Supervised learning of spatial features with STDP and homeostasis using Spiking Neural Networks on SpiNNaker

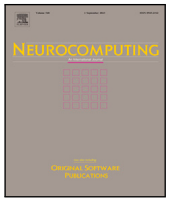
DAVIES, Sergio, GAIT, Andrew, ROWLEY, Andrew and DI NUOVO, Alessandro <http://orcid.org/0000-0003-2677-2650>

Neurocomputing 611 (2025) 128650

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



Supervised learning of spatial features with STDP and homeostasis using Spiking Neural Networks on SpiNNaker☆

Sergio Davies [a,*], Andrew Gait [b], Andrew Rowley [b], Alessandro Di Nuovo [c]

[a] Department of Computing and Mathematics, Manchester Metropolitan University, John Dalton Building, Chester Street, Manchester, M1 5GD, United Kingdom
[b] APT group, School of Computer Science, The University of Manchester, IT Building, Oxford Road, Manchester, M13 9PL, United Kingdom
[c] Department of Computing, Sheffield Hallam University, Cantor Building, 153 Arundel Street, Sheffield, S1 2NU, United Kingdom

ARTICLE INFO

Communicated by F. Perez-Pena

Dataset link: 10.23634/MMU.00634935, https://github.com/sergiodavies/SpiNNakerSpatialLearningCodeAndDataset

Keywords:
Spiking Neural Networks
SNN
Spatial pattern
STDP
Spike Timing Dependent Plasticity
Supervised learning

ABSTRACT

Artificial Neural Networks (ANN) have gained significant popularity thanks to their ability to learn using the well-known backpropagation algorithm. Conversely, Spiking Neural Networks (SNNs), despite having broader capabilities than ANNs, have always posed challenges in the training phase. This paper shows a new method to perform supervised learning on SNNs, using Spike Timing Dependent Plasticity (STDP) and homeostasis, aiming at training the network to identify spatial patterns. Spatial patterns refer to spike patterns without a time component, where all spike events occur simultaneously. The method is tested using the SpiNNaker digital architecture. A SNN is trained to recognise one or multiple patterns and performance metrics are extracted to measure the performance of the network. Some considerations are drawn from the results showing that, in the case of a single trained pattern, the network behaves as the ideal detector, with 100% accuracy in detecting the trained pattern. However, as the number of trained patterns on a single network increases, the accuracy of identification is linked to the similarities between these patterns. This method of training an SNN to detect spatial patterns may be applied to pattern recognition in static images or traffic analysis in computer networks, where each network packet represents a spatial pattern. It will be stipulated that the homeostatic factor may enable the network to detect patterns with some degree of similarity, rather than only perfectly matching patterns. The principles outlined in this article serve as the fundamental building blocks for more complex systems that utilise both spatial and temporal patterns by converting specific features of input signals into spikes. One example of such a system is a computer network packet classifier, tasked with real-time identification of packet streams based on features within the packet content.

1. Introduction

The rising popularity of neural networks can be attributed to their information processing capabilities, despite being regarded as "black-box" systems due to their emulation of the behaviour of biological neural networks, rather than relying on established biological structures [1].

Artificial neural network models draw inspiration from their biological counterparts, attempting to mimic how the human brain performs specific tasks [2]. Based on the computational units (neurons) used in these networks, we can classify three main categories of neural networks [3]. Each of these categories is referred to as a "generation" of neural networks. Each generation simulates biological processes with an increasing degree of accuracy.

The first generation of neural networks were dominated by the McCulloch-Pitts neuron model [4] which allows discrete inputs and outputs (only "0"s or "1"s). The next generation (second generation of neural networks, more commonly known as Artificial Neural Networks) evolved this model to allow input and output values to be continuous within a specified range, either $[0;1]$ or $[-1;1]$.

In both these generations of neural networks, the output of a neuron is transferred to the subsequent neuron(s) through weighted connections. This weight is altered during the training phase by presenting the

☆ Using STDP and homeostasis on spiking neural networks simulated on SpiNNaker, Davies et al. demonstrate that it is possible for such a network to learn and recognise spike patterns by presenting the desired pattern to the network only once. The pattern is presented as a set of simultaneous spikes at the input layer, and the output is produced after a short delay. In addition, the same network is trained with multiple patterns, and the accuracy and other performance metrics are computed.

* Corresponding author.
E-mail address: sergio.davies@mmu.ac.uk (S. Davies).
URL: https://www.mmu.ac.uk/staff/profile/dr-sergio-davies (S. Davies).

https://doi.org/10.1016/j.neucom.2024.128650
Received 14 March 2024; Received in revised form 24 June 2024; Accepted 19 September 2024
Available online 24 September 2024

network with an input for the training session, analysing the network output, and determining the error between the current network output and the desired output. This error is then used to compute the changes in synaptic weights throughout the network using the backpropagation algorithm [5,6].

It is suggested that the popularity of Artificial Neural Networks (ANNs) can be attributed to the use of the backpropagation training algorithm [7]. This algorithm has played a pivotal role in enabling the training of significant ANN models [e.g.: 8–10]. Indeed, backpropagation has provided an advantage to this type of networks with a well-known and robust method of training which has now been embedded in most, if not all, ANN simulation platforms.

However, both the first and the second generations of neural networks do not consider one fundamental aspect: biological networks evolve following biological time. Artificial neural networks perform their operations in abstract time that does not correspond to biological time. Even advanced models such as Continuous Timescale Recurrent Neural Networks (CTRNN) [11] and Multi-Timescale Recurrent Neural Networks (MTRNN) [12] use the recurrent structure of the neural network to keep track of the state and its evolution. However, this does not correspond to biological real-time, but rather follows the number of iterations in the network.

The third generation of neural networks [3], also known as Spiking Neural Networks (SNNs), improves the biological realism of previous generations of neural and synaptic models by introducing the time variable in the models [13,14]. Indeed, the models proposed for this generation of neural networks are directly inspired from biology: the most realistic model is the Hodgkin–Huxley neuron [15], which is also the most complex to simulate numerically on a computer. Other models, instead, limit their biological plausibility to reduce their numerical complexity [16]. All of these neuron models are described using differential equations that depict the evolution of a neuron's state over time [e.g. 14,17–19].

The communication between neurons is also inspired from biology: it is known that neurons interact by means of action potentials, also known as spikes [19]. Such communications use a wide array of mechanisms to encode information, as described by Auge et al. [20], and even more methods could be envisioned.

Similarly, also synapses (the interconnection between neurons) in second generation neural networks are represented by a single number that represents its "strength". This value is altered during training using the backpropagation algorithm. However, from a biological perspective, this algorithm has raised some skepticism on its plausibility [21]. This is also supported by the fact that biological findings have shown that signals transmitted through synapses have a time evolution that may be described through a differential equation [14].

In addition, biology has described a number of mechanisms that allow biological neural networks to adapt to input stimuli. Among these we can mention STP — short-term plasticity [22], STDP — Spike Timing Dependent Plasticity [23,24], homeostasis [25,26], structural plasticity [27] and evolutionary learning [6]. All these mechanisms, following different processes, alter the architecture of the neural network by altering the synaptic strength of existing synapses, by creating new synapses (synaptogenesis), or by removing existing synapses (synaptic pruning). Despite all the changes imposed by these mechanisms, neurons within the network need to maintain their functional stability, and the network itself needs to keep a stable behaviour. This happens through the homeostatic process at two levels: on a neuron scale it helps to keep a healthy neural activity, while on the network scale homeostasis helps keeping the network stability [28].

The learning process underlying mechanism was proposed by Hebb [29], and commonly summarised as *"Cells that fire together wire together"*. More details of this biological process have emerged in the last few decades, leading to a number of learning rules which can affect synapses on a time interval spanning a few milliseconds to a lifetime, or more, through generations of individuals [30,31].

As a general rule, the longer the learning period, the more permanent the effects are on the neural networks: short-term plasticity affects quickly the stability of the network, but the effects do not last very long [32]. On the other hand, long-term plasticity has a stronger impact on the network, so that it allows the network to self-organise towards a stable critical regime [30]. Evolutionary learning has an even stronger impact that allows generation of individuals to behave in a specific way innately [33].

Such learning rules have been replicated in computer simulations, and showed their characteristics in applied tasks. In particular, it is relevant to mention that STDP was successfully applied in many applications related to the identification of spatio-temporal spike patterns [e.g. 34–37].

A neural network is trained within an environment. On the basis of this it is possible to classify four learning paradigms [14] depending on the presence and the structure of the teaching signal: supervised learning, semi-supervised learning, unsupervised learning and reinforcement learning. The learning rules introduced before (STP, STDP, etc.) refer to an unsupervised learning paradigm, where the teaching signal is absent and the network aims to identify autonomously a pattern in the input signal.

In this paper we present a novel method of training a spiking neural network to identify spatial patterns (patterns of spikes presented at the same time as input to the network) using STDP and homeostasis: two learning algorithms acting on different time-scales collaborating to achieve a task. The network is trained initially to identify a single pattern and the accuracy is then evaluated by testing exhaustively all the possible input patterns to the network. In a second step, the network is trained to identify two patterns, and we will show that the accuracy of the identification depends strictly on the degree of similarity between the two patterns on which the network has been trained. This similarity will be measured by the Hamming Distance between input patterns. Finally, a more thorough experiment includes training the network on three patterns and measuring again the detection accuracy, among other classification metrics.

The experiments are performed on the SpiNNaker digital architecture [38], using the sPyNNaker implementation of the PyNN neural network language [39–41]. SpiNNaker is a system designed at the University of Manchester: each SpiNNaker chip comprises eighteen very-low-power ARM986 processors (cores); the main SpiNNaker server, housed at the University of Manchester, consists of a million cores built of multiple boards containing multiple chips. Computations in the brain are inherently parallel and the architecture is designed to mimic this parallelism. SNNs may be simulated on the machine by submitting scripts based on the PyNN neural network language. These scripts are then converted by the software stack into executable files which run on as many cores as required by the neural network.

The remaining sections of this article encompass a detailed account of the experiments outlined in the methodology section (Section 2). This includes an in-depth exploration of the STDP learning rule (Section 2.1), training procedures (Section 2.2), and testing methods (Section 2.3). Subsequently, the results section (Section 3) will shed light on the research outcomes, involving training the network on a single pattern (Section 3.1), two patterns (Section 3.2), and multiple patterns (Section 3.3). In each case, various classification metrics will be employed to assess the network's performance in pattern identification following the training process. Finally, the conclusion section (Section 4) will summarise the key findings of this research and its applicability.

While the architecture of the neural network proposed in this paper may appear simplistic, it is intentionally designed as such to study the training process outlined within this research and isolate each component's effects on the network's performance.

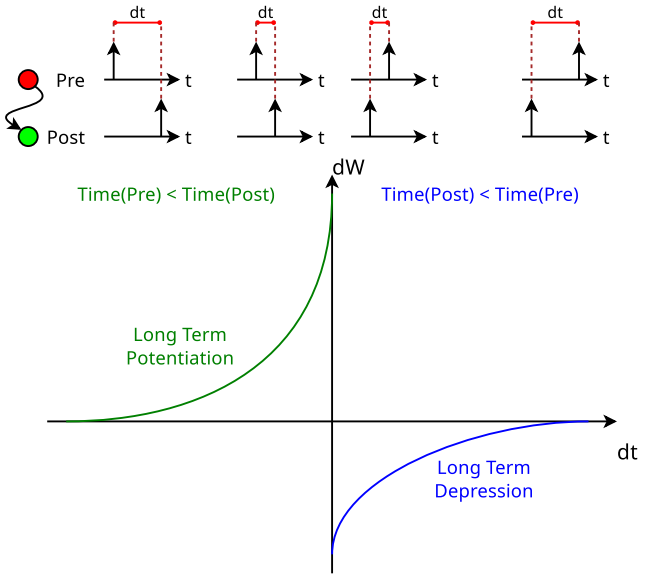


Fig. 1. An example of how weight change (dW) is calculated based on the time difference between pre- and post-synaptic spikes (dt). In green on the left is the Long Term Potentiation (LTP) generated by a pre-synaptic and post-synaptic spike sequence. In blue on the right is the Long Term Depression (LTD) generated by a post-synaptic and pre-synaptic spike sequence.

2. Methodology

In this paper we refer to spatial patterns of spikes as a set of spikes that are presented to the network from different source neurons at the same time, and whose source neuron is meaningful for the pattern.

As spatial patterns relate only to the presence or absence of a spike from a specific source, this type of patterns can be identified and encoded with the use of binary numbers, where "1"s represent the presence of a spike, while "0"s reflect its absence. These numbers represented either in their binary or decimal format will also be referred to as "code words".

The spikes used to transfer information follow two of the possible encodings suggested by Auge et al. [20], namely:

- **Time To First Spike**: This is utilised during the network training phase to ensure that the supervised learning paradigm correctly triggers the relevant side of the STDP learning rule. Depending on whether the input spike represents "0" or "1", the generated spike occurs slightly before or after the training signal.
- **Parallel binary encoding**: Since spatial patterns only correspond to the presence or absence of spikes from each source, we can represent patterns with binary numbers that are presented to the network. A binary number "1" indicates the presence of a spike from that source, while "0" represents its absence.

Two related neural networks are designed for this exercise: the first one is used to train the relevant synapses, while the second network, which is a simplified version of the training network, is used to test and validate the model obtained in the first step.

2.1. STDP on SpiNNaker

STDP is a form of learning whereby the weight of a synapse between two neurons is either potentiated (LTP) or depressed (LTD) dependent upon whether a post-synaptic spike follows or precedes a pre-synaptic spike. The size of this change, in general, drops off exponentially as the time difference between the pre- and post-synaptic spikes gets larger [42]. This is shown graphically in Fig. 1. In PyNN, STDP is defined in a modular fashion such that the user may specify which timing rule (for example, to determine the shape of the exponential decay) and weight update rule (for example, to indicate whether the weight update is additive or multiplicative) they wish to use.

This is how the rules are also implemented on SpiNNaker, with one proviso: due to local memory restrictions on how much data can be held for parameters, multiple STDP projections to the same target population must use the same rule with the same parameters.

On SpiNNaker, the plasticity mechanism for STDP is also only activated when the post-synaptic neuron receives the second (pre-synaptic) spike: at least two pre-synaptic spikes are, therefore, required for the calculations to take place. This is because the conventional method for calculating STDP at every pre-synaptic spike and every post-synaptic spike is difficult on SpiNNaker due to the synaptic weights being held in external memory and only copied into local memory when a pre-synaptic spike arrives. Thus, a deferred event-driven model is used to postpone the STDP calculation until future spike timings determine how the pre-synaptic sensitive scheme is applied [43]. Because of this deferred event-driven model, STDP weight changes can only be computed when a pre-synaptic spike is received. Therefore, to detect the effects of the sequence of spikes to the output neuron at the end of the training phase, a "save neuron" (see Figs. 2 and 3) emits a final spike whose only effect is to trigger the execution of the STDP learning rule on plastic synapses.

2.2. Training phase

The training phase relies on the network shown in Fig. 3. This network consists of two sections: one focused on the "0"s on the left, and, symmetrically, another section dedicated to the "1"s on the right.

The "Spike Source Populations" inject spikes according to specific patterns to train the network. The "Spike injector" populations comprise leaky integrate-and-fire neurons with delta synapses. These synapses have the characteristic that the current transferred to the post-synaptic neuron is applied within a single-millisecond time slot, during which it receives all the current. The neuron parameters are set to the default values provided by the PyNN [39] interface to the SpiNNaker backend simulator [40,41].

Fig. 2 shows the spike times for each neuron in the network in the case of a "0" on the left and in case of a "1" on the right. The information is encoded using the "Time to first spike" method: in case a "0" needs to be presented, the sequence of spikes generated by the Spike Source Population "0" includes spikes at 6, 36 and 59 ms, while to encode a "1" the Spike Source Population "1" emits spikes at 1, 26 and 56 ms. These spikes are propagated through to the output neuron following the time pattern in Fig. 2. Indeed, the neurons in the "Spike Injector" populations emit a spike for each spike they receive.

The output neuron receives the features of the signal from the plastic synapses (in blue) which do not contribute to the membrane potential since their weight is 0. However, they allow the output neuron to store information to trigger the STDP learning rule. This sequence of spikes has been designed considering the peculiarities of both the SpiNNaker architecture and of the software implementation of the STDP algorithm [43]. The STDP algorithm employed in this case is the nearest neighbour spike pair rule, which is only triggered when at least one pre-synaptic spike and one post-synaptic spike are already in memory at the point when an new incoming pre-synaptic spike is received. This is a custom extension for the SpiNNaker backend simulator in PyNN. The parameters used for the weight update rule are as follows: $\tau_+ = 5$, $\tau_- = 5$, $A_+ = 1$ and $A_- = -1$. However, later in this article it will be discussed that the specific values of these parameters have little relevance on the whole set of experiments.

In the model presented above, the output spike generated by the signal from the teacher neuron (in red) is always received at millisecond 31 and generates an output spike at millisecond 32. This output spike always falls between at least two pre-synaptic spikes, both in the case of a "0" and a "1", thus triggering the STDP rule in both cases. This

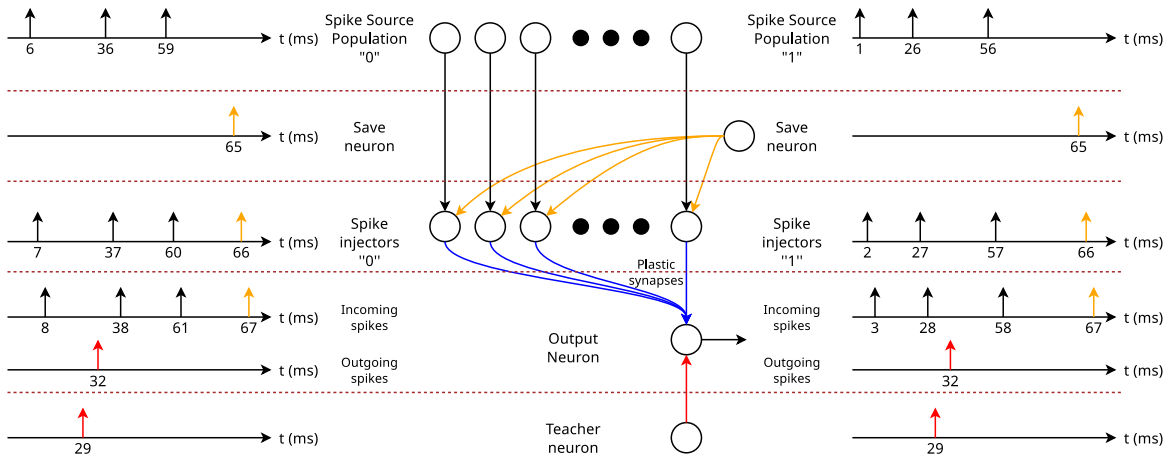


Fig. 2. The sequence of spikes in the network used for training.

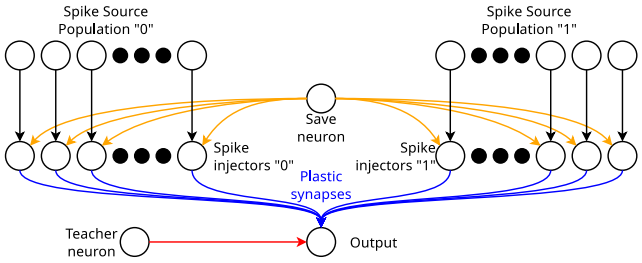


Fig. 3. The network used for training. In blue the STDP-enabled synapses.

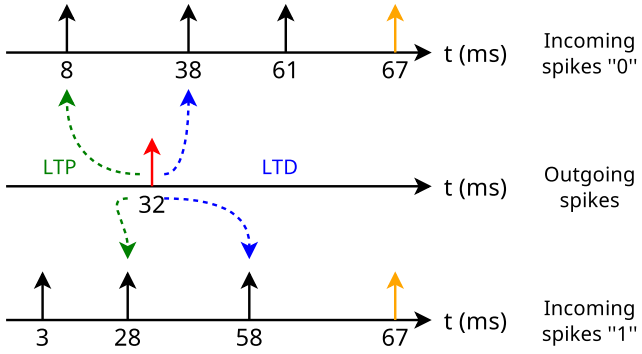


Fig. 4. The precise timing of the spikes for potentiation and depression. The timing for both Long Term Potentiation (LTP) and Long Term Depression (LTD) is considered on the timestep immediately following the outgoing spike value, so the values concerned here are 5 ms when the "1" is potentiated and the "0" is depressed, and 25 ms when the "0" is potentiated and the "1" is depressed.

is further detailed in Fig. 4, which shows which elements of the STDP rule are triggered by each spike in the network.

The precise spike times in this model have been chosen based on experimentation to ensure that the potentiation and the depression induced by the STDP rule on the plastic synapses have the same magnitude but opposite sign. Finally, the save neuron is used to allow the storage of the newly computed synaptic weight to memory, so that these can be retrieved at the end of the simulation.

Initially, synapses are set with a weight of zero, emphasising that the output spike relies solely on the contribution of the teacher neuron. This underscores that the training process exclusively depends on the activity of the teacher neuron and the STDP learning rule.

The synaptic weights obtained during this training phase are used in the testing network in Fig. 5. The input pattern is injected in

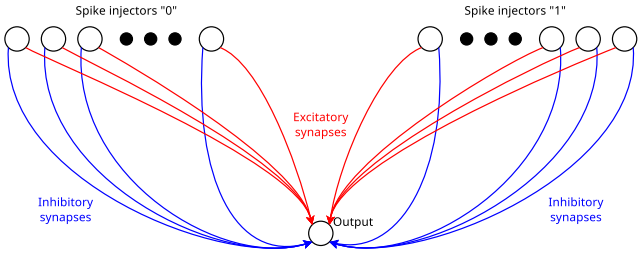


Fig. 5. The network used for testing.

this network through the "Spike Injectors 0" and "Spike Injectors 1" populations. In the first population a neuron fires if the corresponding bit of the input pattern is a "0". On the contrary, if the bit is a "1", then the corresponding neuron of the "Spike Injectors 1" population fires. In addition, all synapses are fixed, and the excitatory weights originating from "Spike injector 0" or "Spike injector 1" to the output neuron mirror the patterns learned by the synapses in the corresponding locations of the preceding network. In contrast, the inhibitory weights stemming from "Spike injector 0" to the output neuron are guided by the weights learned by "Spike injector 1", and conversely, the inhibitory weights originating from "Spike injector 1" to the output neuron are influenced by the weights learned by "Spike injector 0". Excitatory synapses from the "Spike injector 1" population and inhibitory synapses from "Spike injector 0" population have the same weight but opposite sign. The same applies to excitatory synapses from "Spike injector 0" and inhibitory synapses from "Spike injector 1" populations.

Following this pattern of connectivity:

$$W^{I}_{SI0}(n) = -W^{E}_{SI1}(n)$$
$$W^{I}_{SI1}(n) = -W^{E}_{SI0}(n)$$

where:

- $\boldsymbol{W^{E}_{SI0}(n)}$ represents the weight of the excitatory synapse from the nth neuron of the "Spike injector 0" population;
- $\boldsymbol{W^{I}_{SI0}(n)}$ represents the weight of the inhibitory synapse from the nth neuron of the "Spike injector 0" population;
- $\boldsymbol{W^{E}_{SI1}(n)}$ represents the weight of the excitatory synapse from the nth neuron of the "Spike injector 1" population;
- $\boldsymbol{W^{I}_{SI1}(n)}$ represents the weight of the inhibitory synapse from the nth neuron of the "Spike injector 1" population;

This reciprocal relationship between excitatory and inhibitory weights ensures that the output neuron is able to select the pattern or patterns to respond to. Indeed, even in the case that the network is

Table 1
Classification metrics used for the evaluation of the performance of trained networks.

Name	Variable	Description
Positives	P_i	Equals to 1 in case the network emits a spike associated with the ith input pattern
Negatives	N_i	Equals to 1 in case the network does not emit a spike associated with the ith input pattern
True positives	t_p	Number of spikes emitted associated with patterns learned by the network
True negatives	t_n	Number of patterns correctly not identified by the network
False positives	f_p	Number of spikes emitted, but not associated with patterns learned by the network
False negatives	f_n	Number of patterns on which the network was trained but that the network failed to identify
Name	Formula	Description
Accuracy	$\frac{t_p+t_n}{t_p+t_n+f_p+f_n}$	Proximity of the identification task to the training. It evaluates the overall performance of classification
Precision	$\frac{t_p}{t_p+f_p}$	Positive predicted value. This indicates the reliability of identification
Negative prediction	$\frac{t_n}{t_n+f_n}$	Reliability of classification of distractions
Sensitivity	$\frac{t_p}{t_p+f_n}$	Focuses on how good is the performance in classifying attention
Specificity	$\frac{t_n}{t_n+f_p}$	Evaluates the performance in classifying distractions

trained on multiple patterns, the output signal is generated always by the single output neuron present in the network.

In the cases where the network needs training on multiple patterns, each pattern will be trained separately, always starting from a network with plastic synaptic weights set to "0". The final value of the synaptic weights will be obtained by summing the weights resulting from training, synapse-by-synapse. The weights obtained from the previous step are used as the basis for the homeostatic process. During this step, after summing all the weights from the various training iterations, homeostasis is applied. This is modelled as a multiplier factor that re-scales the synaptic weights to obtain the minimum scaling value for each training pattern that allows the output neuron to fire exactly once for each learned pattern. To complete this step, the summed and re-scaled synaptic weights are applied to the testing network in multiple iterations, adjusting the homeostatic factor at each iteration. Each training pattern is injected into this network, obtaining a homeostatic factor that may differ for each trained pattern. Finally, the maximum among these factors is selected as the network homeostatic factor.

The search for the re-scaling factor is performed initially through a binary search in the interval between the values of 0.0001 and 1000, reducing the interval until the values between the two extremes x and y is less than or equal to 0.0001. Then the search becomes linear in the interval $[x-0.00001; y+0.00001]$ with a step equal to 0.00001, one order of magnitude smaller than the interval. The precision of this step descends from the precision of the synaptic weights on SpiNNaker. This precision is determined at runtime based upon the maximum weight value possible within the network [44]. For the network described here the minimum weight that can be represented is $2^{-11} \approx 0.0005$, so a linear search with a step size equal to 0.00001 does not reduce the precision of the network. Once the homeostatic factor is determined to allow the network to spike once for every pattern presented, the process moves to the testing and validation phase. Because the homeostatic process re-scales all weights to obtain a specific required result, the original weights obtained through the STDP is of little relevance to the whole training process.

2.3. Test and validation phase

During this phase, the objective is to validate the methodology for the training of a spatial feature classifier. To this end, we calculate the following classification metrics: Accuracy, Precision, Negative Prediction, Sensitivity, and Specificity, as presented in [45]. These metrics are employed for class identification, as detailed in Table 1.

The network utilised is depicted in Fig. 5. All neurons operate as leaky integrate-and-fire units with non-plastic delta synapses, as previously described.

3. Results

3.1. Single pattern training

To create a sufficiently broad testing space, the network in Fig. 5 undergoes testing and validation using 10-bit patterns. These patterns are represented by numbers in the range $[0; 1023]$, where their binary representation effectively reflects the combination of spikes in the pattern. The initial test focuses on the pattern expressed by the number $992_{10} = 1111100000_2$ (the subscript numbers represent the base in which the code word is expressed). Since the network comprises only 10 synapses per injector population, the details of the weights generated by the training step are documented fully in Table 2 to provide context for the discussion.

It is evident that the weights precisely mirror the pattern of "0"s and "1"s in the pattern. Subsequently, homeostasis is applied to the group of synapses to ensure that the output neuron fires once when the pattern is presented to the network. The resulting homeostasis factor is computed as 4.18817, which re-scales the weight values to those presented in Table 3.

With the weights outlined in the latter table, the network is validated using all possible combinations of spikes, showing that it produces only one output spike in response to the input $992_{10} = 1111100000_2$, in accordance with the training provided, demonstrating perfect pattern recognition, as shown in Table 4.

3.2. Dual pattern training

In addition to the single-pattern testing, a set of two-pattern training experiments has been conducted to investigate how training a single network on multiple patterns influences the recognition process. Building upon the previous experiment, this set of experiments aims to elucidate how the disparity between the two learned patterns impacts the recognition process. The first experiment aims to train the network to identify the patterns $992_{10} = 1111100000_2$ and $960_{10} = 1111000000_2$. Between the two patterns there is only one bit difference in position 5. Network training takes this difference into account both at STDP training stage and at the homeostasis adjustment stage. Indeed, the synaptic weights for the plastic synapses to the output neuron achieve the values presented in Table 5.

At a first glance, two main differences become evident when comparing these values with those related to the single-pattern experiment: in the first instance, weights of neurons 0 to 4 and 6 to 9 are doubled. This is because the two-pattern experiment sums the corresponding synapses trained independently on the two patterns. Therefore these synapses are reinforced twice, and their weights are doubled.

In the second instance, it is possible to notice that the weights of the synapses from neuron 5 is evenly distributed between the two injector populations. This distribution arises from the training process: while the pattern $992_{10} = 1111100000_2$ trained the network to detect a 1 in position 5, the pattern $960_{10} = 1111000000_2$ trained the network to detect a 0 in the same position.

This means that neither of the neurons with ID 5 contributes to the generation of the output spike. In the testing network (Fig. 5) both the

Table 2
Trained synaptic weights for a single pattern expressed by the number $992_{10} = 1111100000_2$.

Neuron ID	9	8	7	6	5	4	3	2	1	0
Population Injector "0"	0	0	0	0	0	0.367	0.367	0.367	0.367	0.367
Population Injector "1"	0.367	0.367	0.367	0.367	0.367	0	0	0	0	0

Table 3
Re-scaled synaptic weights (after homeostasis).

Neuron ID	9	8	7	6	5	4	3	2	1	0
Population Injector "0"	0	0	0	0	0	1.538	1.538	1.538	1.538	1.538
Population Injector "1"	1.538	1.538	1.538	1.538	1.538	0	0	0	0	0

Table 4
Classification metrics for the network trained with a single pattern.

Metric	Formula	Value
Homeostatic value		4.18817
Positives	P_i	1
Negatives	N_i	1023
True positives	t_p	1
True negatives	t_n	1023
False positives	f_p	0
False negatives	f_n	0
Accuracy	$\frac{t_p + t_n}{t_p + t_n + f_p + f_n}$	$\frac{1024}{1024} = 1$
Precision	$\frac{t_p}{t_p + f_p}$	$\frac{1}{1+0} = 1$
Negative prediction	$\frac{t_n}{t_n + f_n}$	$\frac{1023}{1023+0} = 1$
Sensitivity	$\frac{t_p}{t_p + f_n}$	$\frac{1}{1} = 1$
Specificity	$\frac{t_n}{t_n + f_p}$	$\frac{1023}{1023} = 1$

inhibitory and excitatory synapses from both injector populations have the same weight:

$$I_5 = W_{\text{exc5|Inj0}} - W_{\text{inh5|Inj0}} + W_{\text{exc5|Inj1}} - W_{\text{inh5|Inj1}} \quad (1)$$

$$= \underbrace{0.367 - 0.367}_{\substack{\text{for a "0"} \\ \text{spike}}} + \underbrace{0.367 - 0.367}_{\substack{\text{for a "1"} \\ \text{spike}}} = 0 \quad (2)$$

Therefore, the input current to the output neuron contributed by neuron 5 in either spike injector population is null. The homeostasis process takes this into account by increasing the overall value of the other contributing neurons by the same amount of the missing synapses. In fact, the homeostatic parameter in this instance is

$$2.32647 \approx \frac{4.18817}{2} \times \frac{10}{9} \quad (3)$$

where the value 4.18817 is the homeostatic value from the single pattern experiment, and $^{10}/_{9}$ represents the fact that one of the synapses is not contributing to the identification, and therefore all the other synapses need to be stronger.

In these conditions, it is possible to evaluate the network performance metrics in the detection of the patterns by testing all the possible combinations. In this case the network positively identifies only the two trained patterns ($992_{10} = 1111100000_2$ and $960_{10} = 1111000000_2$) behaving as the perfect classifier.

As we test the network with patterns increasingly divergent from the original $992_{10} = 1111100000_2$ pattern, several general trends become evident:

- The synapses related to the bits that are different among the two patterns have weights evenly distributed between the two injector populations. In this way, the output neuron does not depend on these inputs, which can be considered "don't care" synapses or bits.
- The homeostatic factor increases with the number of "don't care" bits, to account for the fewer synapses that contribute to the detection.
- The network's performance metrics demonstrate a noticeable deterioration as the number of dissimilarities between the patterns learned by the network increases. This degradation in performance underscores the sensitivity of the network to discrepancies among the learned patterns.
- The deterioration in the performance of the pattern recognition task is related to the Hamming Distance [46,47] between the two learned patterns: the number of "don't care" bits in the network and therefore the number of patterns that the network is able to identify.

These results can be seen in Table 6. The position of the "don't care" bits or synapses in the pattern is irrelevant for the purpose of this task. This can be seen in the cases where the Hamming Distance is 1 to 5: even though the patterns to learn are different, the classification metrics are equal and dependent only on the Hamming Distance between them.

A special mention should be made for the case of two patterns that are completely opposite. In our case $992_{10} = 1111100000_2$ and $31_{10} = 0000011111_2$ have a Hamming Distance of 10. In this scenario, none of the synapses in the network would contribute to the identification of the pattern, and therefore, the output neuron would never spike. Consequently, the homeostatic factor becomes infinite ($+\infty$). As a result, all the classification metrics become incalculable since no output spike is generated under any circumstances, and therefore this combination of patterns is not included as a result in the table.

3.3. Multiple pattern training

Finally, the network was tested with three code words. As discussed in the previous case, the accuracy of the network relies on the similarities between the various code words, and in particular on the Hamming Distance across all code words trained.

To ensure the test was conducted with the largest possible set comprising all 10-bit code words, the Hamming Distance between all possible combinations of 10-bit numbers was computed, and only the first occurrence of the code words for each Hamming Distance was recorded, where the code words were all different. The resulting combinations of code words is described in Table 9.

The network was trained using the same protocol as in previous experiments, this time using three code words. The weights obtained in this way are then re-scaled simulating an homeostatic process so that, where possible, the network would spike for all three trained patterns. However, as shown in Table 9, not in all cases this is possible: with specific combinations of code words it is not possible to have the positive identification of one or even two code words.

To explain this behaviour we can start considering each training iteration as providing a synaptic weight unit contribution to the "0" population or to the "1" population. When the code word is then re-applied during homeostasis, these unit weights contribute to the

Table 5
Trained synaptic weights for two patterns expressed by numbers $992_{10} = 1111100000_2$ and $960_{10} = 1111000000_2$.

Neuron ID	9	8	7	6	5	4	3	2	1	0
Population Injector "0"	0	0	0	0	0.367	0.734	0.734	0.734	0.734	0.734
Population Injector "1"	0.734	0.734	0.734	0.734	0.367	0	0	0	0	0

Table 6
Classification metrics for the network trained with two patterns.

Pattern 1	992_{10}	992_{10}	992_{10}	992_{10}	992_{10}	992_{10}	992_{10}	992_{10}	992_{10}	992_{10}	992_{10}	992_{10}	992_{10}	992_{10}
Pattern 2	1008_{10}	1016_{10}	1020_{10}	1022_{10}	1023_{10}	960_{10}	896_{10}	768_{10}	512_{10}	0_{10}	16_{10}	24_{10}	28_{10}	30_{10}
Hamming distance	1	2	3	4	5	1	2	3	4	5	6	7	8	9
Homeostasis factor	2.3265	2.6177	2.9914	3.4907	4.1875	2.3265	2.6177	2.9914	3.4907	4.1875	5.2354	6.9814	10.4708	20.9415
Positives	2	4	8	16	32	2	4	8	16	32	64	128	256	512
Negatives	1022	1020	1016	1008	992	1022	1020	1016	1008	992	960	896	768	512
True positives	2	2	2	2	2	2	2	2	2	2	2	2	2	2
True negatives	1022	1020	1016	1008	992	1022	1020	1016	1008	992	960	896	768	512
False positives	0	2	6	14	30	0	2	6	14	30	62	126	254	510
False negatives	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Accuracy	1	0.998	0.994	0.986	0.971	1	0.998	0.994	0.986	0.971	0.939	0.877	0.752	0.502
Precision	1	0.5	0.25	0.125	0.0625	1	0.5	0.25	0.125	0.0625	0.03125	0.0156	0.00781	0.00391
Negative prediction	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Sensitivity	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Specificity	1	0.998	0.994	0.986	0.971	1	0.998	0.994	0.986	0.971	0.939	0.877	0.751	0.501

Table 7
Trained unit synaptic weights for the three code words "0", "1" and "2".

	Population "0"									
Neuron ID	10	9	8	7	6	5	4	3	2	1
Code word "0" unit synaptic contributions	1	1	1	1	1	1	1	1	1	1
Code word "1" unit synaptic contribution	1	1	1	1	1	1	1	1	1	0
Code word "2" unit synaptic contribution	1	1	1	1	1	1	1	1	0	1
Final unit synaptic weights	3	3	3	3	3	3	3	3	2	2
	Population "1"									
Neuron ID	10	9	8	7	6	5	4	3	2	1
Code word "0" unit synaptic contributions	0	0	0	0	0	0	0	0	0	0
Code word "1" unit synaptic contribution	0	0	0	0	0	0	0	0	0	1
Code word "2" unit synaptic contribution	0	0	0	0	0	0	0	0	1	0
Final unit synaptic weights	0	0	0	0	0	0	0	0	1	1

Table 8
Example of synaptic unit contributions for the three code words "0", "1" and "2", coloured numbers indicate the source of the corresponding weight from Table 7.

Code word	Synaptic contribution	Unit weight
"0"	3 + 3 + 3 + 3 + 3 + 3 + 3 + 3 + 2 + 2 − (0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 1 + 1)	26
"1"	3 + 3 + 3 + 3 + 3 + 3 + 3 + 3 + 2 + 1 − (0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 1 + 2)	24
"2"	3 + 3 + 3 + 3 + 3 + 3 + 3 + 3 + 1 + 2 − (0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 2 + 1)	24

final weight applied to the output neuron. If the sum of excitatory and inhibitory unit weights is positive, then the homeostasis finds the smallest factor for which the network fires for all the code words. In alternative, one or two code words are "discarded" during this process and will appear in the "False Negative" count.

For example, in the case of code words "0", "1" and "2", the final weight unit contribution to the output neuron is described in Table 7:

As for the code word 0 all the neurons in population "0" are spiking, these neuron contribute to the output neuron by exciting it through the excitatory "0" synapses and inhibit it through the inhibitory "1" synapses (see Fig. 5). In this case the final contribution is positive and includes 26 unit weights. If we repeat this process for all the code words in the example, the synaptic weights contributions are as described in Table 8, where coloured numbers indicate the source of the corresponding weight from Table 7.

As all the contributions are positive, in this case applying the appropriate homeostatic factor to the weights will lead to the output neuron firing (at least) for the trained code words. Considering the STDP parameters applied to the network, one synaptic weight unit is equal to 0.3671875 and the homeostatic factor required for the network is estimated in this case equal to 1.74513.

The table describing all the combinations of code words, their Hamming Distance, the synaptic unit weights and homeostasis factors computed for this case is presented in Table 9.

Table 10 introduces the performance metrics of the test and validation network, and clearly illustrates two trends: maintaining two code words constant while progressively increasing the Hamming Distance of the remaining one from the others results in the deterioration of synaptic unit weights. These weights reach zero or become negative, rendering one or more code word(s) no longer positively identifiable by the network.

Simultaneously, as the synaptic unit weight decreases, the homeostatic factor increases to compensate for the limited efficacy of the incoming excitation. This holds true until the network is no longer able to detect one of the code words, which is then automatically excluded from the identification task during the search for a valid homeostatic factor: indeed the neural network cannot emit a spike in case the output neuron excitation results in a "0" or even a negative number. The unit weights for which one or two code words are no longer identifiable are highlighted in red in Table 9.

Table 9
Combinations of code words (CW1, CW2 and CW3) trained on the network, their Hamming Distance HD(x,y), the synaptic unit weight, computed as described in the text, and the resulting homeostatic factor.

CW1	CW2	CW3	HD (1,2)	HD (1,3)	HD (2,3)	Code word 1 Unit weight	Code word 2 Unit weight	Code word 3 Unit weight	Homeostatic factor
0	1	2	1	1	2	26	24	24	1.74513
0	1	6	1	2	3	24	22	20	2.09442
0	1	14	1	3	4	22	20	16	2.61791
0	1	30	1	4	5	20	18	12	3.49203
0	1	62	1	5	6	18	16	8	5.23937
0	1	126	1	6	7	16	14	4	10.47873
0	1	254	1	7	8	14	12	0	3.4907
0	1	510	1	8	9	12	10	−4	4.19016
0	1	1022	1	9	10	10	8	−8	5.23804
0	3	5	2	2	2	22	22	22	1.90382
0	3	12	2	2	4	22	18	18	2.32669
0	3	13	2	3	3	20	20	18	2.32713
0	3	28	2	3	5	20	16	14	2.99203
0	3	29	2	4	4	18	18	14	2.99203
0	3	60	2	4	6	18	14	10	4.18883
0	3	61	2	5	5	16	16	10	4.18618
0	3	124	2	5	7	16	12	6	6.98405
0	3	125	2	6	6	14	14	6	6.98405
0	3	252	2	6	8	14	10	2	21
0	3	253	2	7	7	12	12	2	21
0	3	508	2	7	9	12	8	−2	5.23671
0	3	509	2	8	8	10	10	−2	4.18883
0	3	1020	2	8	10	10	6	−6	6.98139
0	3	1021	2	9	9	8	8	−6	5.23671
0	7	25	3	3	4	18	16	16	2.61791
0	7	56	3	3	6	18	12	12	3.49025
0	7	57	3	4	5	16	14	12	3.49025
0	7	120	3	4	7	16	10	8	5.23582
0	7	121	3	5	6	14	12	8	5.23582
0	7	248	3	5	8	14	8	4	10.47873
0	7	249	3	6	7	12	10	4	10.47873
0	7	504	3	6	9	12	6	0	6.98139
0	7	505	3	7	8	10	8	0	5.23671
0	7	1016	3	7	10	10	4	−4	10.47607
0	7	1017	3	8	9	8	6	−4	6.98139
0	15	51	4	4	4	14	14	14	2.99203
0	15	113	4	4	6	14	10	10	4.18972
0	15	115	4	5	5	12	12	10	4.18972
0	15	240	4	4	8	14	6	6	6.9805
0	15	241	4	5	7	12	8	6	6.9805
0	15	243	4	6	6	10	10	6	6.9805
0	15	496	4	5	9	12	4	2	20.95745
0	15	497	4	6	8	10	6	2	20.95745
0	15	499	4	7	7	8	8	2	20.95745
0	15	1008	4	6	10	10	2	−2	20.95213
0	15	1009	4	7	9	8	4	−2	10.47341
0	15	1011	4	8	8	6	6	−2	6.98139
0	31	227	5	5	6	10	8	8	5.23582
0	31	481	5	5	8	10	4	4	10.47164
0	31	483	5	6	7	8	6	4	10.47164
0	31	992	5	5	10	10	0	0	4.19016
0	31	993	5	6	9	8	2	0	20.94681
0	31	995	5	7	8	6	4	0	10.47341
0	63	455	6	6	6	6	6	6	6.98139
0	63	963	6	6	8	6	2	2	20.94681
0	63	967	6	7	7	4	4	2	20.94681

Comparing Table 9 with Table 10, it is possible to notice that the false negatives appear in correspondence to the 0 or negative unit weights. These classification metrics show that the accuracy is linked to the Hamming Distance between code words. However, when one code word is dropped from identification, the negative prediction and specificity parameters receive lower values, but the overall accuracy improves, as the number of false positive identifications drastically reduces. However, the precision parameter decreases rapidly as the number of false positives increases.

The classification metrics presented in Table 10 are also displayed graphically in Fig. 6. In these graphs it is possible to notice how the closer the experiments are to the bottom left corner, the better the classification metrics that represent the outcome. Indeed, the bottom left corner represents experiments using three code words whose Hamming Distance among them is minimum. In these graphs the three axis HD(x,y) indicate the Hamming Distance between the "x" and "y" code words.

In particular it is possible to notice also how there is an inverse relation between the homeostatic factor and the overall accuracy of the identification task. As the distance between code words increases, the number of synapses with overall contribution "0" ("don't care" synapses) or small ("care little" synapses) increases, and these require a higher homeostatic factor to allow the remaining synapses to trigger an output spike. However, this also causes the number of false positive

Table 10
Combinations of code words (CW1, CW2 and CW3) trained on the network, and the corresponding test classification metrics obtained from simulations.

CW1	CW2	CW3	True positives	True negatives	False positives	False negatives	Accuracy	Precision	Negative prediction	Sensitivity	Specificity
0	1	2	3	1021	0	0	1.000	1.000	1.000	1.000	1.000
0	1	6	3	1017	4	0	0.996	0.429	1.000	1.000	0.996
0	1	14	3	1003	18	0	0.982	0.143	1.000	1.000	0.982
0	1	30	3	963	58	0	0.943	0.049	1.000	1.000	0.943
0	1	62	3	873	148	0	0.855	0.020	1.000	1.000	0.855
0	1	126	3	705	316	0	0.691	0.009	1.000	1.000	0.690
0	1	254	2	1014	7	1	0.992	0.222	0.999	0.667	0.993
0	1	510	2	1013	8	1	0.991	0.200	0.999	0.667	0.992
0	1	1022	2	1012	9	1	0.990	0.182	0.999	0.667	0.991
0	3	5	3	1020	1	0	0.999	0.750	1.000	1.000	0.999
0	3	12	3	1013	8	0	0.992	0.273	1.000	1.000	0.992
0	3	13	3	1013	8	0	0.992	0.273	1.000	1.000	0.992
0	3	28	3	998	23	0	0.978	0.115	1.000	1.000	0.977
0	3	29	3	998	23	0	0.978	0.115	1.000	1.000	0.977
0	3	60	3	963	58	0	0.943	0.049	1.000	1.000	0.943
0	3	61	3	963	58	0	0.943	0.049	1.000	1.000	0.943
0	3	124	3	817	204	0	0.801	0.014	1.000	1.000	0.800
0	3	125	3	817	204	0	0.801	0.014	1.000	1.000	0.800
0	3	252	3	591	430	0	0.580	0.007	1.000	1.000	0.579
0	3	253	3	591	430	0	0.580	0.007	1.000	1.000	0.579
0	3	508	2	977	44	1	0.956	0.043	0.999	0.667	0.957
0	3	509	2	1013	8	1	0.991	0.200	0.999	0.667	0.992
0	3	1020	2	967	54	1	0.946	0.036	0.999	0.667	0.947
0	3	1021	2	1012	9	1	0.990	0.182	0.999	0.667	0.991
0	7	25	3	1008	13	0	0.987	0.188	1.000	1.000	0.987
0	7	56	3	982	39	0	0.962	0.071	1.000	1.000	0.962
0	7	57	3	982	39	0	0.962	0.071	1.000	1.000	0.962
0	7	120	3	922	99	0	0.903	0.029	1.000	1.000	0.903
0	7	121	3	922	99	0	0.903	0.029	1.000	1.000	0.903
0	7	248	3	787	234	0	0.771	0.013	1.000	1.000	0.771
0	7	249	3	787	234	0	0.771	0.013	1.000	1.000	0.771
0	7	504	2	892	129	1	0.873	0.015	0.999	0.667	0.874
0	7	505	2	977	44	1	0.956	0.043	0.999	0.667	0.957
0	7	1016	2	847	174	1	0.829	0.011	0.999	0.667	0.830
0	7	1017	2	967	54	1	0.946	0.036	0.999	0.667	0.947
0	15	51	3	1002	19	0	0.981	0.136	1.000	1.000	0.981
0	15	113	3	957	64	0	0.938	0.045	1.000	1.000	0.937
0	15	115	3	957	64	0	0.938	0.045	1.000	1.000	0.937
0	15	240	3	859	162	0	0.842	0.018	1.000	1.000	0.841
0	15	241	3	859	162	0	0.842	0.018	1.000	1.000	0.841
0	15	243	3	859	162	0	0.842	0.018	1.000	1.000	0.841
0	15	496	3	632	389	0	0.620	0.008	1.000	1.000	0.619
0	15	497	3	632	389	0	0.620	0.008	1.000	1.000	0.619
0	15	499	3	632	389	0	0.620	0.008	1.000	1.000	0.619
0	15	1008	2	637	384	1	0.624	0.005	0.998	0.667	0.624
0	15	1009	2	847	174	1	0.829	0.011	0.999	0.667	0.830
0	15	1011	2	967	54	1	0.946	0.036	0.999	0.667	0.947
0	31	227	3	931	90	0	0.912	0.032	1.000	1.000	0.912
0	31	481	3	767	254	0	0.752	0.012	1.000	1.000	0.751
0	31	483	3	767	254	0	0.752	0.012	1.000	1.000	0.751
0	31	992	1	1021	0	2	0.998	1.000	0.998	0.333	1.000
0	31	993	2	637	384	1	0.624	0.005	0.998	0.667	0.624
0	31	995	2	847	174	1	0.829	0.011	0.999	0.667	0.830
0	63	455	3	893	128	0	0.875	0.023	1.000	1.000	0.875
0	63	963	3	638	383	0	0.626	0.008	1.000	1.000	0.625
0	63	967	3	638	383	0	0.626	0.008	1.000	1.000	0.625

identifications to increase, which in turn reduces the overall accuracy of the identification task of the network.

4. Conclusions

This paper presented a method for training a spiking neural network to identify spatial patterns. Validation results show that a spiking network trained this way is able to successfully complete this task. The testing involved training on a single pattern, two patterns and three patterns, covering all meaningful combinations of code words. In these experiments, it became evident that specific parameter values for the STDP learning rule hold little significance. This is because the homeostatic process plays a crucial role in re-scaling synaptic weights to elicit spikes from the output neuron under specific conditions.

The results of the experiments show that a network trained with a single pattern acts as a perfect classifier, which only identifies spatial patterns that are identical to the trained one. The ability of the network to select a specific pattern comes from the structure of the network: while excitatory synapses trigger the output neuron, inhibitory connections perform the selection of the pattern during the testing and validation phase.

When two or more patterns are trained on a single network, the accuracy of the identification task depends only on the Hamming Distance between the code words imprinted in the network.

From the analysis of the classification metrics, it is possible to see that the negative prediction in the case of one or two trained patterns is always 1, which highlights how the absence of an output spike always correctly identifies that the trained pattern is not present.

On the other hand, positive identification spikes depend on the number of “don’t care” and “care little” synapses or bits imprinted in the network. As the number of “don’t care” or “care little” synapses

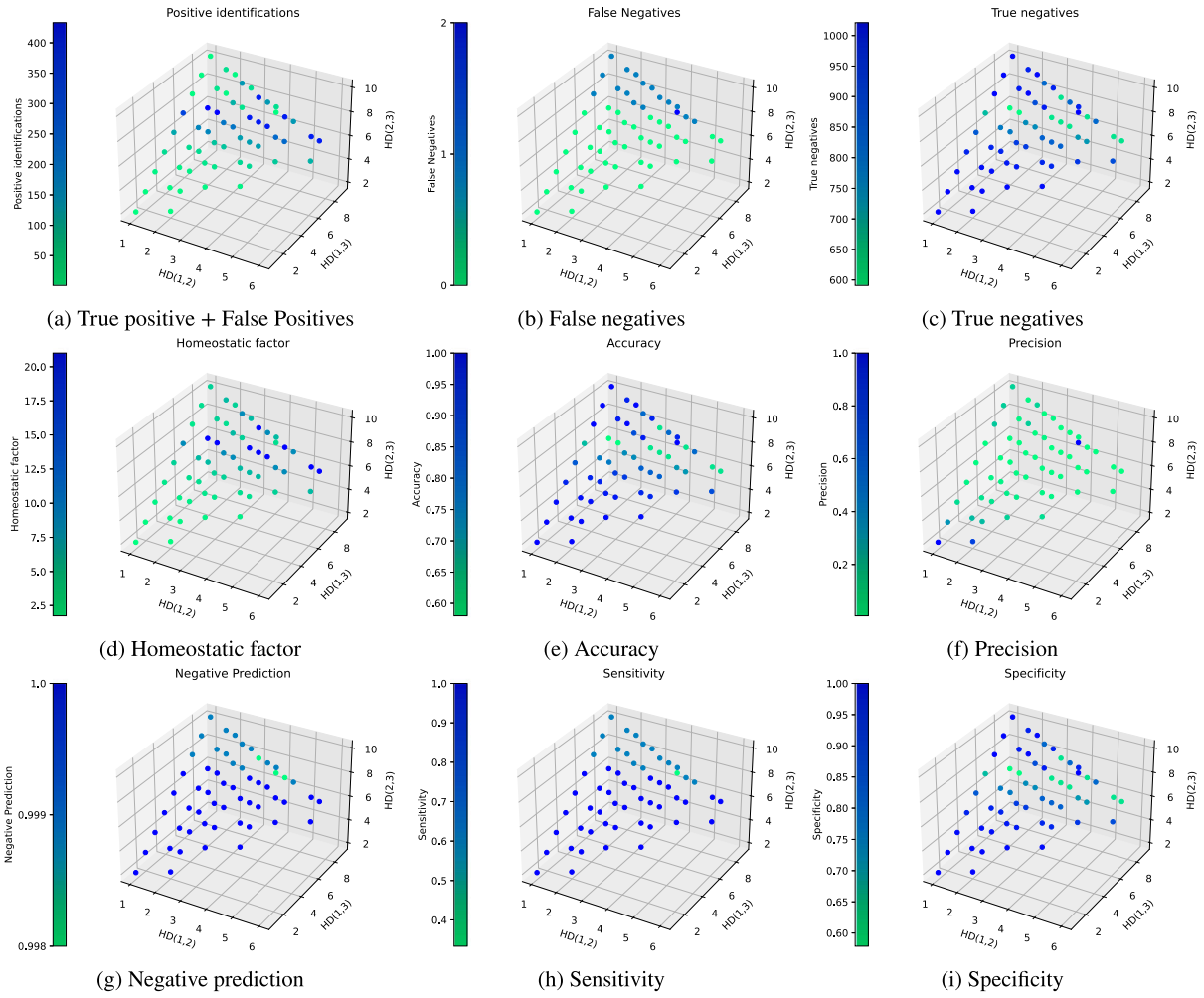


Fig. 6. Statistical classification metrics for the multiple code words training. The axes represent the Hamming Distance between the various code words as indicated in each graph.

increases, the number of false positives increases, and this reduces the overall accuracy of the identification task.

In the multiple code word experiments, the classification metrics extracted show that the accuracy is linked to the Hamming Distance between code words. However, when one code word is dropped from identification because its Hamming Distance is too high, the negative prediction and specificity parameters receive lower values, but the overall accuracy improves because the number of false positive identifications drastically reduces. Overall, precision and sensitivity decrease rapidly as the number of false positives increases when the Hamming Distance between code words increases.

This article has presented a method to train a spiking neural network to detect spatial patterns which do not have a temporal component. These kind of patterns may be found, for example, in the analysis of computer network traffic packets, where each single packet does not have a temporal component, but holds enough information for a neural network to determine the type of traffic that is carries [e.g. 48,49]. Performing such analysis would require to extract bits of information from the packet (from the header and/or from the payload) and encoding such information following the methodology presented above. Following the training, this methodology allows the extraction of features ideal for packet classification. Moreover, using multiple of these classifiers and considering that traffic streams have a typical sequence of packets in a stream, by using a technique such as polychronization [50] it would be possible to identify the sequence and determine to which stream it belongs.

An additional example of spatial pattern is represented by static images which can be encoded into spikes to allow a spiking neural network to perform pattern matching tasks on trained patterns [e.g. 45].

In these applications, it is conceivable to adjust the homeostatic factor, calculated during the training phase, to also serve as a similarity factor in matching the input pattern with the trained pattern. Even in the case of a single trained pattern, increasing the homeostatic factor appropriately may enable the network to identify patterns with a certain degree of similarity, rather than requiring an exact match between the input and the trained pattern.

CRediT authorship contribution statement

Sergio Davies: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Funding acquisition, Data curation, Conceptualization. **Andrew Gait:** Writing – review & editing, Writing – original draft, Visualization, Software, Methodology. **Andrew Rowley:** Writing – review & editing, Software. **Alessandro Di Nuovo:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data and code availability

The data and the code used to generate this article are available both on the MMU data storage servers [51] using the DOI: 10.23634/MMU.00634935, and on GitHub at the URL: https://github.com/sergiodavies/SpiNNakerSpatialLearningCodeAndDataset.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work the authors used ChatGPT in order to improve readability and language. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

Acknowledgements

For the purpose of open access, the authors have applied a Creative Common Attribution (CC BY) licence to any Author Accepted Manuscript version arising from this submission.

This work has been supported by the Department of Computing and Mathematics at the Manchester Metropolitan University and authorised through ethical review 59273.

Development of SpiNNaker software was supported by the EU ICT Flagship Human Brain Project which has received funding from the European Union's FP7 programme under Grant Agreement no. 604102, and from the European Union's Horizon 2020 research and innovation programme under FPA No 650003 (HBP-785907).

Prof. Di Nuovo acknowledges the support of the UK Engineering and Physical Sciences Research Council (grant number EP/X018733/1 for the project ALDENS), and Innovate UK (grant number 10089807 for the Horizon Europe project PRIMI Grant agreement n. 101120727).

The authors extend their gratitude to Prof. Steve Furber and all members, current and past, of the APT group at the University of Manchester for their invaluable support.

References

[1] A. Prieto, B. Prieto, E.M. Ortigosa, E. Ros, F. Pelayo, J. Ortega, I. Rojas, Neural networks: An overview of early research, current frameworks and new challenges, Neurocomputing 214 (2016) 242–268, http://dx.doi.org/10.1016/J.NEUCOM.2016.06.014, URL: https://linkinghub.elsevier.com/retrieve/pii/S0925231216305550.

[2] S.S. Haykin, Neural Networks: A Comprehensive Foundation, second ed., Prentice Hall, 1999, URL: https://archive.org/details/neuralnetworksco0000hayk_2ed.

[3] W. Maass, Networks of spiking neurons: The third generation of neural network models, Neural Netw. 10 (9) (1997) 1659–1671, http://dx.doi.org/10.1016/S0893-6080(97)00011-7, URL: https://linkinghub.elsevier.com/retrieve/pii/S0893608097000117.

[4] W.S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, Bull. Math. Biophys. 5 (4) (1943) 115–133, http://dx.doi.org/10.1007/BF02478259, URL: http://link.springer.com/10.1007/BF02478259.

[5] S. Linnainmaa, Taylor expansion of the accumulated rounding error, BIT 16 (2) (1976) 146–160, http://dx.doi.org/10.1007/BF01931367, URL: https://link.springer.com/article/10.1007/BF01931367.

[6] J. Schmidhuber, Deep Learning in neural networks: An overview, Neural Netw. 61 (2015) 85–117, http://dx.doi.org/10.1016/j.neunet.2014.09.003, URL: https://www.sciencedirect.com/science/article/pii/S0893608014002135.

[7] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444, http://dx.doi.org/10.1038/nature14539, URL: http://www.ncbi.nlm.nih.gov/pubmed/26017442.

[8] P. Sermanet, K. Kavukcuoglu, S. Chintala, Y. Lecun, Pedestrian detection with unsupervised multi-stage feature learning, in: 2013 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2013, pp. 3626–3633, http://dx.doi.org/10.1109/CVPR.2013.465, URL: http://ieeexplore.ieee.org/document/6619309/.

[9] A.-r. Mohamed, G.E. Dahl, G. Hinton, Acoustic modeling using deep belief networks, IEEE Trans. Audio Speech Lang. Process. 20 (1) (2012) 14–22, http://dx.doi.org/10.1109/TASL.2011.2109382, URL: http://ieeexplore.ieee.org/document/5704567/.

[10] G.E. Dahl, D. Yu, L. Deng, A. Acero, Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition, IEEE Trans. Audio Speech Lang. Process. 20 (1) (2012) 30–42, http://dx.doi.org/10.1109/TASL.2011.2134090, URL: http://ieeexplore.ieee.org/document/5740583/.

[11] K.-i. Funahashi, Y. Nakamura, Approximation of dynamical systems by continuous time recurrent neural networks, Neural Netw. 6 (6) (1993) 801–806, http://dx.doi.org/10.1016/S0893-6080(05)80125-X, URL: https://linkinghub.elsevier.com/retrieve/pii/S089360800580125X.

[12] Y. Yamashita, J. Tani, Emergence of functional hierarchy in a multiple timescale neural network model: A humanoid robot experiment, PLoS Comput. Biol. 4 (11) (2008) e1000220, http://dx.doi.org/10.1371/JOURNAL.PCBI.1000220, URL: https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1000220.

[13] W. Gerstner, W.M. Kistler, R. Naud, L. Paninski, Neuronal Dynamics, Cambridge University Press, 2014, http://dx.doi.org/10.1017/CBO9781107447615, URL: https://www.cambridge.org/core/product/identifier/9781107447615/type/book.

[14] N.K. Kasabov, Time-Space, Spiking Neural Networks and Brain-Inspired Artificial Intelligence, in: Springer Series on Bio- and Neurosystems, vol. 7, Springer Berlin Heidelberg, Berlin, Heidelberg, 2019, http://dx.doi.org/10.1007/978-3-662-57715-8, URL: http://link.springer.com/10.1007/978-3-662-57715-8.

[15] A.L. Hodgkin, A.F. Huxley, A quantitative description of membrane current and its application to conduction and excitation in nerve, J. Physiol. 117 (4) (1952) 500–544, http://dx.doi.org/10.1113/jphysiol.1952.sp004764, URL: http://www.ncbi.nlm.nih.gov/pubmed/12991237.

[16] E. Izhikevich, Which model to use for cortical spiking neurons? IEEE Trans. Neural Netw. 15 (5) (2004) 1063–1070, http://dx.doi.org/10.1109/TNN.2004.832719, URL: http://ieeexplore.ieee.org/document/1333071/.

[17] E. Izhikevich, Simple model of spiking neurons, IEEE Trans. Neural Netw. 14 (6) (2003) 1569–1572, http://dx.doi.org/10.1109/TNN.2003.820440, URL: http://ieeexplore.ieee.org/document/1257420/.

[18] L. Lapicque, Recherches quantitatives sur l'excitation electrique des nerfs traitee comme une polarization, J. Physiol. Pathol. Gén. 9 (1907) 620–635, URL: https://fr.wikisource.org/wiki/Recherches_quantitatives_sur_l%27excitation_%C3%A9lectrique_des_nerfs_trait%C3%A9e_comme_une_polarisation.

[19] P. Dayan, L.F. Abbott, Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems, first ed., The MIT Press, 2001, URL: https://mitpress.mit.edu/9780262041997/theoretical-neuroscience/.

[20] D. Auge, J. Hille, E. Mueller, A. Knoll, A survey of encoding techniques for signal processing in spiking neural networks, Neural Process. Lett. 53 (6) (2021) 4693–4710, http://dx.doi.org/10.1007/S11063-021-10562-2, URL: https://link.springer.com/article/10.1007/s11063-021-10562-2.

[21] A. Tavanaei, M. Ghodrati, S.R. Kheradpisheh, T. Masquelier, A. Maida, Deep learning in spiking neural networks, Neural Netw. 111 (2019) 47–63, http://dx.doi.org/10.1016/j.neunet.2018.12.002, URL: https://linkinghub.elsevier.com/retrieve/pii/S0893608018303332.

[22] R.S. Zucker, W.G. Regehr, Short-term synaptic plasticity, Annu. Rev. Physiol. 64 (1) (2002) 355–405, http://dx.doi.org/10.1146/annurev.physiol.64.092501.114547, URL: https://www.annualreviews.org/doi/10.1146/annurev.physiol.64.092501.114547.

[23] H. Markram, J. Lübke, M. Frotscher, B. Sakmann, Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs, Science 275 (5297) (1997) 213–215, http://dx.doi.org/10.1126/science.275.5297.213, URL: https://www.science.org/doi/10.1126/science.275.5297.213.

[24] G.-q. Bi, M.-m. Poo, Synaptic modification by correlated activity: Hebb's postulate revisited, Annu. Rev. Neurosci. 24 (1) (2001) 139–166, http://dx.doi.org/10.1146/annurev.neuro.24.1.139, URL: http://www.annualreviews.org/doi/10.1146/annurev.neuro.24.1.139.

[25] G.G. Turrigiano, Homeostatic plasticity in neuronal networks: the more things change, the more they stay the same, Trends Neurosci. 22 (5) (1999) 221–227, http://dx.doi.org/10.1016/S0166-2236(98)01341-1, URL: https://linkinghub.elsevier.com/retrieve/pii/S0166223698013411.

[26] G. Turrigiano, Too many cooks? Intrinsic and synaptic homeostatic mechanisms in cortical circuit refinement, Annu. Rev. Neurosci. 34 (1) (2011) 89–103, http://dx.doi.org/10.1146/annurev-neuro-060909-153238, URL: https://www.annualreviews.org/doi/10.1146/annurev-neuro-060909-153238.

[27] N. Zecevic, P. Rakic, Synaptogenesis in monkey somatosensory cortex, Cerebral Cortex 1 (6) (1991) 510–523, http://dx.doi.org/10.1093/cercor/1.6.510, URL: https://academic.oup.com/cercor/article-lookup/doi/10.1093/cercor/1.6.510.

[28] A. Maffei, A. Fontanini, Network homeostasis: a matter of coordination, Curr. Opin. Neurobiol. 19 (2) (2009) 168–173, http://dx.doi.org/10.1016/j.conb.2009.05.012, URL: http://www.ncbi.nlm.nih.gov/pubmed/19540746.

[29] D.O. Hebb, The Organization of Behavior: A Neuropsychological Theory, Wiley, 1949, URL: https://archive.org/details/in.ernet.dli.2015.168156.

[30] R. Zeraati, V. Priesemann, A. Levina, Self-organization toward criticality by synaptic plasticity, Front. Phys. 9 (2021) http://dx.doi.org/10.3389/fphy.2021.619661, URL: https://www.frontiersin.org/articles/10.3389/fphy.2021.619661.
[31] S. Nolfi, D. Parisi, J.L. Elman, Learning and evolution in neural networks, Adapt. Behav. 3 (1) (1994) 5–28, http://dx.doi.org/10.1177/105971239400300102, URL: http://journals.sagepub.com/doi/10.1177/105971239400300102.
[32] M. Tsodyks, S. Wu, Short-term synaptic plasticity, Scholarpedia 8 (10) (2013) 3153, http://dx.doi.org/10.4249/scholarpedia.3153, URL: http://www.scholarpedia.org/article/Short-term_synaptic_plasticity.
[33] A.J. Tierney, The evolution of learned and innate behavior: Contributions from genetics and neurobiology to a theory of behavioral evolution, Anim. Learn. Behav. 14 (4) (1986) 339–348, http://dx.doi.org/10.3758/BF03200077, URL: https://link.springer.com/article/10.3758/BF03200077.
[34] S. Davies, Learning in Spiking Neural Networks (Ph.D. thesis), The University of Manchester, Kilburn Building, Oxford road, M13 9PL, 2013, p. 177, URL: https://apt.cs.manchester.ac.uk/people/daviess/thesis.pdf https://research.manchester.ac.uk/en/studentTheses/learning-in-spiking-neural-networks.
[35] R. Guyonneau, R. VanRullen, S.J. Thorpe, Neurons tune to the earliest spikes through STDP, Neural Comput. 17 (4) (2005) 859–879, http://dx.doi.org/10.1162/0899766053429390, URL: https://direct.mit.edu/neco/article/17/4/859-879/6942.
[36] T. Masquelier, R. Guyonneau, S.J. Thorpe, Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains, PLoS ONE 3 (1) (2008) e1377, http://dx.doi.org/10.1371/journal.pone.0001377, URL: https://dx.plos.org/10.1371/journal.pone.0001377.
[37] S. Davies, F. Galluppi, A. Rast, S. Furber, A forecast-based STDP rule suitable for neuromorphic implementation, Neural Netw. 32 (2012) 3–14, http://dx.doi.org/10.1016/j.neunet.2012.02.018, URL: https://linkinghub.elsevier.com/retrieve/pii/S0893608012000470.
[38] S. Furber, P. Bogdan (Eds.), SpiNNaker: A Spiking Neural Network Architecture, Now Publishers, Boston-Delft, 2020, http://dx.doi.org/10.1561/9781680836523, URL: https://nowpublishers.com/article/BookDetails/9781680836523.
[39] A.P. Davison, D. Brüderle, J. Eppler, J. Kremkow, E. Muller, D. Pecevski, L. Perrinet, P. Yger, PyNN: a common interface for neuronal network simulators, Front. Neuroinform. 2 (2009) http://dx.doi.org/10.3389/neuro.11.011.2008, URL: http://journal.frontiersin.org/article/10.3389/neuro.11.011.2008.
[40] O. Rhodes, P.A. Bogdan, C. Brenninkmeijer, S. Davidson, D. Fellows, A. Gait, D.R. Lester, M. Mikaitis, L.A. Plana, A.G.D. Rowley, A.B. Stokes, S.B. Furber, sPyNNaker: A software package for running PyNN simulations on SpiNNaker, Front. Neurosci. 12 (2018) 816, http://dx.doi.org/10.3389/fnins.2018.00816, URL: https://www.frontiersin.org/article/10.3389/fnins.2018.00816.
[41] A.G.D. Rowley, C. Brenninkmeijer, S. Davidson, D. Fellows, A. Gait, D.R. Lester, L.A. Plana, O. Rhodes, A.B. Stokes, S.B. Furber, SpiNNTools: The execution engine for the SpiNNaker platform, Front. Neurosci. 13 (2019) http://dx.doi.org/10.3389/fnins.2019.00231, URL: https://www.frontiersin.org/article/10.3389/fnins.2019.00231.
[42] J. Sjöström, W. Gerstner, Spike-timing dependent plasticity, Scholarpedia 5 (2) (2010) 1362, http://dx.doi.org/10.4249/scholarpedia.1362, URL: http://www.scholarpedia.org/article/Spike-timing_dependent_plasticity.
[43] X. Jin, A. Rast, F. Galluppi, S. Davies, S. Furber, Implementing spike-timing-dependent plasticity on SpiNNaker neuromorphic hardware, in: The 2010 International Joint Conference on Neural Networks, IJCNN, Sch. of Comput. Sci., Univ. of Manchester, IEEE, Manchester, UK, 2010, pp. 1–8, http://dx.doi.org/10.1109/IJCNN.2010.5596372, URL: http://ieeexplore.ieee.org/document/5596372/.
[44] S.J. van Albada, A.G. Rowley, J. Senk, M. Hopkins, M. Schmidt, A.B. Stokes, D.R. Lester, M. Diesmann, S.B. Furber, Performance comparison of the digital neuromorphic hardware SpiNNaker and the neural network simulation software NEST for a full-scale cortical microcircuit model, Front. Neurosci. 12 (MAY) (2018) http://dx.doi.org/10.3389/fnins.2018.00291, URL: https://www.frontiersin.org/article/10.3389/fnins.2018.00291.
[45] S. Davies, A. Lucas, C. Ricolfe-Viala, A. Di Nuovo, A database for learning numbers by visual finger recognition in developmental neuro-robotics, Front. Neurorobot. 15 (2021) 12, http://dx.doi.org/10.3389/fnbot.2021.619504, URL: https://www.frontiersin.org/articles/10.3389/fnbot.2021.619504.
[46] R.W. Hamming, Error detecting and error correcting codes, Bell Syst. Tech. J. 29 (2) (1950) 147–160, http://dx.doi.org/10.1002/J.1538-7305.1950.TB00463.X, URL: https://ieeexplore.ieee.org/document/6772729.
[47] M. Tomlinson, C.J. Tjhai, M.A. Ambroze, M. Ahmed, M. Jibril, Error-Correction Coding and Decoding, Springer International Publishing, Cham, 2017, http://dx.doi.org/10.1007/978-3-319-51103-0, URL: http://link.springer.com/10.1007/978-3-319-51103-0.
[48] A. Rasteh, F. Delpech, C. Aguilar-Melchor, R. Zimmer, S.B. Shouraki, T. Masquelier, Encrypted internet traffic classification using a supervised spiking neural network, Neurocomputing 503 (2022) 272–282, http://dx.doi.org/10.1016/j.neucom.2022.06.055.
[49] G. Aceto, D. Ciuonzo, A. Montieri, A. Pescapé, DISTILLER: Encrypted traffic classification via multimodal multitask deep learning, J. Netw. Comput. Appl. 183–184 (2021) 102985, http://dx.doi.org/10.1016/j.jnca.2021.102985, URL: https://linkinghub.elsevier.com/retrieve/pii/S1084804521000126.
[50] E.M. Izhikevich, Polychronization: Computation with spikes, Neural Comput. 18 (2) (2006) 245–282, http://dx.doi.org/10.1162/089976606775093882, URL: https://direct.mit.edu/neco/article/18/2/245-282/7033.
[51] S. Davies, A. Gait, A. Rowley, A. Di Nuovo, SpiNNaker spatial learning code and dataset, 2024, http://dx.doi.org/10.23634/MMU.00634935, URL: https://e-space.mmu.ac.uk/634935/.

Sergio Davies is a senior lecturer in the Department of Computing and Mathematics at Manchester Metropolitan University. He received his Laurea (equivalent to MSc Eng) in Telecommunication Engineering from the University "Federico II" in Napoli, Italy in 2006. Following his Ph.D. in Computer Science at the University of Manchester in 2012, Sergio continued his research on spiking neural networks as a postdoc in the Human Brain Project until 2016. Transitioning to industry, he took leadership roles in strategic advising as a consultant, and then managed various research projects covering computer hardware design, embedded systems architecture, software development and system networking. In 2019, Sergio returned to academia, initially at Sheffield Hallam University before joining Manchester Metropolitan University. His current research focuses on practical application of spiking neural networks in real-world scenarios.

Andrew Gait is a Research Software Engineer at the University of Manchester within the Research IT department. Following his PhD completed at the University of Leeds in 2007, he has worked in multiple different departments and institutes across the University of Manchester on various software projects and within multiple cross-disciplinary teams. This has included work on a multi-physics software library, the design of software for use in segmenting medical images, and the development and user support of spiking neural network software as part of the Human Brain Project in the APT group in Computer Science.

Andrew Rowley is a Senior Research Software Engineer at the University of Manchester within the Research IT department. After completing his PhD in Artificial Intelligence at the University of St. Andrews in 2004, he joined the University of Manchester as a Research Software Engineer. There he worked on various projects related to Access Grid video conferencing before becoming a Senior Research Software Engineer for NaCTeM working on text mining projects. He then joined the Human Brain Project team in Manchester and was working on the SpiNNaker project designing, building and supporting the sPyNNaker software since 2014.

Alessandro Di Nuovo is Professor of Machine Intelligence at Sheffield Hallam University. He received the Laurea (MSc Eng) and the PhD in Informatics Engineering from the University of Catania, Italy, in 2005 and 2009, respectively. He is the leader of the Smart Interactive Technologies research laboratory of the Department of Computing. He has published over 120 articles in computational intelligence and its application to cognitive modelling, human–robot interaction, computer-aided assessment of intellectual disabilities, and embedded computer systems. Prof. Di Nuovo has an extensive track record of leading interdisciplinary research and innovation in fundamental and applied topics in AI and Robotics, for which he has received several grants from prestigious funders (EPSRC, European Union) and companies. Currently, Prof. Di Nuovo is editor-in-chief (topics AI in Robotics; Human Robot/Machine Interaction) of the International Journal of Advanced Robotic Systems (SAGE). He is serving as Associate Editor for IEEE Journal of Translational Engineering in Health and Medicine.