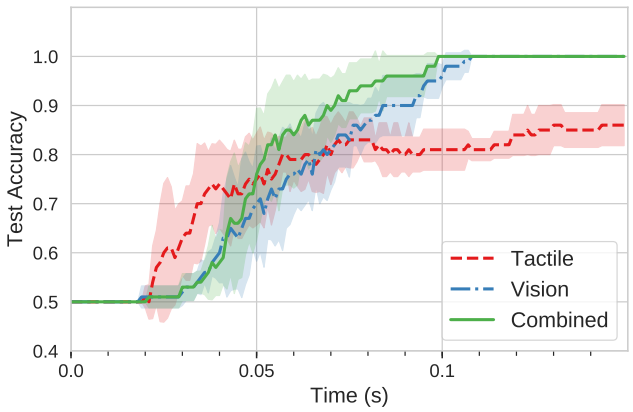


Figure 7.3: Rotational slip classification accuracy over time. Lines show average test accuracy and shaded regions represent the standard deviations. Classification using tactile-only data results in higher early accuracy, but vision-only classification is more accurate as sensory data is accumulated. Combining both modalities results in higher accuracy, especially after 0.05s.

need to run further experiments with different skin materials to see if that would mitigate this issue.

7.3.1 Early Classification

While at face value, the tactile modality seems to provide no benefit for this task, conducting the same early classification analyses as earlier yields an interesting result.

Figure 7.3 summarizes slip test accuracy over time. We see that the tactile-only model is more accurate earlier (between $0.025-0.05$s), than both the multi-modal and vision-only model. In addition, the multi-modal model has an accuracy profile that is almost strictly superior to the vision model: if a prediction were to be made at any time-step, the multi-modal model yields better predictions. This suggests that the tactile modality provides valuable information during early stages of slippage, and would be useful for the task of rapid slippage detection. This is further validated when we plot the sensor spikes over time (see Figure 7.4). The NeuTouch starts spiking earlier than the Prophesee camera.

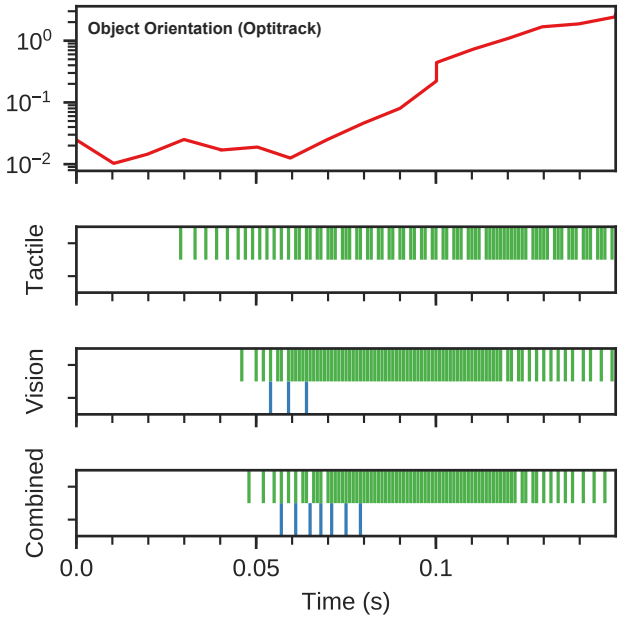


Figure 7.4: Output spikes for the slip detection models trained with different modalities with correct prediction class in green. All models predict this sample correctly, but the tactile model spikes earlier compared to the other two models.

Chapter 8

Gains on Neuromorphic Hardware

In subsection 2.2.2, we discussed that one of the main motivations of using event-based sensors, and spiking neural networks is the promise of power efficiency and real-time performance. In this chapter, we attempt to quantify these gains by evaluating VT-SNN on the Loihi chip against a GPU. This portion of work was completed after the paper submission.

8.1 The Intel Loihi

Loihi is designed for running spiking neural networks. Loihi uses a fixed-size discrete time-step model, carrying out computation through the interacting dynamics of its neuron states. It was designed to exploit the parallelism of SNN-based algorithms, by computing all individual neuron states within a time-step concurrently. This was previously not possible on conventional CPU architectures.

Loihi uses a variation of the leaky integrate-and-file neuron model that has two internal state variables: the synaptic response current and the membrane potential. The synaptic response current is the sum of filtered input spike trains, and some constant bias current. In our models, we set the bias current to 0.

8.2 The VT-SNN Loihi Model

SLAYER [22] PyTorch library supports the Loihi neuron model. To learn a model that can run on the Loihi, we replace the default Spike-response Model with the Loihi neuron model, and run the same training procedure as described in section 5.3. We obtain a SNN architecture that is identical to the VT-SNN model, with only the underlying neuron model being different.

8.3 Methodology

We benchmark power usage and inference speeds on two environments:

1. The CLeAR lab's compute cluster "beast": using a GeForce RTX 2080 Ti
2. Intel's neuromorphic cluster. The Loihi board is accessed via Slurm from Intel's clusters.

Because the SLAYER models can only be run on a GPU or on neuromorphic hardware, we use only these two devices in our comparison. In the interest of time, we evaluate Tactile SNN model for the task of container classification.

We adapt the approach of Blouw, Choo, Hunsberger, *et al.*, who quantified the power consumption and inference time across various hardware devices on the task of keyword spotting [25]. We use a script that performs a forward pass of the model for our test set containing 60 test examples. We run this script for a set amount of time (15 minutes), and compute the average time taken for each forward pass. We integrate the energy consumed over the 15 minute interval to obtain the total amount of energy consumed during the time period. We then average it over the total number of inferences made. We account for idle power consumption for each device by subtracting a baseline average computed over 15 minutes.

In this experiment, we used a batch size of 1. We note that while batching can significantly improve power efficiency on a cost-per-inference basis [25], this can only be done in cases involving offline data analysis, and an online application of our models is of interest.

For the GPU device, information about power consumption is exposed through the command-line utility `nvidia-smi`. We run a script that logs power draw with the corresponding times-

	GPU	Loihi
Idle Power Consumption (W)	5.123	0.0295
Running Power Consumption (W)	56.329	0.104
Dynamic Power Consumption (W)	51.206	0.0745
Inferences Per Second	243.76	90.10 (296.46*)
Energy Cost per Inference (J)	0.210	0.000826 (0.000272*)

Table 8.1: Power consumption and inference speeds for the Loihi and the GPU on the task of container classification. The Loihi uses significantly less power. The inference speeds for Loihi are lower, but discounting communication time (in brackets), the inference speeds become comparable to GPUs.

tamps. We use the timestamps in model inference to filter for the power draw during model inference, and compute the average power used for each inference. We track the number of forward passes made by the script, and compute the number of forward passes made per second. We then use the power draw information over the 15 minute run, to compute the amount of energy consumed per inference.

Intel's NxSDK, which is used to program the Loihi, exposes performance profiling using probes. Energy consumption is measured using the host CPU, while the inference time is measured by the embedded CPU on the Loihi chip, which drives the advancement of the algorithmic timesteps. The probe output the energy per timestep during different phases of execution. We use the total energy per timestep in all phases for our comparison.

8.4 Benchmarking Results

The benchmark results are presented in Table 8.1. The benchmarks show that the power consumption on the Loihi board is many orders of magnitude lower than that of the GPU. In fact, the running power consumption of the Loihi is even lower than that of an idle GPU. This is an encouraging find, verifying the power savings of using neuromorphic hardware.

```
INFO:DRV:  Executor: 325 timesteps.........Done 0.06s
INFO:DRV:       Transferring spikes.........Done 0.03s
INFO:DRV:       Configuring registers.......Done 3.09ms
INFO:DRV:       Transferring probes.........Done 0.93ms
INFO:DRV:       Executing...................Done 2.17ms
INFO:DRV:       Processing timeseries.......Done 0.90ms
```

Figure 8.1: Log of a typical run on Loihi. Much of the time is spent transferring the spiking data onto the board. If we consider just the execution and processing time, the inference speed is slightly better than the GPU.

The results we obtained for the inference speeds were contradictory to the findings in [25]. At first glance, it seems that GPUs are able to perform inference almost $3\times$ faster than Loihi. However, looking deeper into the time spent in each phase during the Loihi benchmark, we find that a large amount of time is spent communicating between the cluster and the board (see Figure 8.1). If we discount the communication time, we find that Loihi is able to perform 296.46 inferences per second, which is slightly faster than the GPU — similar to the results found in [25].

8.5 Conclusion

The initial benchmark results are encouraging: the power efficiency gains demonstrated are significant, and would be beneficial in mobile robotic appliances. To obtain a fairer comparison for the Loihi, obtaining the physical device would be helpful.

Chapter 9

Conclusion

In this work, we propose an event-based perception framework, VT-SNN, that combines vision and touch to achieve better performance on two robot tasks. In contrast to conventional synchronous systems, our event-driven framework can asynchronously processes discrete events and as such, may achieve higher temporal resolution and low latency, with low power consumption. We also demonstrated a unique feature of spiking neural networks: early classification, which has interesting applications in robotic applications that require rapid inference, such as slippage detection.

Our proposed framework, VT-SNN is able to learn directly from event data. We demonstrated that in both robot tasks, the combination of vision and tactile modalities yield the best results, which encourages further research on such multi-modal systems. Our benchmark runs on neuromorphic hardware also demonstrated that our model can run with orders-of-magnitude lower power consumption, while maintaining the same inference speeds. We believe that event-based sensing and learning will form essential parts of next-generation real-time autonomous robots that are power-efficient.

9.1 Future Work

In this thesis we have only taken the first steps towards power-efficient event-driven robotic applications: there is still much to explore. First, we have only adopted a simple multi-modal

architecture that performs implicit multi-modal alignment. Future work can consider using architectures that better exploit relations between the two modalities. For example, one may consider using attention models to allow the event-based perception system to fuse modalities more effectively.

In addition, our experimental setup we binned the sensor data into large discrete time-steps, resulting in loss of temporal information. Neuromorphic hardware should be able to handle the event data despite the high temporal resolution, and in future we hope to run experiments to confirm this.

Finally, the trained models are currently classifiers, and have not been implemented as robotic controllers. Using spiking neural networks for closed-loop control seems like a natural next step.

Bibliography

[1] D. Li, X. Chen, M. Becchi, and Z. Zong, "Evaluating the energy efficiency of deep convolutional neural networks on cpus and gpus", in *2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom)*, Oct. 2016, pp. 477–484.

[2] M. Pfeiffer and T. Pfeil, "Deep learning with spiking neurons: Opportunities and challenges", *Frontiers in neuroscience*, vol. 12, 2018.

[3] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning", *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[4] W. W. Lee, Y. J. Tan, H. Yao, S. Li, H. H. See, M. Hon, K. A. Ng, B. Xiong, J. S. Ho, and B. C. K. Tee, "A neuro-inspired artificial peripheral nervous system for scalable electronic skins", *Science Robotics*, vol. 4, no. 32, 2019. eprint: `https://robotics.sciencemag.org/content/4/32/eaax2198.full.pdf`. [Online]. Available: `https://robotics.sciencemag.org/content/4/32/eaax2198`.

[5] W. Maass, "Networks of spiking neurons: The third generation of neural network models", *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997, ISSN: 0893-6080. [Online]. Available: `http://www.sciencedirect.com/science/article/pii/S0893608097000117`.

[6] M. Stemmler, "A single spike suffices: The simplest form of stochastic resonance in model neurons", *Network: Computation in Neural Systems*, vol. 7, no. 4, pp. 687–716, 1996. [Online]. Available: https://doi.org/10.1088/0954-898x_7_4_005.

[7] S. Thorpe, A. Delorme, and R. Van Rullen, "Spike-based strategies for rapid processing", *Neural networks*, vol. 14, no. 6-7, pp. 715–725, 2001.

[8] R. Gütig, "To spike, or when to spike?", *Current Opinion in Neurobiology*, vol. 25, no. nil, pp. 134–139, 2014. [Online]. Available: https://doi.org/10.1016/j.conb.2014.01.004.

[9] I. M. Comsa, K. Potempa, L. Versari, T. Fischbacher, A. Gesmundo, and J. Alakuijala, "Temporal coding in spiking neural networks with alpha synaptic function", *CoRR*, 2019. arXiv: 1907.13223 [cs.NE]. [Online]. Available: http://arxiv.org/abs/1907.13223v2.

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[11] J. Backus, "Can programming be liberated from the von neumann style?: A functional style and its algebra of programs", *Communications of the ACM*, vol. 21, no. 8, pp. 613–641, Aug. 1978, ISSN: 0001-0782. [Online]. Available: http://dx.doi.org/10.1145/359576.359579.

[12] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, "Deep learning in spiking neural networks", *Neural Networks*, vol. 111, pp. 47–63, 2019, ISSN: 0893-6080. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608018303332.

[13] R. Nagyfi. (2018). The differences between artificial and biological neural networks, [Online]. Available: https://towardsdatascience.com/the-differences-between-artificial-and-biological-neural-networks-a8b46db828b7.

[14] E. M. Izhikevich, "Which model to use for cortical spiking neurons?", *IEEE transactions on neural networks*, vol. 15, no. 5, pp. 1063–1070, 2004.

[15] 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*, vol. 117, no. 4, pp. 500–544, 1952.

[16] C. Morris and H. Lecar, “Voltage oscillations in the barnacle giant muscle fiber”, *Biophysical Journal*, vol. 35, no. 1, pp. 193–213, 1981. [Online]. Available: https://doi.org/10.1016/s0006-3495(81)84782-0.

[17] R. FitzHugh, “Mathematical models of threshold phenomena in the nerve membrane”, *The bulletin of mathematical biophysics*, vol. 17, no. 4, pp. 257–278, 1955.

[18] E. M. Izhikevich, “Simple model of spiking neurons”, *IEEE Transactions on neural networks*, vol. 14, no. 6, pp. 1569–1572, 2003.

[19] L. Abbott, “Lapicque’s introduction of the integrate-and-fire model neuron (1907)”, *Brain Research Bulletin*, vol. 50, no. 5-6, pp. 303–304, 1999. [Online]. Available: https://doi.org/10.1016/s0361-9230(99)00161-6.

[20] W. Teka, T. M. Marinov, and F. Santamaria, “Neuronal spike timing adaptation described with a fractional leaky integrate-and-fire model”, *PLoS Computational Biology*, vol. 10, no. 3, e1003526, 2014. [Online]. Available: https://doi.org/10.1371/journal.pcbi.1003526.

[21] W. Gerstner, “A framework for spiking neuron models-the spike response model”, *Handbook of Biological Physics*, vol. 4, no. BOOK_CHAP, pp. 469–516, 2001.

[22] S. B. Shrestha and G. Orchard, “Slayer: Spike layer error reassignment in time”, in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., Curran Associates, Inc., 2018, pp. 1412–1421. [Online]. Available: http://papers.nips.cc/paper/7415-slayer-spike-layer-error-reassignment-in-time.pdf.

[23] R. Jolivet, T. J. Lewis, and W. Gerstner, “Generalized integrate-and-fire models of neuronal activity approximate spike trains of a detailed model to a high degree of accu-

racy", *Journal of Neurophysiology*, vol. 92, no. 2, pp. 959–976, 2004. [Online]. Available: https://doi.org/10.1152/jn.00190.2004.

[24] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, "A million spiking-neuron integrated circuit with a scalable communication network and interface", *Science*, vol. 345, no. 6197, pp. 668–673, 2014, ISSN: 0036-8075. eprint: https://science.sciencemag.org/content/345/6197/668.full.pdf. [Online]. Available: https://science.sciencemag.org/content/345/6197/668.

[25] P. Blouw, X. Choo, E. Hunsberger, and C. Eliasmith, "Benchmarking keyword spotting efficiency on neuromorphic hardware", *CoRR*, 2018. arXiv: 1812.01739 [cs.LG]. [Online]. Available: http://arxiv.org/abs/1812.01739v2.

[26] G. Tang, A. Shah, and K. P. Michmizos, "Spiking neural network on neuromorphic hardware for energy-efficient unidimensional slam", *CoRR*, 2019. arXiv: 1903.02504 [cs.RO]. [Online]. Available: http://arxiv.org/abs/1903.02504v2.

[27] W. Severa, O. Parekh, K. D. Carlson, C. D. James, and J. B. Aimone, "Spiking network algorithms for scientific computing", *2016 IEEE International Conference on Rebooting Computing (ICRC)*, pp. 1–8, 2016.

[28] S. Grossberg, "Competitive learning: From interactive activation to adaptive resonance", *Cognitive science*, vol. 11, no. 1, pp. 23–63, 1987.

[29] D. O. Hebb, *The organization of behavior: a neuropsychological theory*. J. Wiley; Chapman & Hall, 1949.

[30] A. Sboev, D. Vlasov, R. Rybka, and A. Serenko, "Spiking neural network reinforcement learning method based on temporal coding and stdp", *Procedia Computer Science*, vol. 145, no. nil, pp. 458–463, 2018. [Online]. Available: https://doi.org/10.1016/j.procs.2018.11.107.

[31] R. Legenstein, D. Pecevski, and W. Maass, "A learning theory for reward-modulated spike-timing-dependent plasticity with application to biofeedback", *PLoS Computational Biology*, vol. 4, no. 10, e1000180, 2008. [Online]. Available: https://doi.org/10.1371/journal.pcbi.1000180.

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

[33] Z. Bing, I. Baumann, Z. Jiang, K. Huang, C. Cai, and A. Knoll, "Supervised learning in snn via reward-modulated spike-timing-dependent plasticity for a target reaching vehicle", *Frontiers in Neurorobotics*, vol. 13, p. 18, 2019, ISSN: 1662-5218. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnbot.2019.00018.

[34] C. Lee, P. Panda, G. Srinivasan, and K. Roy, "Training deep spiking convolutional neural networks with stdp-based unsupervised pre-training followed by supervised fine-tuning", *Frontiers in Neuroscience*, vol. 12, p. 435, 2018, ISSN: 1662-453X. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2018.00435.

[35] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune, "Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning", *CoRR*, 2017. arXiv: 1712.06567 [cs.NE]. [Online]. Available: http://arxiv.org/abs/1712.06567v3.

[36] N. Pavlidis, O. Tasoulis, V. Plagianakos, G. Nikiforidis, and M. Vrahatis, "Spiking neural network training using evolutionary algorithms", in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, - nil, nil. [Online]. Available: https://doi.org/10.1109/ijcnn.2005.1556240.

[37] J. D. Victor, "Spike train metrics", *Current Opinion in Neurobiology*, vol. 15, no. 5, pp. 585–592, 2005. [Online]. Available: https://doi.org/10.1016/j.conb.2005.08.002.

[38] B. Rueckauer, I.-A. Lungu, Y. Hu, and M. Pfeiffer, "Theory and tools for the conversion of analog to spiking convolutional neural networks", *CoRR*, 2016. arXiv: 1612.04052 [stat.ML]. [Online]. Available: http://arxiv.org/abs/1612.04052v1.

[39] D. Zambrano and S. M. Bohte, "Fast and efficient asynchronous neural computation with adapting spiking neural networks", *CoRR*, 2016. arXiv: 1609.02053v1 [cs.NE]. [Online]. Available: http://arxiv.org/abs/1609.02053v1.

[40] J.-P. Pfister, T. Toyoizumi, D. Barber, and W. Gerstner, "Optimal spike-timing-dependent plasticity for precise action potential firing in supervised learning", *Neural Computation*, vol. 18, no. 6, pp. 1318–1348, 2006. [Online]. Available: https://doi.org/10.1162/neco.2006.18.6.1318.

[41] S. Bohte, J. Kok, and J. Poutré, "Spikeprop: Backpropagation for networks of spiking neurons.", vol. 48, Jan. 2000, pp. 419–424.

[42] D. Huh and T. J. Sejnowski, "Gradient descent for spiking neural networks", in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., Curran Associates, Inc., 2018, pp. 1433–1443. [Online]. Available: http://papers.nips.cc/paper/7417-gradient-descent-for-spiking-neural-networks.pdf.

[43] R. Urbanczik and W. Senn, "A gradient learning rule for the tempotron", *Neural Computation*, vol. 21, no. 2, pp. 340–352, 2009. [Online]. Available: https://doi.org/10.1162/neco.2008.09-07-605.

[44] J. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation", *Frontiers in Neuroscience*, vol. 10, Aug. 2016.

[45] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks", *CoRR*, 2019. arXiv: 1901.09948 [cs.NE]. [Online]. Available: http://arxiv.org/abs/1901.09948v2.

[46] F. Zenke and S. Ganguli, "Superspike: Supervised learning in multi-layer spiking neural networks", *CoRR*, 2017. arXiv: 1705.11146 [q-bio.NC]. [Online]. Available: http://arxiv.org/abs/1705.11146v2.

[47] J. Kaiser, H. Mostafa, and E. Neftci, "Synaptic plasticity dynamics for deep continuous local learning (decolle)", *CoRR*, 2018. arXiv: 1811.10766 [cs.NE]. [Online]. Available: http://arxiv.org/abs/1811.10766v3.

[48] B. Scellier and Y. Bengio, "Equilibrium propagation: Bridging the gap between energy-based models and backpropagation", *Frontiers in Computational Neuroscience*, vol. 11, p. 24, 2017, ISSN: 1662-5188. [Online]. Available: https://www.frontiersin.org/article/10.3389/fncom.2017.00024.

[49] P. O'Connor, E. Gavves, and M. Welling, "Training a spiking neural network with equilibrium propagation", in *Proceedings of Machine Learning Research*, K. Chaudhuri and M. Sugiyama, Eds., ser. Proceedings of Machine Learning Research, vol. 89, PMLR, 16–18 Apr 2019, pp. 1516–1523. [Online]. Available: http://proceedings.mlr.press/v89/o-connor19a.html.

[50] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor, "Converting static image datasets to spiking neuromorphic datasets using saccades", *Frontiers in Neuroscience*, vol. 9, no. nil, nil, 2015. [Online]. Available: https://doi.org/10.3389/fnins.2015.00437.

[51] A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. D. Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza, J. Kusnitz, M. Debole, S. Esser, T. Delbruck, M. Flickner, and D. Modha, "A low power, fully event-based gesture recognition system", in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 7388–7397.

[52] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations", in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., Curran Associates, Inc., 2018, pp. 6571–6583. [Online]. Available: http://papers.nips.cc/paper/7892-neural-ordinary-differential-equations.pdf.

[53] S. Lacey and K. Sathian, “Crossmodal and multisensory interactions between vision and touch”, *Scholarpedia*, vol. 10, no. 3, p. 7957, 2015, revision #150498.

[54] P. Allen, “Surface descriptions from vision and touch”, in *Proceedings. 1984 IEEE International Conference on Robotics and Automation*, - nil, nil. [Online]. Available: `https://doi.org/10.1109/robot.1984.1087191`.

[55] J. Sinapov, C. Schenck, and A. Stoytchev, “Learning relational object categories using behavioral exploration and multimodal perception”, in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, nil. [Online]. Available: `https://doi.org/10.1109/icra.2014.6907696`.

[56] Y. Gao, L. A. Hendricks, K. J. Kuchenbecker, and T. Darrell, “Deep learning for tactile understanding from visual and haptic data”, in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2016, pp. 536–543.

[57] J. Li, S. Dong, and E. Adelson, “Slip detection with combined tactile and visual information”, in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, nil. [Online]. Available: `https://doi.org/10.1109/icra.2018.8460495`.

[58] M. A. Lee, Y. Zhu, K. Srinivasan, P. Shah, S. Savarese, L. Fei-Fei, A. Garg, and J. Bohg, “Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks”, in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 8943–8950.

[59] S. Luo, J. Bimbo, R. Dahiya, and H. Liu, “Robotic tactile perception of object properties: A review”, *Mechatronics*, vol. 48, pp. 54–67, 2017.

[60] H. Liu, Y. Yu, F. Sun, and J. Gu, “Visual–tactile fusion for object recognition”, *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 2, pp. 996–1008, 2016.

[61] J. Varley, D. Watkins, and P. Allen, “Visual-tactile geometric reasoning”, in *RSS Workshop*, 2017.

[62] J. Reinecke, A. Dietrich, F. Schmidt, and M. Chalon, "Experimental comparison of slip detection strategies by tactile sensing with the biotac® on the dlr hand arm system", in *2014 IEEE international Conference on Robotics and Automation (ICRA)*, IEEE, 2014, pp. 2742–2748.

[63] R. Calandra, A. Owens, D. Jayaraman, J. Lin, W. Yuan, J. Malik, E. H. Adelson, and S. Levine, "More than a feeling: Learning to grasp and regrasp using vision and touch", *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3300–3307, 2018.

[64] Z. Su, K. Hausman, Y. Chebotar, A. Molchanov, G. E. Loeb, G. S. Sukhatme, and S. Schaal, "Force estimation and slip detection/classification for grip control using a biomimetic tactile sensor", in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, IEEE, 2015, pp. 297–303.

[65] W. Yuan, S. Dong, and E. H. Adelson, "Gelsight: High-resolution robot tactile sensors for estimating geometry and force", *Sensors*, vol. 17, no. 12, p. 2762, 2017.

[66] G. Gallego, T. Delbr, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, K. Daniilidis, D. Scaramuzza, S. Leutenegger, and A. Davison, "Event-based Vision : A Survey", Tech. Rep., 2018, pp. 1–25. arXiv: `arXiv:1904.08405v1`.

[67] A. Mitrokhin, C. Ye, C. Fermuller, Y. Aloimonos, and T. Delbruck, "EV-IMO: Motion Segmentation Dataset and Learning Pipeline for Event Cameras", in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[68] A. Z. Zhu and L. Yuan, "EV-FlowNet: Self-Supervised Optical Flow Estimation for Event-based Cameras", in *Robotics: Science and Systems*, 2018.

[69] A. I. Maqueda, A. Loquercio, G. Gallego, N. Garc\'\ia, and D. Scaramuzza, "Event-based vision meets deep learning on steering prediction for self-driving cars", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5419–5427.

[70] S. Chevallier, H. Paugam-Moisy, and F. Lemaıtre, "Distributed processing for modelling real-time multimodal perception in a virtual robot.", in *Parallel and Distributed Computing and Networks*, 2005, pp. 393–398.

[71] N. Rathi and K. Roy, “Stdp-based unsupervised multimodal learning with cross-modal processing in spiking neural network”, *IEEE Transactions on Emerging Topics in Computational Intelligence*, pp. 1–11, 2018, ISSN: 2471-285X.

[72] E. Mansouri-Benssassi and J. Ye, “Speech emotion recognition with early visual cross-modal enhancement using spiking neural networks”, in *2019 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2019, pp. 1–8.

[73] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder–decoder for statistical machine translation”, in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1724–1734.

[74] R. Florian, “A reinforcement learning algorithm for spiking neural networks”, vol. 2005, Oct. 2005, 8 pp.-, ISBN: 0-7695-2453-2.

[75] R. V. Florian, “Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity”, *Neural Computation*, vol. 19, no. 6, pp. 1468–1502, 2007. [Online]. Available: https://doi.org/10.1162/neco.2007.19.6.1468.

[76] D. Baras and R. Meir, “Reinforcement learning, spike-time-dependent plasticity, and the bcm rule”, *Neural Computation*, vol. 19, no. 8, pp. 2245–2279, 2007. [Online]. Available: https://doi.org/10.1162/neco.2007.19.8.2245.

[77] W. Potjans, A. Morrison, and M. Diesmann, “A spiking neural network model of an actor-critic learning agent”, *Neural Computation*, vol. 21, no. 2, pp. 301–339, 2009. [Online]. Available: https://doi.org/10.1162/neco.2008.08-07-593.

[78] A. Vitanza, L. Patané, and P. Arena, “Spiking neural controllers in multi-agent competitive systems for adaptive targeted motor learning”, *Journal of the Franklin Institute*, vol. 352, no. 8, pp. 3122–3143, 2015, Special Issue on Advances in Nonlinear Dynamics and Control, ISSN: 0016-0032. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S001600321500174X.

[79] S. Aenugu, A. Sharma, S. Yelamarthi, H. Hazan, P. S. Thomas, and R. Kozma, "Reinforcement learning with spiking coagents", *CoRR*, 2019. arXiv: 1910.06489 [cs.LG]. [Online]. Available: http://arxiv.org/abs/1910.06489v2.

[80] T. Miconi, A. Rawal, J. Clune, and K. O. Stanley, "Backpropamine: Meta-training self-modifying neural networks with gradient descent", in *Workshop on Neural Information Processing Systems*, 2018.

[81] OpenAI, *Openai gym*, https://gym.openai.com/envs/CartPole-v0/, Online; accessed 02 November 2019, 2019.

[82] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning", *arXiv preprint arXiv:1312.5602*, 2013.

[83] D. Heeger, "Poisson model of spike generation", *Handout, University of Standford*, vol. 5, pp. 1–13, 2000.

[84] J. Schulman, "Optimizing expectations: From deep reinforcement learning to stochastic computation graphs", PhD thesis, UC Berkeley, 2016.

[85] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation", *arXiv preprint arXiv:1506.02438*, 2015.

Appendix A

Hyperparameters

A.1 Neuron Model

The primary neuron model we chose was the spike-response model section 2.4. The neuron has several hyperparameters, which we hold fixed throughout our experiments (see Table A.1).

A.2 Loss Parameters

Little was done to tune the neuron model parameters, because we found that the performance of the network was much more sensitive to the loss parameters. In particular, the loss computes the deviation of the spike counts from the expected spike counts of the output neurons. There is a fine balance for these parameters; on one hand, it is desirable to set low spike counts, to encourage the model to communicate the same information with fewer spikes. How often a neuron can spike is also related to hyperparameters in the neuron model: the membrane threshold and refractory response most directly affect the value. We chose the parameters to be as low as possible, without significant degradation in performance (see Table A.2).

Parameter	Value
θ	10
τ_{sr}	10.0
τ_{ref}	1.0
$\text{scale}_{\text{ref}}$	2
τ_{ρ}	1
scale_{ρ}	1

Table A.1: Hyperparameters for the neuron model (spike-response model).

Parameter	Container & Weight Classification	Rotational Slip Classification
True spike count	150	80
False spike count	5	5

Table A.2: Hyperparameters for the spiking loss.

Appendix B

Reinforcement Learning in Spiking Neural Networks

There is immense value from the computational neuroscience perspective, in developing neural agents that are able to learn by interacting with the environment. These agents develop their own internal representations of the environment through unsupervised learning or reinforcement learning. Local learning rules (see section 2.6) like the Hebbian plasticity rule conduct such unsupervised learning, but these rules do not maximize a global reward, and cannot model goal-oriented agents.

Recent advancements in training of spiking neural networks, in particular surrogate gradient methods (see section 2.6.1), have opened up gradient-based Reinforcement Learning methods to spiking neural networks. It is our belief that reinforcement learning is especially attractive for spiking neural networks, for the following reasons:

1. Spiking neural networks bear a strong resemblance with biological neural networks, allowing us to draw inspiration from biological neurons and how they interact with the environment

2. Many reinforcement learning environments are temporal in nature: the agent picks receives an observation, and performs an action at each timestep, affecting the environment. SNNs incorporate time into their operating model, and are well suited as agents.

3. Reinforcement Learning is a commonly used paradigm for robotic applications, and neuromorphic hardware can be demonstrated to use significantly less power (see chapter 8).

In this chapter, we evaluate the feasibility of SNNs in simple Reinforcement Learning environments. We wish to see if gradient-based learning methods work with spiking neural agents.

B.1 Prior Work

The exploration of reinforcement learning in spiking neural networks has largely been limited to the exploration of its relations to plasticity-based methods such as the spike-timing-dependent-plasticity (STDP) rule. The reward-modulated STDP rule was derived using reinforcement learning for spiking neurons that fire stochastically [74]. The modulation of STDP by a global reward signal leads to reinforcement learning [75], [76]. These local update rules have been used to train spiking neural agents for robotic tasks, such as in target-reaching vehicles [33]. In a similar fashion, spiking neural models have been shown to be able to implement the actor-critic temporal-difference learning algorithm by combining local Hebbian plasticity rules with a global reward signal [77]. Where reinforcement learning tasks are concerned, these tasks are often solved in a multi-agent setting [78], [79]. Frameworks such as BindsNET [32] implement these learning rules, but have poor performance in reinforcement learning environments [1].

Some works take inspiration from neuroscience, applying these concepts to boost the performance of deep learning models. For example, introducing differentiable plasticity rules to LSTMs have improved their performance on benchmark language modelling tasks [80]. We are however more interested in applying advancements in deep learning to spiking neural networks.

Beyond STDP learning rules, ANNs have been trained and converted into SNNs using gradient-based reinforcement learning techniques such as Deep-Q learning, and demonstrated competitive results in ATARI game environments with observations converted into binary inputs [2].

In this work, we consider a single-agent setting, and explore the training of gradient-based spiking neural architectures *end-to-end* using recently developed surrogate gradient methods,

[1] See `https://github.com/BindsNET/bindsnet/issues/345`

[2] Paper by anonymous authors, found at `https://bit.ly/2UVNpQT`

tapping on the wealth of research on gradient-dependent reinforcement learning methods.

B.2 The ImageCartpole Environment

In our experiments, we use a modified version of the Cartpole-v0 environment by OpenAI. The Cartpole-v0 environment is described as follows [81]:

> A pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The system is controlled by applying a force of +1 or -1 to the cart. The pendulum starts upright, and the goal is to prevent it from falling over. A reward of +1 is provided for every time-step that the pole remains upright. The episode ends when the pole is more than 15 degrees from vertical, or the cart moves more than 2.4 units from the center.
>
> CartPole-v0 defines "solving" as getting average reward of 195.0 over 100 consecutive trials.

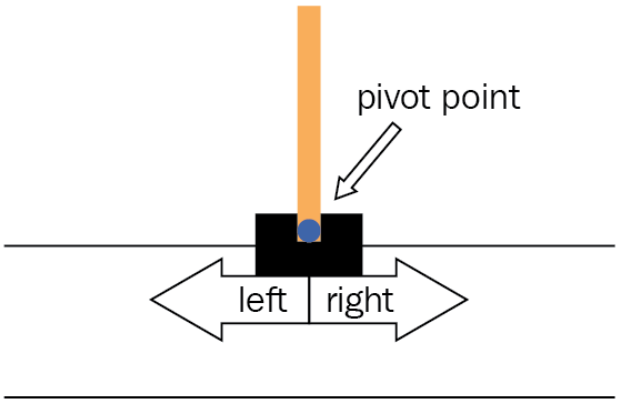


Figure B.1: The OpenAI CartPole environment

The observations in Cartpole-v0 are continuous, real-numbered values, two of which have infinite domain. This poses a challenge for conversion into spike trains. To overcome this issue, we pass as the observation pixels of the rendered environment. Inspired by [82], we use the following processing steps:

First, the observation values are rendered into a colored cartpole, with variable size using OpenAI's environment visualization. A 70×90 region that's centered at the CartPole is cropped for each observation. This cropped region is grayscaled, and has its colour inverted such that the white background has values 0, and the region of interest takes positive values. Each pixel is converted into a spike train by modelling them as a Poisson process [83]. The process is as follows. First, we divide time into short, discrete intervals δt, and generate a sequence of random numbers $x[i]$ uniformly between 0 and 1. For each interval, if $x[i] \leq r\delta t$, generate a spike. We produce spike trains of 300 time-steps.

The agents are presented with $N = 4$ image frames at each time-step, with the previous frame repeated when there are not enough frames (at the beginning of the episode). We find that the agents were unable to learn a good policy with smaller N.

B.3 The Models

The models we employed in this experiment are simple MLPs. While the input data are images, MLPs were sufficient in solving the ImageCartPole environment. The agents takes in the image frames as input, and outputs as logits the probability of each action. The ANN outputs the logits directly, while the SNN dense output layer produces spike counts (similar to section 5.3) for the supervised task, which are later normalized into logits. The agents sample from a 2-outcome categorical distribution with the logits as the outcome probabilities.

B.4 Model Training

We train both models using the vanilla policy gradients (VPG) algorithm [84], with generalized advantage estimation (GAE) [85]. We implement this in PyTorch[3], and train our agents for 500 epochs.

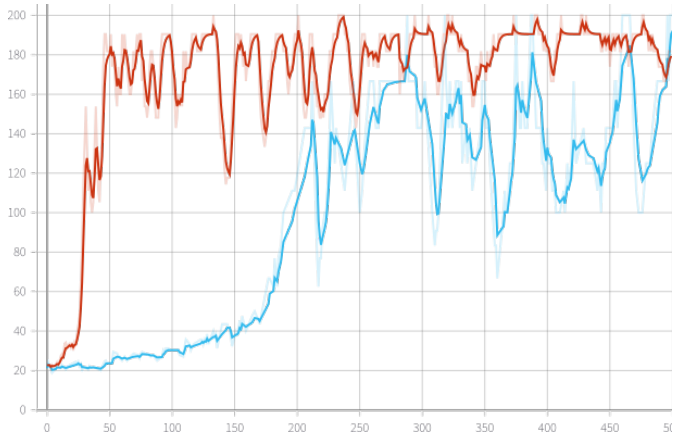


Figure B.2: Mean episodic return of the two agents. The ANN agent is depicted in red, while the SNN agent is depicted in blue. The ANN agent solves the task quicker than the SNN.

B.5 Results

Our experimental results (see Figure B.2) show that both the ANN and SNN agents are able to solve the ImageCartpole Environment. The ANN agent is able to solve the ImageCartPole environment within 50 epochs, while the SNN agent takes 300 epochs to obtain a solution. However, this suggests that Reinforcement Learning is still a viable method for learning spiking neural agents.

B.6 Discussion

The results we obtained demonstrated that spiking neural agents can be directly trained end-to-end on simple reinforcement learning tasks. While the results were positive, the current environment setup does not fully exploit the benefit of spiking neural networks, and it is difficult to motivate the use of spiking neural networks in this context. At each time-step, observations are converted to full spike trains. SNNs are often able to make predictions based on early spikes, albeit less reliable. This is not exploited in the current formulation, where a full spike train of fixed length is received before an action is taken.

One can imagine a RL environment with the following satisfied:

[3] `https://github.com/jethrokuan/snnrl/`

1. The action space should contain a no-op: in the case of the Cartpole environment, this is as simple as adding the action of not applying force to the cart, instead of choosing between left or right.

2. At each observation, the observation should be spikes, rather than spike trains.

The SNN agent receives spikes at each time-step, and after gaining enough confidence to take an action, it takes the action at that time-step. SNN agents may be able to solve such environments by exploiting the temporal information encoded in the relative timing between spikes. We found it difficult to reconcile the temporal domain of the spiking neural network, and the RL environment to define a gradient-based learning algorithm that performs the appropriate credit-assignment, and hence decided to pause on this front.