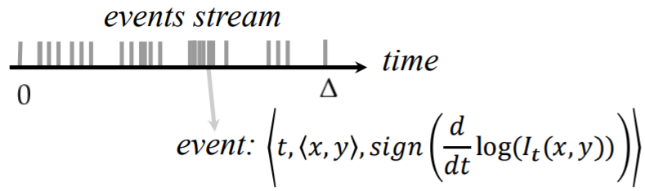
# 1 DYNAMIC VISION SENSORS

Current camera systems are frame-based. This requires the sampling of all pixels at regular time intervals. However, for motion-based applications this is incredibly inefficient. Firstly, each frame contains a large amount of redundant information as often many pixels are unchanged between frames. Secondly, frames are transferred synchronously. This has two implications: 1) bandwidth acts as a bottleneck, limiting the sampling rate (fps) and resulting in a large amount of ‘lost’ information between frames; 2) frames pass through processing stages sequentially, further reducing the achievable sampling rate (fig. 1). This effect is exacerbated at higher resolutions due to higher computational load.

Dynamic Visions Sensors (DVS) offer a solution by departing from the traditional frame-based paradigm. These systems are inspired by the sparse event-based output of biological vision systems (1). Instead of the repetitive, uniform sampling of the visual environment, DVS capture only the changes, dramatically reducing the quantity of redundant information. Interestingly, this is similar to the method used for video compression (2). Pixels respond instantaneously to intensity changes, sending a spike to signify this (equation 1). Crucially, this information transfer is asynchronous and rapid – on the order of microseconds. This allows for effectively continuous sampling of the visual environment. Furthermore, captured images are inherently robust to different lighting conditions since the pixels encode changes rather than absolute values. Changes are conveyed with address-event representation (AER), a communication protocol in which the event (luminance change ON/OFF) is sent instantaneously along with its address (pixel location). The dynamic vision sensor used in this project is the DVS240C, produced by Ini Labs (1).



(1)

Conventional image-processing algorithms operate on frames and are therefore incompatible with DVS. This may be naively rectified by taking a time-window of events to create a ‘frame’, which is then operated on as usual. This is far from ideal, as the algorithm must wait for events to be ‘collected’, resulting in a high latency. This is further compounded by the previously mentioned sequential nature of frame-based image processing (fig. 1). Convolutional architectures in particular have high computational cost, thus forcing the system to run at low fps. This does not take advantage of the key feature of the DVS: rapid, asynchronous sampling of visual space.

On the other hand, an alternative approach is to develop algorithms capable of operating on events immediately as they arrive, thereby taking full advantage of the DVS. This literature review will focus on biologically inspired algorithms known as SNNs, which are capable of asynchronously processing event-based data.

|  |
| --- |
| Figure 1: Comparison between frame-based (middle) and event-based (bottom) image processing. Colour-coded blocks/dots represent outputs from respective stages of processing. (3) |
|  |

# 2 BIOLOGICAL BACKGROUND

## 2.1 Introduction

There are three reasons for which the mammalian visual system is a good source of inspiration and highly relevant to this project. First, the human visual system is a massively parallel asynchronous system, capable of extraordinarily rapid classification, with delays as low as 100ms (4). Second, the underlying principles of the DVS closely mirror those of the retina - the primary input to the Visual Cortex. Third, the incredible success of existing convolutional neural networks (CNN) which themselves are inspired by the Visual Cortex.

In this section the hierarchical architecture, the base unit of communication (spikes) and learning mechanisms of the Visual Cortex will be discussed.

## 2.2 Cortical Anatomy

N.B still learning about this

Basic architecture/operations of cortical columns etc

## 2.4 Spikes

Spikes are the basic unit of communication between neurons, via which information is encoded. How this information is encoded however, is still debated. Traditionally, rate-based encoding was assumed to be the primary method of encoding data. However, in order to compute a rate, spikes must be counted over a prolonged period of time, leading to high latency (5). The incredible speed of response in the visual system suggests that spike timing is crucial. Indeed, after subtracting the time of crossing consecutive synapses during visual inference, Guyonneau et al. estimate a total processing time of 10ms per stage. Given that neurons rarely fire at a rate exceeding 100Hz, at most one spike is expected per 10ms window (4). This suggests that the first spikes generated by a stimulus are sufficient. Furthermore, the extreme time constraints imply that rapid visual processing is performed in a feedforward manner (6).

## 2.3 Learning Mechanisms

N.B. STDP covered in engineering context in section 3.3.1.

Topics still to read about:

* Biological background of STDP
* Competitive firing & Lateral inhibition (WTA)

Papers still to read: (4,7-12)

# 3 SPIKING NEURAL NETWORKS

Spiking Neural Networks (SNN) have been described as the third generation of Neural Networks (13). While conventional Artificial Neural Networks (ANN) are inspired only by biological network structure, SNNs are an attempt to also incorporate the temporal mechanics observed in biology. As mentioned in section 2, spike timing is believed to be integral to rapid neural computation, particularly in the visual system. ANNs are typically operated on synchronous systems, however the temporal cost of this was described in section 1. In section 5 methods of implementing SNNs asynchronously will be described.

The basic unit of communication in both SNNs and biological neural networks is the spike (in biology this is an action potential). For simplicity in SNNs, the spike is often modelled as a Dirac-delta function. This is based on the assumption that spike is timing more important than shape (7). Alternatively, spikes can be modelled as continuous-time functions however this increases computational complexity (14).

While each node of an ANN typically performs summation followed by a sigmoidal function, SNNs instead use neuron models to process input signals. These are discussed in section 3.1 in the context of real-time event-based processing.

There are two approaches to learning: Supervised and Unsupervised. State-of-the-art approaches to these for SNNs are discussed in sections 3.2 and 3.3.

## 3.1 Spiking Neuron Models

|  |
| --- |
|  |
| Figure 2: Presynaptic spike trains and current (top), membrane potential (middle), output spike train (bottom) for LIF neuron. Upon reaching the threshold, membrane potential resets and remains unchanged for the refractory period. (7)  (2) |

There are two commonly used models in SNNs: integrate-and-fire (IF) and leaky integrate-and-fire (LIF). Both model the neuron as a point system. In essence, the modelled neuron integrates input current, increasing its membrane potential. If the membrane potential reaches a threshold, a spike is emitted, and the neuron returns to its reset potential. A refractory period may also be implemented.

The LIF differs from IF in that membrane potential decreases exponentially over time when no input current is applied. This gives temporal significance to inputs and allows neurons to ‘forget’ previous inputs (15). As such LIF neurons are generally favoured in SNNs for temporal tasks.

The dynamics of the LIF neuron is given by equation 2 (7):



Where C is membrane capacitance, u is the membrane potential, R the input resistance, io(t) the external current, ij(t) the input current from the jth synapse, and wj the weight of the jth synapse.

Figure 2 shows the effect of presynaptic firing on a neuron’s membrane potential and output spike train.

Still to read: Gerstner & Kistler 2002 paper about neuron models.

## 3.2 Supervised Learning in SNNs

Conventional multi-layered Neural Networks require the use of back-propagation to tune weights. There are two fundamental problems in applying this algorithm to train SNNs (16). Firstly, the backpropagation algorithm requires the derivative of the activation function with respect to synaptic weights. For spiking neurons, this is often represented by a series of delta functions and is therefore non-differentiable. Secondly, backpropagation requires feedforward and feedback weights to be exactly matched. This is not biologically plausible, however simple to implement in practice. There have been many attempts to circumvent these issues. Three successful approaches will be discussed: ANN-to-SNN conversion, ReSuMe, and the backpropagation algorithm recently proposed by Lee, Delbruck and Pfeiffer (17).

### 3.2.1 ANN-to-SNN conversion

A method of training SNNs is to first train an ANN before mapping the weights to a spiking network. This is frequently done via rate-based coding: spike rates are analogous to the floating-point activation values in an ANN (17). However, as mentioned in section 2.4 it is the timings of spikes rather than rate that allows for rapid processing in the visual system. Furthermore, information in an event-based stream is not encoded in rate, but rather in the presence and timing of an event. For these reasons training an ANN before conversion into an SNN leads to suboptimal results.

Stromatias et al. (15) propose an alternative approach. Instead of training an entire SNN using an ANN, the method trains a two-stage network. This two-stage network is composed of an unsupervised SNN for feature extraction (as in section 4.1), followed by a supervised SNN for classification. The weights for the classification network are learnt by an ANN on the features from the previous stage. This avoids the problem of needing to convert event data into rates.

The input to the ANN during training is a histogram which contains the counts for each neuron on the outer layer of the SNN upon presenting a training sample. The learnt weights are then mapped onto a network containing LIF neurons. The implementation and performance of the method are discussed in section 4.4.1.

### 3.2.2 ReSuMe

ReSuMe (18) is a popular algorithm that may be used to train a neuron to reproduce a desired input-output spike pattern. Several existing algorithms utilise gradient-based learning, however in SNNs this frequently requires many assumptions and restrictions (19). However, ReSuMe is suitable only for single layer training. An approach for multi-layer training is discussed in section 3.2.3.

|  |
| --- |
|  |
| Figure 3: Successive training of a network using the ReSuMe algorithm. A network containing 10 LIF neurons and 500 inputs is given a desired spiking output, shown in grey. Note that after successive training epochs the neurons begin to match the desired output (7). |

The ReSuMe algorithm involves imposing a reference (teacher) signal on a neuron. The aim is to train the neuron to reproduce the reference signal for a given stimulus. Weights are updated according to equation 3 (7):

(3)

S

Where a is the learning rate, Sd(t) the reference spike train, Sj(t) the output spike train, and Si(t) the low-pass filtered input spike train. The learning process is shown graphically in fig. 4. Temporal correlation between input and target spikes results in synaptic potentiation. Conversely, temporal correlation between input and output spikes, in the absence of a reference spike, causes depression. Intuitively, if an output spike is desired but not observed, the synaptic weight is increased in an attempt to produce these spikes in future. The synaptic weight eventually converges when actual output consistently matches desired output. Conversely, if an output spike is observed when not desired, synaptic weight is decreased in an attempt to counteract this. The magnitude of the weight update depends on the latency of response.

The results of training using ReSuMe are shown in fig 3. 15 training epochs are sufficient for the output and reference spike trains to be matched.

### 3.2.3 Backpropagation for SNNs

N.B I’m not very confident with the maths of this study yet and would like to read it more and/or discuss with you.

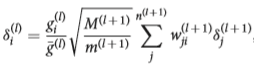
As mentioned previously, backpropagation cannot be applied directly to SNNs due to the non-differentiable nature of spikes. Lee, Delbruck and Pfeiffer (17) utilise the continuous membrane potential of LIF neurons to train a multi-layered SNN with Stochastic Gradient Descent (SGD). Refer to section 4 for implementation details.

Previous supervised methods for training SNNs require time-windowing of event-data, meaning that training and test data are intrinsically different. The proposed technique is highly relevant as it allows for the network to be trained on data of the same format as real-life data. Furthermore, the technique is shown to achieve a high accuracy on event-based data (section 4.4.2).



(4)

Where ∆wij(l) is the weight update for the jth synapse to the ith neuron, ∆Vth,i(l) is the threshold update for the ith neuron, nw and nth are the respective learning rates, N and M the number of neurons in the l-th and (l-1)th layers, n and m the number of synapses. δi(l) is calculated using equation 5:



(5)

Where gi(l) = 1/Vth,i(l)…

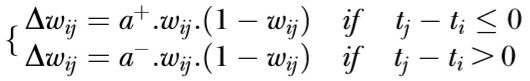
|  |
| --- |
|  |
| Figure 4: Graphical illustration of different cases for learning in ReSuMe algorithm. Si  is the input spike train, Sd is the reference spike train and So is the output spike train. |

## 3.3 Unsupervised Learning in SNNs

Unsupervised approaches in SNNs are generally used to extract features from an image or event stream. As described in section 4, this is typically followed by a classifier such as SVM.

### 3.3.1 STDP

Due to the computational cost of employing the standard STDP rule, simulations frequently utilise a simplified rule. This frequently involves the independence of synaptic weight updates with respect to time (2,6,12). Masquelier and Thorpe (12), and Kheradpisheh et al (6) utilise a simplified STDP rule:

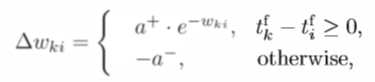


(6)

Where wij is the synaptic weight between presynaptic and postsynaptic neurons, a+ and a- are the learning rates, and tj and ti are the firing times of the postsynaptic and presynaptic neurons. Note that the wij(1-wij) term ensures wij remains in the range [0,1]. An infinite time window is used as each neuron sends only one spike and stimuli are presented one at a time. This would be problematic in applications involving a temporal component.

|  |  |
| --- | --- |
|  |  |
| (a) | (b) |
| Figure 5: (a) Standard GA. (b) Improved GA proposed by (20) | |

Tavanaei, Masquelier and Maida (21) show that a probabilistic interpretation of STDP improves performance of algorithms that apply the STDP rule given in equation 6.

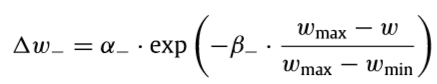
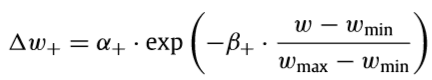


(7)

Where a+ and a- are learning rates, k and i the post and presynaptic neurons respectively, wki the synaptic weight between these, and tkf and tif the corresponding firing times. Note that in equations 6 and 7 the ratio a+/a- is generally set as 4/3.

In (2) a modification is made to the criteria for LDP. While in standard STDP a temporally uncorrelated postsynaptic potential results in LDP, in this method neurons that do not fire also undergo LDP. Why? The rule is given by equation 8.

(8)



Where w+ and w-­ are the synaptic weight changes during LTP and LDP respectively. Four learning parameters are set: a+ > 0, β+ ≥ 0, a- < 0 and β- ≥ 0. Synaptic weight, w, is kept in the range wmin > 0 and wmax. Note that a finite time window is used.

STDP has been applied successfully to learn features from both static images and event-based footage. Two widely-cited studies using STDP in a practical context are discussed in sections 4.1 and 4.3.

## 3.4 Genetic Algorithms to tune SNNs

Genetic algorithms (GA) provide an effective method to tune parameters and topology of neural networks (20,22). For SNNs in particular, GA may be used to tune model parameters such as refractory period, threshold for spiking, leak time-constant, time window for STDP, and inhibitory period (2).

In GA, a population, X, contains a set of individuals containing chromosomes (a set of parameters to be tuned). A chromosome **p** for a single individual takes the form:

Where pi is a parameter and n is the number of parameters. Parameter value pi is generally kept between [pi,min, pi,max]. The population is updated incrementally according to three operators: reproduction, mutation and crossover (22). These involve random individual selection, random chromosome changes, and mixing of selected chromosomes respectively. During reproduction, the probability of choosing an individual is frequently proportional to their fitness *f.* Fitness is a user-defined function to measure an individual’s performance.

The algorithmic processes for standard GA and an improved approach (20) are shown in figure 5. Refer to (20) an in-depth summary of the algorithms and operations.

# 4 Previous application of SNNs to Visual Processing & Pattern Recognition Tasks

## 4.1 AER car tracking

Bichler et al (2) utilise an unsupervised SNN to track cars from AER camera footage with resolution 128x128. The network is 2-layer feedforward (fig. 6 a, b). *Both* layers are trained using STDP (i.e. simple features are not prescribed) as shown in equation 8, while parameters are tuned with Genetic Evolution. 10 minutes of training footage was required to achieve an accuracy of 98%. This was obtained in real-time while running on a CPU using XNet.

The proposed architecture (fig. 6 (a)) consists of 3 layers: 16,384 spiking pixels, 60 fully connected neurons, and 10 fully connected neurons. This equates to 1,966,680 synapses. Each neuron laterally inhibits all other neurons in the same layer. A second proposed architecture (fig. 6 (b)) reduces the RF of each first-layer neuron to a patch of 16x16 pixels. Four neurons are assigned to each patch. This reduces synapse count to 131,712 and was found to have little effect on performance (98% compared to 95% detection rate) while drastically reducing processing time. It was also found that training each layer sequentially, as opposed to simultaneously, improved performance.

The study utilises a simplified STDP rule which is independent of time. Unlike in standard STDP, LDP is performed on all postsynaptic neurons which do not fire in a prescribed time interval tLP, including even those that did not fire. This improves selectivity in latter layers. The synaptic weights are constrained between 0 and wmax to ensure excitatory stimulation. The spiking neuron model used is LIF.

Two key learning mechanisms are employed: lateral inhibition and refractory period. Lateral inhibition is crucial to prevent neurons from learning the same features. Lateral Inhibition is disabled after learning as it may inhibit neighbouring neurons with temporally overlapping features. A refractory period is necessary in order to prevent a single neuron from acting ‘greedily’, by repeatedly adapting its RF and firing as the stimulus varies, while inhibiting other neurons.

|  |  |
| --- | --- |
|  |  |
| (a) | (b) |
| Figure 6: Network architectures utilised in (2) for unsupervised feature extraction from AER data (a) Fully connected structure (b) Reduced structure with 10-fold decrease in number of synapses. | |

A genetic algorithm is used to tune the five shared parameters (Ithres, TLTP, Trefrac, Tinhibit, τleak) of each layer. Each generation has 80 simulated networks, whose activity is compared with reference activity obtained from hand labelling. The best 8 networks in a generation are selected and mutated. Ten generations was sufficient to achieve good parameters.

## 4.2 AER Gesture Recognition

Lee et al (23) utilise an asynchronous layer of LIF neurons to detect hand movements from a stereo pair of AER cameras with resolution 128x128. However, the method relies on the assumption that the body is stationary and hand movements are the primary source of events. No learning is performed.

LIF neurons select for events occurring simultaneously within their RF. In this way they are able to detect ‘active’ regions. The RFs of neighbouring neurons overlap to create spatial correlation. Simultaneously firing neighbouring neurons are clustered and the centre located with a weighted average.

The two camera images are aligned in such a way to ensure the hands overlap. Events occurring at different depths (noise) are therefore not aligned and do not activate LIF neurons. LIF threshold is adaptively tuned based distance, to counteract its effect on the size of visual motion and therefore the number of events generated.

## 4.3 3D Object classification

Kheradpisheh et al (6) utilise an unsupervised feedforward SNN for feature extraction, heavily inspired by the visual cortex and a previous model proposed by Masquelier and Thorpe (12). The features are used along with a SVM to classify 3D objects. Position and scale invariance is built-in, while invariance to viewing angle is learnt. An accuracy of 96% is obtained, outperforming supervised algorithms such as DeepConvNet (reference) and HMAX (reference). Notably, far fewer images were required for training than the DeepConvNet.

The architecture consists of 4 layers: S1, C1, S2, C2 (fig. 7). Successive layers identify more complex features. Retinotopy is maintained until the C2 layer. Learning is performed only in the S2 layer, via simplified STDP. S1 cells perform edge detection in one of four orientations, with the first to spike inhibiting its neighbours. The latency of an S1 spike is the inverse of the convolution value. C1 cells perform max pooling resulting in shift invariance. S2 perform learnt feature extraction, while C2 select the global maximum. IF neurons are used.

Simplified STDP, independent of time, is employed (equation 6). Spikes within a time window cause LTP, whereas others cause LDP. Synaptic weights are kept in the range [0,1] to ensure excitatory stimulation. Learning is slowed down over time via a learning rate. Furthermore, each cell sends only a single spike. This is based on the assumption that for the rapid visual processing observed in the visual cortex, the first spike is the most important (4).

Weight sharing is a key technique used to achieve invariance to scale and position. As may be seen in figure X, multiple identical S1/C1/S2 maps are each trained on a different scale image. Each of the duplicated S1 maps has the same Gabor filter wavelength. This process has the same effect as having Gabor filters of varying wavelengths in the S1 layer. Crucially, the S2 maps at different scales compete to fire first. The learnt weights are then shared at all positions and scales, to ensure frequently occurring patterns are learnt independently of these.

## 4.4 N-MNIST digit recognition

The N-MNIST dataset (24) is a neuromorphic version of the MNIST dataset. Recordings were made using a DVS camera, which tilts three times per input image, generating a 300ms video. Each saccade lasts 100ms. Crucially, events are generated due to motion, as opposed to previous neuromorphic datasets which use Poisson processes to generate spikes based on greyscale intensity.

|  |
| --- |
|  |
| Figure 7: Architecture used in (6) for unsupervised training to perform 3D-object recognition. |

### 4.4.1 ANN-SNN Conversion for N-MNIST

Stromatias et al. (15) utilise the ANN-SNN conversion method described in section 3.2.1. A classification accuracy of 97.77% is reported on the N-MNIST dataset.

The study utilises an unsupervised SNN for feature extraction similar to that described in section 4.1. Crucially, the study utilises LIF neurons and is therefore applicable to tasks with a temporal component.

Training was performed in batches of 500 samples. Mini-batch stochastic gradient descent without biases was used to train the ANN. A softmax activation function was used. The cost function used was the negative log-likelihood:



(9)

Where C is the Cost function, D is the mini-batch, K is the set of classes, and Ii(j) = 1 for j = Li (target output) and 0 otherwise. Y­j the output of unit j, Xi is the input and W is the matrix of weights. A fixed learning rate was used.

After training, weights are scaled by a constant which is on the order of Vth. For details of parameters used refer to (15).

It is interesting to note that the study reports far higher latencies on event data as opposed to synthetic neuromorphic data (i.e. intensity-rate conversion using Poisson processes). The latency – time delay between the first input and output spikes - for event data was on the order of milliseconds, while for other neuromorphic data latency was on the order of microseconds. This is most likely due to the sparsity of events in real-life event data.

### 4.4.1 Backpropagation for N-MNIST

Lee, Delbruck and Pfeiffer (17) utilise the supervised learning method described in section 3.1.3 to train on N-MNIST recordings. A state-of-the-art accuracy of 98.66% was reported using a feedforward SNN with 800 hundred hidden units. No pre-processing was required – training and testing were performed directly on event-data.

An LIF neuron is used as the basic unit for the network. Membrane potential is updated asynchronously when spiking occurs. A dynamic weight wdyn is used to model the refractory period, as shown in equations 9 & 10.

(9)



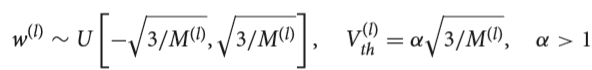
(10)



Where Vmp(tp) is the membrane potential at timepoint tp, at which a spike arrives, wi(p) is the synaptic weight through which the spike arrives, τmp the time-constant, and wdyn the dynamic weight given by equation 10. ∆t is the time difference tout – tp where tout is the time of an output spike or incoming lateral inhibition spike.

The network utilises winner-take-all (WTA) circuits; each neuron laterally inhibits all others upon firing. The effect of lateral inhibition on a neuron’s membrane potential is proportional to its threshold voltage, as seen by the final term in equation X.

Weights and threshold for the lth layer, w(l) and Vth(l), are initialised as follows (17):

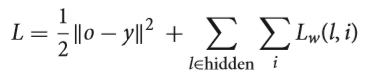
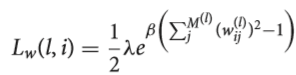


(11)

Where U[-a,a] is a uniform distribution between -a and a, M(l) is the number of synapses of each neuron, and α is a user-defined parameter (a value between 3 and 10 was found to be sufficient).

Training is performed by updating weights and thresholds using the method described in section 3.1.3. For each training sample (300ms input recording) spikes are accumulated and counted for each output neuron. The total number of spikes for each neuron is normalised and represented in a vector o:

Where n is the number of neurons in the output layer (classes), and #spikesi the number of accumulated output spikes for neuron i. Weight regularisation is performed according to the cost function in equation 12. Threshold regularisation is required to prevent ‘dead’ neurons that do not respond to stimuli. This is performed by increasing the thresholds of firing neurons, while decreasing those of non-firing neurons. The objective function to be minimised during training is shown in equation 13.



(13)

(12)

It is interesting to note that the accuracy of the classifier during testing is highly dependent on the event rate. Fig. 8 shows that error decreases due to each successive saccade.

|  |
| --- |
|  |
| Figure 8: A) Event rate for single N-MNIST sample. B) Classification error of algorithm on sample. Note that large decreases are observed during periods of high event rate. |

The primary drawback of this method is that spike statistics (times of all input and output spikes for each neuron) must be recorded during runtime. However, this is only required during training. The authors note that a platform such as SpiNNaker may suitable for this algorithm.

# 5 Implementation

SNNs are difficult to implement as standard computers are von Neumann machines, performing operations sequentially and relying on a central processing unit and shared memory. Neurons meanwhile, are massively parallel and SNNs frequently utilise AER. SpiNNaker is a parallel platform designed specifically for simulating large numbers of neurons and synapses in real time.

## SpiNNaker (25,26)

A Spinnaker chip is composed of multiple processing cores, each with their own dedicated memory. A few hundred neurons can be simulated on a single chip, each with ~1000 input synapses (26).

PACMAN is a software layer that may be used to write models. Languages such as PyNN and Nengo may be implemented within PACMAN.

Papers still to read:

* Simulation of cortical microcircuits on spinnaker
* A million spiking neuron integrated circuit
* Implementing STDP on Spinnaker hardware

Topics still to read about:

* PACMAN
* PyNN
* Nengo

## CPU/GPU Methods (2)

## Other (11,27)

References

(1) Lichtsteiner P, Posch C and Delbruck T. *A 128× 128 120 dB 15 μs latency asynchronous temporal contrast vision sensor.* IEEE; 2008. Available from: <https://www.openaire.eu/search/publication?articleId=od_______885::9ad842d32b2c52842f5d6ec7e1d9d526> .

(2) Bichler O, Querlioz D, Thorpe SJ, Bourgoin J and Gamrat C. Extraction of temporally correlated features from dynamic vision sensors with spike-timing-dependent plasticity. *Neural Networks.* 2012; 32 339-348. Available from: doi: 10.1016/j.neunet.2012.02.022 Available from: <https://www.sciencedirect.com/science/article/pii/S0893608012000512> .

(3) Farabet C, Paz R, Pérez-Carrasco J, Zamarreño-Ramos C, Linares-Barranco A, Lecun Y, et al. *Comparison between frame-constrained fix-pixel-value and frame-free spiking-dynamic-pixel convNets for visual processing.* Switzerland: Frontiers Media S.A; 2012. Available from: <https://www.ncbi.nlm.nih.gov/pubmed/22518097> .

(4) Guyonneau R, VanRullen R and Thorpe SJ. Temporal codes and sparse representations: A key to understanding rapid processing in the visual system. *Journal of Physiology - Paris.* 2004; 98 (4): 487-497. Available from: doi: 10.1016/j.jphysparis.2005.09.004 Available from: <https://www.sciencedirect.com/science/article/pii/S0928425705000185> .

(5) VanRullen R, Guyonneau R and Thorpe SJ. Spike times make sense. *Trends in Neurosciences.* 2005; 28 (1): 1-4. Available from: doi: 10.1016/j.tins.2004.10.010 Available from: <https://www.sciencedirect.com/science/article/pii/S0166223604003546> .

(6) Kheradpisheh SR, Ganjtabesh M and Masquelier T. Bio-inspired unsupervised learning of visual features leads to robust invariant object recognition. *Neurocomputing.* 2016; 205 382-392. Available from: doi: 10.1016/j.neucom.2016.04.029 Available from: <https://www.sciencedirect.com/science/article/pii/S0925231216302880> .

(7) Ponulak F, Kasinski A. Introduction to spiking neural networks: Information processing, learning and applications. *Acta neurobiologiae experimentalis.* 2011; 71 (4): 409. Available from: <https://www.ncbi.nlm.nih.gov/pubmed/22237491> .

(8) Mu-ming Poo. *Neuron.* 2018; 99 (5): 884-886. Available from: doi: 10.1016/j.neuron.2018.08.023 Available from: <https://www.sciencedirect.com/science/article/pii/S0896627318307268> .

(9) Timothée Masquelier, Rudy Guyonneau and Simon J Thorpe. *Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains.* San Francisco: Public Library of Science; 2008. Available from: <https://www.ncbi.nlm.nih.gov/pubmed/18167538> .

(10) Feldman D. The Spike-Timing Dependence of Plasticity. *Neuron.* 2012; 75 (4): 556-571. Available from: doi: 10.1016/j.neuron.2012.08.001 Available from: <https://www.sciencedirect.com/science/article/pii/S0896627312007039> .

(11) Zamarreño-Ramos C, Camuñas-Mesa LA, Pérez-Carrasco JA, Masquelier T, Serrano-Gotarredona T and Linares-Barranco B. *On spike-timing-dependent-plasticity, memristive devices, and building a self-learning visual cortex.* Switzerland: Frontiers Research Foundation; 2011. Available from: <https://www.ncbi.nlm.nih.gov/pubmed/21442012> .

(12) Masquelier T, Thorpe SJ. *Unsupervised learning of visual features through spike timing dependent plasticity.* United States: Public Library of Science; 2007. Available from: <https://www.ncbi.nlm.nih.gov/pubmed/17305422> .

(13) Maass W. Networks of spiking neurons: The third generation of neural network models. *Neural Networks.* 1997; 10 (9): 1659-1671. Available from: doi: 10.1016/S0893-6080(97)00011-7 Available from: <https://www.sciencedirect.com/science/article/pii/S0893608097000117> .

(14) Grüning A, Bohte S. Spiking Neural Networks: Principles and Challenges. ; 2014. Available from: <http://epubs.surrey.ac.uk/806438/>.

(15) Stromatias E, Soto M, Serrano-Gotarredona T and Linares-Barranco B. *An event-driven classifier for spiking neural networks fed with synthetic or dynamic vision sensor data.* Switzerland: Frontiers Media S.A; 2017. Available from: <https://www.ncbi.nlm.nih.gov/pubmed/28701911> .

(16) Tavanaei A, Ghodrati M, Kheradpisheh SR, Masquelier T and Maida A. Deep learning in spiking neural networks. *Neural Networks.* 2019; 111 47-63. Available from: doi: 10.1016/j.neunet.2018.12.002 Available from: <https://www.sciencedirect.com/science/article/pii/S0893608018303332> .

(17) Lee JH, Delbruck T and Pfeiffer M. *Training deep spiking neural networks using backpropagation.* Switzerland: Frontiers Research Foundation; 2016. Available from: <https://www.ncbi.nlm.nih.gov/pubmed/27877107> .

(18) Filip Ponulak, Andrzej Kasi ński. Supervised Learning in Spiking Neural Networks with ReSuMe: Sequence Learning, Classification, and Spike Shifting. *Neural Computation.* 2010; 22 (2): 467-510. Available from: doi: 10.1162/neco.2009.11-08-901 Available from: <http://www.mitpressjournals.org/doi/abs/10.1162/neco.2009.11-08-901> .

(19) Ponulak F. Analysis of the ReSuMe Learning Process For Spiking Neural Networks. *International Journal of Applied Mathematics and Computer Science.* 2008; 18 (2): 117-127. Available from: doi: 10.2478/v10006-008-0011-1 Available from: <https://search.proquest.com/docview/1321087370> .

(20) Leung FHF, Lam HK, Ling SH and Tam PKS. Tuning of the structure and parameters of a neural network using an improved genetic algorithm. *IEEE Transactions on Neural Networks.* 2003; 14 (1): 79-88. Available from: doi: 10.1109/TNN.2002.804317 Available from: <https://ieeexplore.ieee.org/document/1176129> .

(21) AmirhossTeaivnanaTeimo, thee MatsaqnudeAlnitehronSy.Maida. *AcquisitoifVoinsuaFleaturTehsrough ProbabilSipsitkiec-Timing-DepPelnadsetnitcity.*

(22) Maniezzo V. Genetic evolution of the topology and weight distribution of neural networks. *IEEE Transactions on Neural Networks.* 1994; 5 (1): 39-53. Available from: doi: 10.1109/72.265959 Available from: <https://ieeexplore.ieee.org/document/265959> .

(23) 伊東春樹. CPX-1 CPX Live demonstration. *日本臨床生理学会雑誌.* 2016; 46 (4): 47. Available from: <http://mol.medicalonline.jp/library/journal/download?GoodsID=cc8jjapf/2016/004604/004&name=0047-0047j> .

(24) Orchard G, Jayawant A, Cohen GK and Thakor N. *Converting static image datasets to spiking neuromorphic datasets using saccades.* Switzerland: Frontiers Research Foundation; 2015. Available from: <https://www.ncbi.nlm.nih.gov/pubmed/26635513> .

(25) Khan MM, Lester DR, Plana LA, et al. SpiNNaker: Mapping neural networks onto a massively-parallel chip multiprocessor. *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)* : IEEE; Jun 2008. pp. 2849-2856. Available from: <https://ieeexplore.ieee.org/document/4634199>. Available from: 10.1109/IJCNN.2008.4634199.

(26) Furber SB, Galluppi F, Temple S and Plana LA. The SpiNNaker Project. *Proceedings of the IEEE.* 2014; 102 (5): 652-665. Available from: doi: 10.1109/JPROC.2014.2304638 Available from: <https://ieeexplore.ieee.org/document/6750072> .

(27) Sharp T, Galluppi F, Rast A and Furber S. Power-efficient simulation of detailed cortical microcircuits on SpiNNaker. *Journal of Neuroscience Methods.* 2012; 210 (1): 110-118. Available from: doi: 10.1016/j.jneumeth.2012.03.001 Available from: <https://www.sciencedirect.com/science/article/pii/S0165027012000866> .