Information Theory for Deep Spiking Neural Networks

Leonardo Capozzi and Tiago Gonçalves
Faculdade de Engenharia
Universidade do Porto
Porto, Portugal
{up201503708, up201607753}@fe.up.pt

Abstract

Deep learning methodologies have been very successful in a large number of tasks, sometimes surpassing human performance. One of the most simple neural network architectures is the multi-layer perceptron (MLP) which tries to mimic the brain in some ways. These methodologies are generally trained using gradient descent as they are differentiable. In recent years, a new methodology called spiking neural networks (SNNs) was proposed. These networks can be seen as a more biologically realistic approach than artificial neural networks (ANNs), as they use discrete spikes to transmit information, instead of continuous values. In this paper, we study these networks, and present results achieved by training these models. We also evaluate several information theory quantities such as entropy and mutual information of these networks to extract the relationship between the inputs and the outputs.

1. Introduction

In recent years deep learning methodologies have proved to be very successful in a large number of problems, namely classification in computer vision tasks, sometimes challenging human performance. One of the most simple neural network architectures is the multi-layer perceptron (MLP), which consists of a neural network where every neuron in a certain layer is connected to every neuron in the next layer. These kinds of networks are trained using the backpropagation algorithm, since the loss function that we are trying to minimize is differentiable, allowing us to calculate the gradients of the weights and optimize them. Neurons in a MLP contain single and continuous activations. Biological brains, on the other hand, use discrete spikes to transmit information. With this in mind, a new methodology called spiking neural networks (SNNs) was proposed.

SNNs can be seen as a more biologically realistic approach than artificial neural networks (ANNs). SNNs typically require fewer operations, are more energy-efficient, and also more hardware-efficient, making them very appealing for the future [25]. One of the current problems with these kinds of networks is the difficulty in training them since they are non-differentiable and therefore the backpropagation algorithm cannot be used. SNNs can be trained using supervised or unsupervised approaches. Currently, SNNs are still inferior in terms of accuracy compared to ANNs, but the gap is closing due to advances in recent years.

This paper intends to present an exploratory study using three benchmark data sets (MNIST, Fashion-MNIST, and CIFAR-10) to train and test SNNs and to perform an analysis from the perspective of information theory, through the computation of several metrics.

Besides the Introduction, the rest of this paper is organized as follows: section 2 presents the main concepts of the field of spiking neural networks, section 3 describes the data sets and the details of the implementation of the experiments, section 4 shows the results and exposes a brief discussion and section 5 concludes this paper and proposes future research directions. The code related to this paper is available in a public GitHub repository[1].

2. Related Work

2.1. Fundamental Concepts of SNNs

2.1.1 Overview of the Architecture of SNNs

The development of spiking neural networks (SNNs) was inspired by the structure and functioning of the brain. In the neurons, the information is processed via discrete action potentials (spikes) in time through synapses [25]. In the realm of Biology, a spike is generated when the running sum of changes in the membrane potential crosses a threshold. The information about these external stimuli (or ongoing computations) is encoded in the rate of spike generation and on the temporal pattern of spike trains [9, 23].

[1] https://github.com/TiagoFilipeSousaGoncalves/computational-neuroscience

The baseline architecture of a SNN consists of spiking neurons and interconnecting synapses that are modeled by adjustable scalar weights. One of the main concerns relies on the proper encoding of the data to be fed to the SNN during the training phase. The input data can be modelled using a rate-based method [9, 10], a temporal encoding [1, 14] or population coding [2]. Although biological neurons have highly complex dynamics regarding action potential generation and network interaction, in SNNs, we consider that the modeled spiking neurons only deal with threshold dynamics [25]. This approach was initially proposed by Hodgkin and Huxley and considers that the activity of presynaptic neurons modulates the membrane potential of postsynaptic neurons, generating an action potential or spike when the membrane potential crosses a threshold [13]. Recently, several dynamic models have been proposed, such as the spike response model (SRM) [16], the Izhikevich neuron model [15], and the leaky integrated-and-fire (LIF) neuron [6]. In SNNs, the spike trains are propagated through synaptic connections: a synaptic can be either excitatory (*i.e.*, increases the neuron's membrane potential upon receiving input) or inhibitory (*i.e.*, decreases the neuron's membrane potential) [17].

2.1.2 The Learning Process in SNNs

The spike trains are formally represented as sums of Dirac delta ($\delta(\cdot)$) functions and do not have derivatives. Therefore, gradient-based optimization rules may not be trivial to implement in SNNs. However, similarly to ANNs, the main objective is to learn the best synaptic weights. One of the paradigms in computational neuroscience relies on the concept of spike-timing-dependent plasticity (STDP). The intuition behind this approach is that the weight (synaptic efficacy) connecting a pre- and post-synaptic neuron is adjusted according to their relative spike times within an interval of milliseconds in length [3]. To understand this concept, we need to revisit Biology: if we know that the pre-synaptic neuron fires for a brief time (*e.g.*, $\approx$ 10ms) before the post-synaptic neuron, the weight connecting them is strengthened; on the other hand, if the pre-synaptic neuron fires briefly after the post-synaptic neuron, then the causal relationship between the temporal events is nonsense and the weight connecting these two neurons is weakened. The first case is called long-term potentiation (LTP) and the second is called long-term depression (LTD). Please note the usage of the expression "long-term": we want to distinguish between very transient effects on the scale of a few ms that are observed in experiments. Mathematically, both LTP and LTD can be described by Equation 1:

$$\Delta w = \begin{cases} Ae^{-\frac{(|t_{pre}-t_{post}|)}{\tau}}, t_{pre} - t_{post} \leq 0, A > 0 \\ Be^{-\frac{(|t_{pre}-t_{post}|)}{\tau}}, t_{pre} - t_{post} > 0, B < 0 \end{cases} \quad (1)$$

where, w is the synaptic weight, A and B are constant learning rates, τ is the time constant for the temporal learning window, and t_{pre} and t_{post} are the spiking times for the pre- and post-synaptic neurons, respectively. From this equation, it is possible to observe that the first branch represents LTP and the second branch represents LTD.

2.1.3 Deep Learning in SNNs

Deep learning can be defined as the set of algorithms that employ ANNs with multiple hidden layers. The popularity of these algorithms is mainly due to the concept of *representation learning*, which is the capability of the algorithms to discover the representations needed for detection or classification [19]. This ability allows practitioners to use data in its raw form to train the models. Deep learning methods can thus be seen as representation-learning methods with multiple levels of representation, starting from the raw input to a representation of a more abstract level. With the composition of enough such transformations, deep learning creates the conditions to learn very complex functions. Another interesting aspect of deep learning is that it generally relies on a gradient-based optimization strategy. A major drawback of deep learning is related to the fact that these models usually have millions of parameters, thus being very difficult to deploy into energy-efficient neuromorphic computation hardware [25]. On the other hand, SNNs are known to be energy-efficient and have the plausibility to be deployed into neuromorphic computation hardware. Thus, several efforts have been put into the development of deep SNNs with the same performance as deep-based ANNs. Current approaches propose several variations to the conventional STDP-based learning in the sense that it becomes possible to apply the backpropagation algorithm to optimize the synaptic weights of the SNN [22].

2.2. Fundamental Concepts of Information Theory

We present the fundamental concepts of information theory, needed to understand the intuition behind our analysis. All the concepts are presented as in [5].

2.2.1 Information

Claude Shannon derived a way of measuring information content called "self-information". Formally, it is defined as:

$$I_X(x) = -log_2[p_X(x)] \quad (2)$$

where X is a random variable with probability mass function $p_X(x)$.

Shannon's definition of information follows several axioms:

1. An event with a probability of 100% is completely unsurprising and therefore yields no information.

2. The lower the probability of an event the more information it yields, as it is more surprising.

3. The total amount of information of two independent events measured separately is the sum of the information of the individual events.

2.2.2 Entropy

The definition of entropy is the expected information content of random variable X. It is defined as:

$$H(X) = -\sum_{x} p_X(x) log_2[p_X(x)] \tag{3}$$

The joint entropy of two random variables X and Y is defined as:

$$H(X,Y) = -\sum_{x,y} p(x,y) log_2[p(x,y)] \tag{4}$$

The conditional entropy is defined as the entropy of random variable X given Y, and is written as:

$$H(X|Y) = -\sum_{y} p(y) \sum_{x} p(x|y) log_2[p(x|y)] \tag{5}$$

2.2.3 Mutual Information

Mutual information refers to the amount of information that can be obtained about one random variable by observing another random variable. The mutual information between two random variables X and Y is given as:

$$I(X;Y) = \sum_{x,y} p(x,y) log_2 \frac{p(x,y)}{p(x)p(y)} \tag{6}$$

Mutual information is also related to entropy in the following ways:

$$I(X;Y) = H(X) - H(X|Y) \tag{7}$$

$$I(X;Y) = H(X) + H(Y) - H(X,Y) \tag{8}$$

3. Methodology

3.1. Data

We performed experiments with three benchmark data sets, explained below.

3.1.1 MNIST

Proposed by LeCun *et al.* [20], the MNIST data set is composed of 28×28 grey-scale images of $70,000$ handwritten digits from 10 categories. The training set has $60,000$ images and the test set has $10,000$ images.

3.1.2 Fashion-MNIST

Proposed by Xiao, Rasul and Vollgraf [26], the Fashion-MNIST data set was created "to serve as a direct drop-in replacement for the original MNIST data set for benchmarking machine learning algorithms, as it shares the same image size, data format and the structure of training and testing splits". This data set is composed of 28×28 grey-scale images of $70,000$ fashion products from 10 categories, with $7,000$ images per category. The training set has $60,000$ images and the test set has $10,000$ images.

3.1.3 CIFAR-10

Collected by Krizhevsky, Nair and Hinton [18], the CIFAR-10 data set is composed of 32×32 colour images of $60,000$ variable classes from 10 categories. The training set has $50,000$ images and the test set has $10,000$ images.

3.2. Implementation

All the models were implemented in Python, using the BindsNET library. Written on top of the PyTorch [2] framework, BindsNET is a spiking neural network simulation library that can leverage the SNN development in CPU or GPU [12].

3.2.1 SNN Architecture

We follow the examples provided in [11], which are based on the SNN architecture proposed by [8, 7]. This architecture (see Figure 1) consists of two layers: 1) an input layer, that contains 28×28 or 32×32 neurons (*i.e.*, one neuron per pixel); 2) a processing layer, composed of a variable, but equal, number of excitatory and inhibitory neurons [7]. Each input image is modeled as a Poisson spike-train and is fed into the excitatory neurons of the second layer. These excitatory neurons are then connected in a one-to-one fashion to inhibitory neurons (*i.e.*, each spike in an excitatory neuron will trigger a spike in its corresponding inhibitory neuron). On the other hand, each of the inhibitory neurons is connected to all excitatory ones, except for the one from which it receives a connection.

3.2.2 Training

This approach uses the integrate-and-fire (IF) spiking neuron model, which is given by Equation 9:

$$\frac{dv_{mem}(t)}{dt} = \sum_{i} \sum_{s \in S_i} w_i \delta(t-s) \tag{9}$$

where v_{mem} is the membrane voltage, t is the time, w_i is the weight of the i-th incoming synapse, $\delta(\cdot)$ is the Dirac delta

[2] Available at: https://pytorch.org

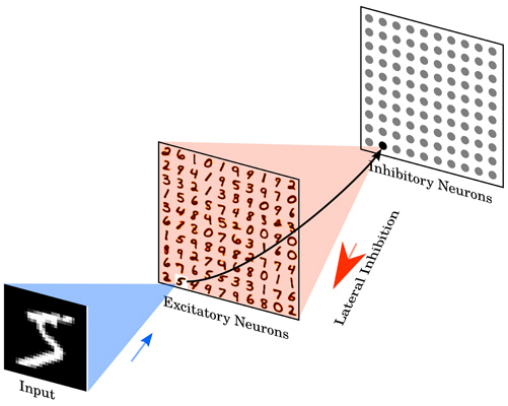

Figure 1. The SNN architecture, from [7].

function, and $S_i = \{t_i^0, t_i^1, ...\}$ contains the spike times of the i-th presynaptic neuron. A spike is generated if the membrane crosses a given threshold v_{thr}; the membrane voltage is then reset to a reset potential $vres$. As in [8], the continuous-time description of this model is discretised into 1ms time-steps.

The synapses from input neurons to excitatory neurons are learned by STDP. The weight dynamics are computed using synaptic traces, as proposed in [21]. This approach ensures that besides the synaptic weight, each synapse keeps track of another value (*i.e.*, the presynaptic trace x_{pre}), which models the recent presynaptic spike history. Hence, every time a presynaptic spike arrives at the synapse, the trace is increased by 1 and x_{pre} decays exponentially. The weight change Δw provoked by the arrival of a postsynaptic arrival is computed according to Equation 10:

$$\Delta w = \eta(x_{pre} - x_{tar})(w_{max} - w)^{\mu} \tag{10}$$

where η is the learning-rate, w_{max} is the maximum weight, μ determines the dependence of the update on the previous weight, and x_{tar} is the target value of the presynaptic trace at the moment of a postsynaptic spike [7]. From this result, we may conclude that the higher the target value, the lower the synaptic weight will be. Also, this offset promotes that presynaptic neurons that rarely lead to the firing of the postsynaptic neuron will become more and more disconnected. This is particularly useful if the postsynaptic neuron is only rarely active.

The fact that the inputs of the network are not homogeneous leads to different firing rates of the excitatory neurons. This is not very desirable, as the goal is to have approximately equal firing rates across neurons to prevent the same neurons from being always active. To achieve this [7] implemented an adaptive membrane threshold. Each excitatory neuron's membrane has a value of θ added to the threshold value v_{thresh}, therefore the final threshold of an excitatory neuron becomes $v_{thresh} + \theta$. The value of θ is increased every time the neuron fires and decays exponentially. This causes neurons that fire very frequently to have a higher membrane threshold, making them require more inputs to spike soon. This causes the neuron to fire less frequently since once the neuron's membrane threshold reaches values close or higher than the excitatory reversal potential E_{exc}, the neuron will fire less, or stop firing at all, until θ decreases.

The inputs of the network are based on the MNIST, Fashion MNIST, and CIFAR-10 datasets, each containing 10 classes. The CIFAR-10 dataset was converted to grayscale to be consistent with the other datasets. The inputs are presented to the network in the form of Poisson-distributed spike trains, with firing rates proportional to the intensity of the pixels.

After training is done we assign a class to each neuron based on the highest average response for each of the 10 classes. This is the only time that the labels are used since they are not used for the training of the synaptic weights.

To predict the class of the input, we calculate the average firing rate of the neurons of each class and choose the class with the highest average firing rate.

3.2.3 Applying information theory to SNNs

After training the model, we generate predictions for the test set. We record samples of the inputs (*i.e.*, one image per label) and samples of the outputs (*i.e.*, the spike train generated by the output neurons for a given input image). With this data, we performed simple tests based on information theory:

1. **Computation of the entropy of the input spike train:** we have the distribution of spikes per neuron along time. Hence, we can compute, per neuron, the following probabilities: $P_n(i = 1)$ and $P_n(i = 0)$, where i represents the input spike, $i \in \{0, 1\}$, n represents the input neuron index and $n \in \{0, ..., N\}$. Please note that N represents the number of input neurons: for the MNIST and Fashion-MNIST data sets $N = 28 \times 28 = 784$, and for the CIFAR10 data set $N = 32 \times 32 = 1024$. Although it may not be useful at an individual level, this result allows to estimate the amount of information of a given neuron.

2. **Computation of the entropy of the output spike train:** we have the distribution of spikes per neuron along time. Hence, we can compute, per neuron, the following probabilities: $P_n(o = 1)$ and $P_n(o = 0)$, where o represents the output spike, $o \in \{0, 1\}$. n represents the output neuron index and $n \in \{0, ..., 100\}$. Although it may not be useful at an individual level, this result allows to estimate the amount of information of a given neuron.

Table 1. Accuracy performances obtained by the SNN trained on different data sets.

Dataset	Accuracy (%)
MNIST	72
Fashion-MNIST	46
CIFAR-10	9

3. **Computation of the mutual information between the spike trains of the input and output neurons:** from the previous points, we obtain the marginal distributions of the input and output neurons. To compute the mutual information between the input and output neurons, we need to compute the joint distribution $P(i, o)$ which is necessary for the computation of the joint entropy $H(i, o)$. We consider mutual information an interesting metric since it may provide useful insights about the neurons that are related during the processing of a given input. We compute this metric through the combination of the set of input neurons and the output neurons. Hence, we are capable of computing the mutual information between a single output neuron and a single input neuron. If the mutual information between a pair of neurons is greater than zero, it means that those neurons have information in common. Therefore, we sum all the single contributions from the input neurons related to an output neuron, thus, obtaining a final value for that output neuron.

4. Results and Discussion

Table 1 presents the accuracy performance of the SNN on the test subset of each benchmark data set.

Table 2 presents the set of indices of the neurons that spiked on the output layer and the set of indices of the output neurons that have mutual information with the input of the network.

The results regarding the analysis from the perspective of information theory are presented and discussed below, divided by data set.

4.1. MNIST

Figure 2 shows an example of spike train obtained after modelling the input image into a Poisson-distribution. This example shows the result for an image with label "4".

Figure 3 shows an example of spike train obtained by the output neurons after the processing of the inputs. This example shows the result for an image with label "4", to be coherent with Figure 2.

4.2. Fashion-MNIST

Figure 4 shows an example of spike train obtained after modelling the input image into a Poisson-distribution. This example shows the result for an image with label "3".

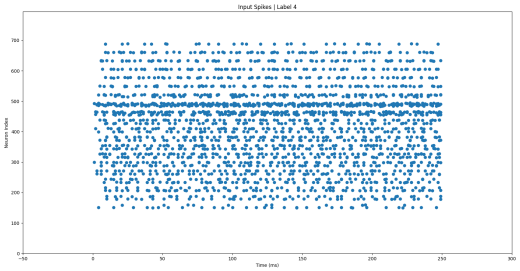

Figure 2. The result of a Poisson-distributed spike train for an example of an image from the MNIST data set, with label "4".

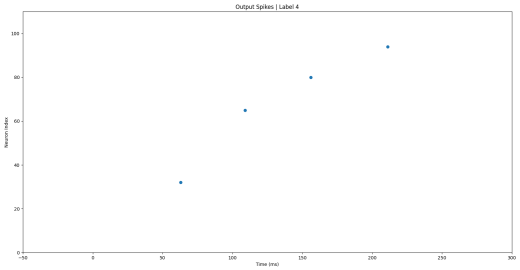

Figure 3. The result of the spike train obtained by the output neurons after the processing of information obtained form an image from the MNIST data set, with label "4".

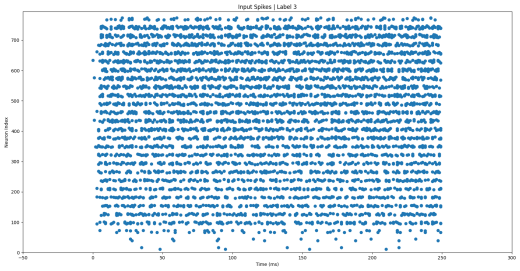

Figure 4. The result of a Poisson-distributed spike train for an example of an image from the Fashion-MNIST data set, with label "3".

Figure 5 shows an example of spike train obtained by the output neurons after the processing of the inputs. This example shows the result for an image with label "3", to be coherent with Figure 4.

4.3. CIFAR-10

Figure 6 shows an example of spike train obtained after modelling the input image into a Poisson-distribution. This example shows the result for an image with label "0".

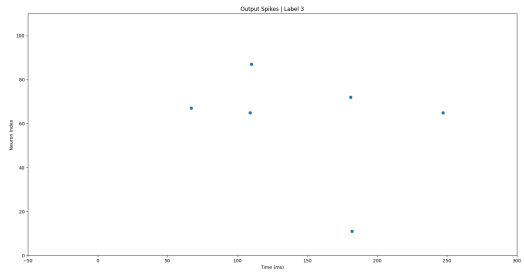

Figure 5. The result of the spike train obtained by the output neurons after the processing of information obtained form an image from the Fashion-MNIST data set, with label "3".

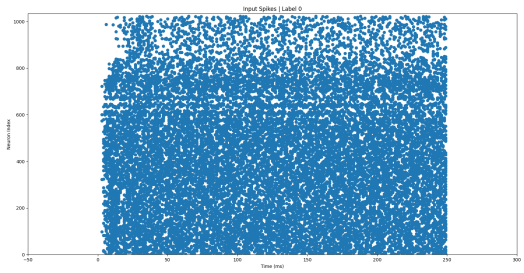

Figure 6. The result of a Poisson-distributed spike train for an example of an image from the CIFAR-10 data set, with label "0".

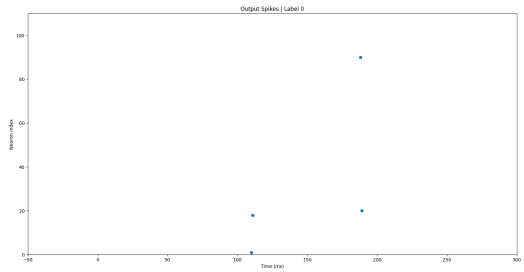

Figure 7. The result of the spike train obtained by the output neurons after the processing of information obtained form an image from the CIFAR-10 data set, with label "0".

Figure 7 shows an example of spike train obtained by the output neurons after the processing of the inputs. This example shows the result for an image with label "0", to be coherent with Figure 6.

5. Conclusions and Future Work

This work presented an exploratory study with the SNN model proposed by [8] in three benchmark data sets. Please note that the main goal was to train a SNN and to compute information theory quantities related to the input and output neurons of the SNN. Hence, although we also present the results in terms of accuracy performances, they are not relevant for the analysis we wanted to perform in this work.

We computed the entropy of the input neurons, the entropy of the output neurons, and the mutual information between the input neurons and the output neurons, for a sample of a given input. Results suggest that the output neurons that present higher values of mutual information related to the input neurons are the ones that fire when given an image of a certain class, which confirms our initial intuition. We also acknowledge that this is an early-stage study. For instance, we point to the work of [24] which introduces a new class of spike train metrics, inspired by the Pompeiu-Hausdorff distance [4], which measures how far two subsets of a metric space are from each other.

Further work should be developed to the improvement of the accuracy performances and the analysis of the impact of this metric on the information theory analysis quantities.

References

[1] S. M. Bohte. The evidence for neural information processing with precise spike-times: A survey. *Natural Computing*, 3(2):195–206, 2004. 2

[2] S. M. Bohte, H. La Poutré, and J. N. Kok. Unsupervised clustering with spiking neurons by sparse temporal coding and multilayer rbf networks. *IEEE Transactions on neural networks*, 13(2):426–435, 2002. 2

[3] N. Caporale and Y. Dan. Spike timing–dependent plasticity: a hebbian learning rule. *Annu. Rev. Neurosci.*, 31:25–46, 2008. 2

[4] A. Conci and C. Kubrusly. Distance between sets-a survey. *arXiv preprint arXiv:1808.02574*, 2018. 6

[5] T. M. Cover. *Elements of information theory*. John Wiley & Sons, 1999. 2

[6] A. Delorme, J. Gautrais, R. Van Rullen, and S. Thorpe. Spikenet: A simulator for modeling large networks of integrate and fire neurons. *Neurocomputing*, 26:989–996, 1999. 2

[7] P. Diehl and M. Cook. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in Computational Neuroscience*, 9:99, 2015. 3, 4

[8] P. U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu, and M. Pfeiffer. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2015. 3, 4, 6

[9] W. Gerstner and W. M. Kistler. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002. 1, 2

[10] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014. 2

[11] G. Hadash, E. Kermany, B. Carmeli, O. Lavi, G. Kour, and A. Jacovi. Estimate and replace: A novel approach to in-

Table 2. Relationship between the index of the neurons that spiked on the output layer, and the index of the output neurons that have mutual information with the input of the network.

Class	MNIST		Fashion-MNIST		CIFAR-10	
	Output neurons that spiked	Neurons with mutual information	Output neurons that spiked	Neurons with mutual information	Output neurons that spiked	Neurons with mutual information
0	{3, 4, 29, 31, 52}	{3, 4, 29, 31, 52}	{22, 24, 64, 67, 85}	{22, 24, 64, 67, 85}	{1, 18, 20, 90}	{1, 18, 20, 90}
1	{8, 97}	{8, 97}	{7, 11, 16, 65, 67, 72, 87}	{7, 11, 16, 65, 67, 72, 87}	{}	{}
2	{23, 71, 85}	{23, 71, 85}	{4, 12, 29, 32, 66, 71}	{4, 12, 29, 32, 66, 71}	{18}	{18}
3	{2, 7, 14}	{2, 7, 14}	{11, 65, 67, 72, 87}	{11, 65, 67, 72, 87}	{45, 73}	{45, 73}
4	{32, 65, 80, 94}	{32, 65, 80, 94}	{}	{}	{}	{}
5	{32, 37, 67, 75, 78}	{32, 37, 67, 75, 78}	{}	{}	{}	{}
6	{17, 28}	{17, 28}	{2, 15, 19, 39}	{2, 15, 19, 39}	{11, 28}	{11, 28}
7	{18, 26, 92, 99}	{18, 26, 92, 99}	{77}	{77}	{}	{}
8	{11, 33, 35, 90}	{11, 33, 35, 90}	{3, 21, 96, 97}	{3, 21, 96, 97}	{1, 18, 23, 36}	{1, 18, 23, 36}
9	{38, 65, 94}	{38, 65, 94}	{0, 21, 92}	{0, 21, 92}	{1, 14, 18, 23}	{1, 14, 18, 23}

tegrating deep neural networks with existing applications. *arXiv preprint arXiv:1804.09028*, 2018. 3

[12] 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*, 12:89, 2018. 3

[13] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500–544, 1952. 2

[14] J. J. Hopfield. Pattern recognition computation using action potential timing for stimulus representation. *Nature*, 376(6535):33–36, 1995. 2

[15] E. M. Izhikevich. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572, 2003. 2

[16] R. Jolivet, J. Timothy, and W. Gerstner. The spike response model: a framework to predict neuronal spike trains. In *Artificial neural networks and neural information processing—ICANN/ICONIP 2003*, pages 846–853. Springer, 2003. 2

[17] E. R. Kandel, J. H. Schwartz, T. M. Jessell, S. Siegelbaum, A. J. Hudspeth, and S. Mack. *Principles of neural science*, volume 4. McGraw-hill New York, 2000. 2

[18] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images, 2009. 3

[19] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015. 2

[20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 3

[21] A. Morrison, A. Aertsen, and M. Diesmann. Spike-timing-dependent plasticity in balanced random networks. *Neural computation*, 19(6):1437–1467, 2007. 4

[22] E. O. Neftci, C. Augustine, S. Paul, and G. Detorakis. Event-driven random back-propagation: Enabling neuromorphic deep learning machines. *Frontiers in neuroscience*, 11:324, 2017. 2

[23] F. Rieke. *Spikes: exploring the neural code*. MIT press, 1999. 1

[24] C. V. Rusu and R. V. Florian. A new class of metrics for spike trains. *Neural Computation*, 26(2):306–348, 2014. 6

[25] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. S. Maida. Deep learning in spiking neural networks. *CoRR*, abs/1804.08150, 2018. 1, 2

[26] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017. 3