


CASE STUDY

Discriminative training of spiking neural networks organised in columns for stream-based biometric authentication

 Enrique Argones Rúa¹  | Tim Van hamme² | Davy Preuveneers² | Wouter Joosen²
¹Department of Electrical Engineering, imec—COSIC, KU Leuven, Heverlee, Belgium

²Department of Computer Science, imec—DistriNet, KU Leuven, Heverlee, Belgium

Correspondence

 Enrique Argones Rúa, Department of Electrical Engineering, imec—COSIC, KU Leuven, Kasteelpark Arenberg 10, bus 2452, B-3001, Heverlee, Belgium.
Email: enrique.argonesrua@esat.kuleuven.be

Funding information

Flemish Research Programme Cybersecurity, Grant/Award Number: VR20192203; imec-Security & Privacy Centre, Grant/Award Number: Biometrics & Authentication; Research Fund KU Leuven

Abstract

Stream-based biometric authentication using a novel approach based on spiking neural networks (SNNs) is addressed. SNNs have proven advantages regarding energy consumption and they are a perfect match with some proposed neuromorphic hardware chips, which can lead to a broader adoption of user device applications of artificial intelligence technologies. One of the challenges when using SNNs is the discriminative training of the network since it is not straightforward to apply the well-known error backpropagation (EBP), massively used in traditional artificial neural networks (ANNs). A network structure based on neuron columns is proposed, resembling cortical columns in the human cortex, and a new derivation of error backpropagation for the spiking neural networks that integrate the lateral inhibition in these structures. The potential of the proposed approach is tested in the task of inertial gait authentication, where gait is quantified as signals from Inertial Measurement Units (IMU), and the authors' approach to state-of-the-art ANNs is compared. In the experiments, SNNs provide competitive results, obtaining a difference of around 1% in half total error rate when compared to state-of-the-art ANNs in the context of IMU-based gait authentication.

1 | INTRODUCTION

Artificial Neural Networks (ANNs) have become the most prevalent pattern recognition tool, being used in a multiplicity of applications. Regarding biometrics, it is already used in most biometric modalities, such as speaker authentication [1], face recognition [2], fingerprint recognition [3], hand-based biometrics [4, 5], electrocardiogram-based recognition [6], hand-written online signature recognition [7], or inertial gait recognition [8]. However, deep learning neural networks often come at the cost of larger complexity and computational requirements in terms of memory and processing power, which may thwart its deployment on constrained user devices, especially for continuous authentication. Additionally, in the case of biometrics, there are security and privacy concerns regarding storing biometric models and processing biometric data on cloud facilities.

In recent years, there has been an increasing interest in a different type of neural network, the Spiking Neural Networks (SNN) [9]. This interest has been driven mainly by the

possibility to use these networks within ultra-low power consumption-specific hardware modules called neuromorphic hardware [10–12], ideally suited for instance for continuous authentication. Results shown in [13] show gains in power consumption that go from 20 to 100 times more efficient. However, the drawback of this technology is the lack of mature learning approaches as opposed to standard ANNs. The most widely used method for SNNs is Synaptic Time-Dependent Plasticity (STDP), a generative approach that is a biologically inspired unsupervised method based on Hebbian rules [14], reaching limited discrimination performance when compared to discriminative approaches based on gradient descent for ANNs. Therefore, some approximations to gradient descent have been recently proposed as we will discuss later.

SNNs provide a natural integration of temporal information, since the spiking neurons have an internal state that evolves with time, spiking when it reaches a certain value. This makes networks using this type of neurons well suited for processing time series, a category of data to which many biometric modalities belong. Moreover, the promise of energy

This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

© 2022 The Authors. *IET Biometrics* published by John Wiley & Sons Ltd on behalf of The Institution of Engineering and Technology.

efficient evaluation of biometric data would be a key enabler for continuous authentication systems.

In this paper, we study the problem of gait authentication from inertial signals in light of SNNs. Specifically, we propose the use of SNNs topologically organised in columns, resembling cortical neuron columns in the mammal's cortex [15], and a novel error backpropagation method that allows to perform competitive supervised learning. Neurons in a column share the synaptic field and perform lateral inhibition, so each neuron in a column will specialise in specific input patterns. These columns may be understood as improved convolutional kernels, since they incorporate a mathematically sound mechanism to encourage intracolumn competition. Therefore, the proposed approach combines the structural benefits of neuron columns, which helps neuron specialisation, with the performance of discriminative approaches. We demonstrate the potential of this approach in the challenging biometric problem of inertial gait recognition. Inertial measurement unit (IMU)-based gait recognition uses the inputs captured by IMU sensors placed somewhere on the subjects' body. Thus, gait is modelled as a six dimensional time series: 3D linear acceleration and 3D angular velocity. We augment this signal and represent it as a 26D time sequence as described in prior work [16]. To summarise, the key contributions of our research are the following: use of column-based SNNs for gait authentication based on inertial signals and to provide this architecture with a new error backpropagation algorithm.

We must notice that this paper does not intend to perform a complete feasibility analysis of the proposed techniques for its application to biometric recognition. Instead, in this work, we show the potential of the proposed technique and technologies not only for the concrete inertial gait recognition process, but in general for processing streams, where information in the temporal dimension is paramount. To the best of our knowledge, this constitutes the first work on stream-based biometrics using the SNN technology. We compare the obtained results in terms of authentication performance to state-of-the-art ANNs to get a clear idea of the potential of this technology. This paper is an extended version of our previous work [17], where we additionally propose: (i) an alternative neuron model with additive thresholding instead of multiplicative thresholding, which avoids numerical problems in derivatives, especially when using several layers, and enables successful multilayer architectures; (ii) two different and complementary homeostatic mechanisms, which successfully avoid neuronal death within the neuronal columns; (iii) present more thorough experiments, including multilayer SNNs, and non-column-based SNNs; (iv) and perform a much deeper analysis of the experimental results, which also allows us to reach also more thorough conclusions.

The rest of this paper is structured as follows: in Section 2, we describe the works related to this paper with regard to both gait recognition and SNNs. In Section 3, we present the SNNs used in this paper, whereas the employed ANNs are described in Section 4. The experimental setup and data pre-processing are shown in Section 5. The experimental results are shown and discussed in Section 5. The paper draws conclusions in

Section 7, and finally, future research directions derived from this work on SNNs are shown in Section 8.

2 | RELATED WORK

In this section, we describe the state of the art in both IMU-based gait recognition and spiking neural networks.

2.1 | Previous work on gait recognition

For well over a decade, researchers have studied human locomotion as a means for authentication. The first works [18–21] all took a similar approach: considerable effort was spent to extract and align gait cycles and steps. Once cycles were extracted they were length normalised and analysed in both the time and frequency domain, after which comparisons were made using techniques, such as dynamic time warping and k-NN. However, results between these works were hard to compare due to a lack of a reliable benchmark. In 2014, Ngo et al. [22] published a large-scale benchmarking dataset and re-evaluated the aforementioned works, which shows that the top performing method achieved an Equal Error Rate (EER) of $\approx 14.3\%$.

Further improvements in recognition performance were achieved by relying on Gaussian Mixture Models (GMM) [23–26] and Hidden Markov Models [27, 28]. These methods are less dependent on robust gait cycle extraction and apply a sliding window over the gait sequence. Verification is done by computing the log-likelihood ratio of the query sample w.r.t. a personalised model and a general – universal background – model (UBM). The elegance of these methods is in the limited amount of enrolment samples required to train the personalised models as they can be adapted from the UBM. This benefit was not exploited by the first works that exploited HMMs, that is, the work by Nickel et al. [27]. The state-of-the-art performance on the OU-ISIR dataset obtained with GMMs [23] is $\approx 5.6\%$, while with HMMs [28], an EER of $\approx 1.4\%$ is achieved.

The most recent IMU-based gait authentication solutions rely on deep learning. One of the first approaches that leverage deep learning by Nguyen et al. [29] uses very shallow (1-layer) Convolutional Neural Networks (CNNs). On the OU-ISIR dataset, Nguyen et al. achieve an EER of $\approx 11.6\%$ for a closed set scenario, that is, all users used for evaluation are used during the training phase. Another CNN-based approach is IDNet [30], which extracts deep features that are fed to a one-class SVM, thereby adhering to an open set assumption. IDNet, was not evaluated with OU-ISIR, instead they collected their own data with only 50 users. But IDNet was later outperformed by the approach from Zou et al. [31] who did evaluate their system with OU-ISIR. Zou et al. combine CNNs and Long Short-Term Memory (LSTM) Networks, thereby achieving a Half Total Error Rate (HTER) of $\approx 0.4\%$ ¹.

¹Zou et al. reported an accuracy of $\approx 99.6\%$, which, due to their experimental setup, is equivalent to a Half Total Error Rate (HTER) of $\approx 0.4\%$

However, this number is optimistic as they perform their evaluation with pairs of sequences that can overlap or can belong to the same gait cycle. Another system by Fernandez et al. [32] that is also based on RNNs does not introduce such dependencies. Fernandez et al. report an EER of $\approx 6.7\%$ on the OU-ISIR dataset for an open set scenario. The deep learning-based approaches mentioned above again use heuristics to extract or align gait cycles. Such heuristics are often tuned to perform optimally for the dataset. Delgado-Escãno et al. [33] avoid such biases by not extracting or aligning gait cycles. Furthermore, they adopt a multi-task learning approach and achieve an EER of $\approx 1.1\%$ on the OU-ISIR dataset for a closed set scenario. A last recent work by Van hamme et al. [34] leverages RNNs with attention mechanisms. Moreover, they generate four different deep learnt systems based on CNNs and RNNs, which are trained with state-of-the-art loss functions from face recognition, that is, triplet loss [35] and arcface [36] loss. Their evaluation comprises also of HMMs and GMMs. They report EERs that range between $\approx 0.97\%$ and $\approx 2.13\%$ on the OU-ISIR dataset with an open-set assumption.

We refer to the recent survey of dos Santos et al. [37] on the application of deep learning for gait recognition. The authors review several deep learning methods, including the well-known CNNs and RNNs, but also Deep Belief Networks (DBNs), Capsule Networks (CNs), AutoEncoders (AEs), and Generative Adversarial Networks (GANs). They review how these methods were applied for video-based and accelerometer-based gait recognition and biometric identification on top of different datasets. While broad in the techniques surveyed, SNNs are out of scope in this survey study.

2.2 | Previous work on spiking neural networks

Although SNNs are a relatively new neural network paradigm, a lot of research efforts have focussed on this area lately. Regarding the learning technique, initially most of the works focussed on diverse versions of Synaptic Time Dependent Plasticity (STDP), a non-supervised technique that produces generative models, which produce spikes related to the most relevant or frequent inputs observed by the network. There are plenty of examples, such as [38–44]. However, all of them suffer from the same intrinsic limitation: its generative nature makes them not as accurate when dealing with classification tasks as other state-of-the-art discriminative approaches. The main problem is the non-differentiability of the spikes, the output of the networks, which makes traditional error backpropagation (EBP) not directly applicable.

There exist two main approaches to tackle this problem when using SNNs. The first one consists of converting conventional neural networks trained with traditional supervised EBP-based techniques as a way to circumvent the difficulties to backpropagate the error due to the non-differentiability of spikes. Some examples of this approach can be found in [45–55]. The second approach involves developing a framework where it is possible to perform EBP

on the SNN. A comparison of the mentioned approaches can be found in [38]. Although both approaches can provide reasonable performances, specific training can be theoretically more accurate, especially if SNNs have special topological features, which do not easily map from ANNs, as the columns used in this paper. This motivated us to use a specific training approach. Developments shown in [56–63] are some representative examples of EBP on SNNs. Since our approach belongs to this category, we will briefly describe these works and compare with our proposed algorithm to clearly state what is novel with respect to the state of the art in SNNs.

In [56], membrane potential plays the role of a differentiable output, and spikes are considered as noise. EBP is performed at all times (independently of spiking times). In [57], an EBP framework is derived for SNNs using temporal encoding, where the information is carried on the time of the first spike that each neuron produces (if any). In [58], conversion from traditional networks and Spatio-Temporal BackPropagation (STBP) are compared, and specific training on SNNs shows improvements in terms of efficiency. In [59], Normalised Approximate Descent learning rule is used to derive a backpropagation rule that efficiently minimises the differences between membrane potentials of target and training examples. In [60], SLAYER is presented as an error backpropagation method that allows to learn spiking output sequences from target sequences. SuperSpike is presented in [62] as a surrogate derivative rule that allows to perform EBP on SNNs. In [61], Spatio-Temporal Backpropagation is derived for SNNs, combining layer-by-layer spatial domain and timing-dependent temporal domain.

All the backpropagation frameworks made for SNNs so far share that the objective function is either specialised for classification tasks (trying to get a specific label as the average spiking rate at the output layer) or it tries to provide spike trains very similar in different inputs, implying a homogeneous dimensionality and duration (if applicable) of input signals. In both cases, it is hard to use these ideas to get a universal (open set, where the same network will be used to process samples from individuals not included in the training set) network for sequence-based biometric authentication, where the length of the sequence is variable (non-homogeneous, making the use of sequence approximation cumbersome). Both SLAYER and SuperSpike are exceptions since they provide enough flexibility, allowing for a similar objective function. However, none of these approaches are considering the specific structure of neuron columns, which play a crucial role in obtaining competitive learning. We can consider this feature of our approach as not only a novelty in discriminative learning-based SNNs, but also in ANNs in general.

3 | GAIT RECOGNITION SYSTEM USING SNNs

In this paper, we use Leaky Integrate and Fire (LIF) neurons [64] as fundamental computation units. Our networks will be

constituted of columns of these neurons, organised in layers. Each layer feeds with the outputs of the columns from previous layers, after an optional max pool layer.

We train the SNNs in two phases. In the first one, we use the unsupervised method Synaptic Time-Dependent Plasticity to get a good baseline for the second phase, which is a novel supervised Gradient Descent. In the following sections, we describe in detail the topological elements of the networks and the learning algorithms.

3.1 | Columns of LIF neurons

As mentioned above, neurons are organised in columns as shown in Figure 1. Neurons in a column share the synaptic field (input space), and only one can emit a spike at a given instant in time. This behaviour is modelled using a Winner Takes All (WTA) circuit, which chooses the neuron with the maximum normalised membrane potential, above the spiking threshold, as the spiking one. When a neuron spikes, a lateral inhibition signal $i^c[n]$ is fed back to the neurons in the column, which will put them in a refractory state, as explained below.

3.2 | LIF neuron

Two different discrete time LIF neurons are explored in this paper. The first one is shown in Figure 2, and it has been previously presented in our previous work [17]. These LIF neurons use multiplicative thresholds. In contrast, the LIF neurons presented in this work, shown in Figure 3, use additive thresholds.

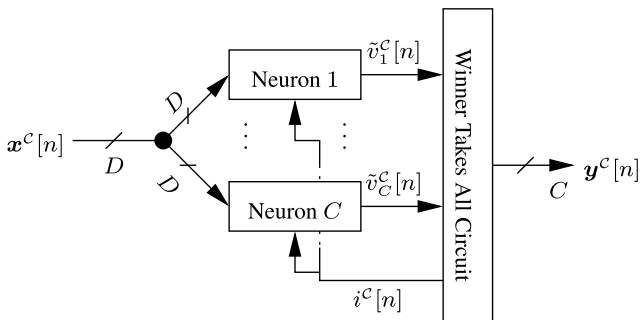


FIGURE 1 Column of neurons with lateral inhibition

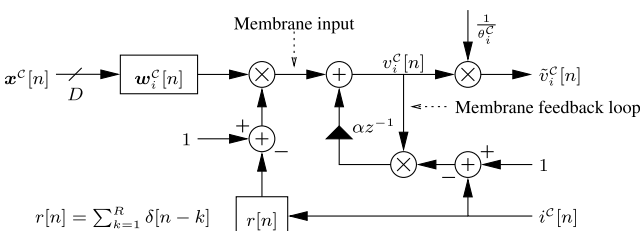


FIGURE 2 Diagram of neuron i in column C with the multiplicative threshold

LIF neurons can be in two different states: *active* and *refractory*. During the active state, the membrane potential leaky integrates the inputs filtered by their corresponding synaptic response filters as shown in the membrane feedback loop. Before the membrane potential is fed into the WTA circuit, the threshold θ either divides the membrane potential when multiplicative or it is subtracted when the threshold is additive. In the case of multiplicative thresholding neurons, the neuron can spike when the normalised potential exceeds unity, while in the case of additive thresholding neurons, the neuron can spike when the normalised potential is positive. The WTA circuit selects the most active neuron among the ones that can spike as follows for multiplicative and additive thresholding, respectively:

$$y_i^c[n] = \begin{cases} 1 & \Leftrightarrow i = \operatorname{argmax}_j \{ \hat{v}_j^c[n] | \hat{v}_j^c[n] \geq 1 \} \\ 0 & \Leftrightarrow \nexists j | \hat{v}_j^c[n] \geq 1 \\ & i^c[n] = \max_i \{ y_i^c[n] \} \end{cases}, \quad (1)$$

$$y_i^c[n] = \begin{cases} 1 & \Leftrightarrow i = \operatorname{argmax}_j \{ \hat{v}_j^c[n] | \hat{v}_j^c[n] > 0 \} \\ 0 & \Leftrightarrow \nexists j | \hat{v}_j^c[n] > 0 \\ & i^c[n] = \max_i \{ y_i^c[n] \} \end{cases}, \quad (2)$$

When any neuron in the column emits a spike, each neuron in the column receives an inhibition impulse from the column WTA circuit (i.e., $i^c[n] = 1$) and goes into the refractory state, where the membrane potential is reset and synaptic inputs are ignored by using $1 - i^c[n]$ and $1 - i^c[n] * r[n]$ as gate signals in the membrane potential feedback loop and input, respectively, with $r[n] = \sum_{k=1}^R \delta[n - k]$, and R the refractory period duration. It must be noticed that if $R = 0$, the inhibition only affects the feedback loop. Both R and the membrane persistence α (or equivalently) are the same in all the neurons in our model, since we understand that these are parameters related to the neurons' physiology in this biologically inspired system.

3.3 | Input layers

In many examples of SNNs, the input layer needs a transformation from continuous signals to spike trains. However, in

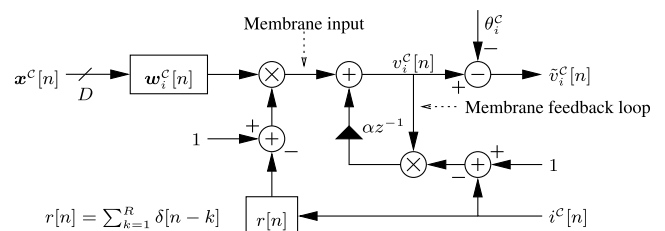


FIGURE 3 Diagram of neuron i in column C with the additive threshold

our case, this is not formally required. Our proposed LIF model can deal with continuous inputs, while providing a spike train output. However, we focussed on two different input transformations, which are convenient due to the positive filter weight initialisation required by STDP, as it will be shown later.

3.3.1 | MinMax

First, each input signal $x[n]$ is normalised using the minimax transformation:

$$x_{\text{minimax}}[n] = \frac{x[n] - m_x}{M_x - m_x}, \quad (3)$$

where $m_x = \min_{\text{Train Set}}\{x[n]\}$ and $M_x = \max_{\text{Train Set}}\{x[n]\}$. Then, two signals are derived: $x^+[n] = x_{\text{minimax}}[n]$ and $x^-[n] = -x_{\text{minimax}}[n]$. These two signals constitute the transformed input.

3.3.2 | Gaussian

This transformation computes 4 different signals from each input. First, we obtain $x_{\text{minimax}}[n]$ following Equation (3). Then, four different signals are derived as inputs to the input layer from each signal:

$$x_{\text{Gaussian}}^i[n] = e^{-\frac{(x_{\text{minimax}}[n] - \mu^i)^2}{2\sigma^2}}, \quad (4)$$

where $\mu^i = i/3$ and $\sigma = \frac{1}{6}\sqrt{\frac{1}{2\ln(2)}}$, and $i \in \{0, 1, 2, 3\}$. Each derived signal will provide information on the amplitude of the original input. All four signals will always be fed jointly, so neurons will always have a complete information on the relative amplitude of their inputs.

3.4 | Unsupervised initialisation using STDP

Synaptic weights are randomly initialised from Gaussian distributions and clamped into the interval $[0, 1]$. Then, STDP is used to adapt these weights to the statistics of the gait input sequences. Different update rules for the weights have been proposed for STDP, but the main idea is to implement (i) Long-Term Potentiation (LTP) to weights recently contributing to the potential of a neuron that spikes, that is, increasing this weight; and (ii) Long-Term Depression (LTD) to weights that did not contribute to the potential of a neuron that spikes. In other words, neurons that spike reinforce the weights that contribute to the spiking and weaken those weights that do not contribute. In our case, we implement LTP and LTD by increasing the weights proportionally to the observed input contribution. Given the synaptic filter $\mathbf{w}[n] = \sum_{i=0}^L w^k \delta[n-k]$, if a spike is emitted at time s , then $\Delta w^k = p \sum_{j=0}^k x[s-(k+j)] \alpha^j$ if $\sum_{j=0}^k x[s-(k+j)] \alpha^j > \epsilon$, and $\Delta w = -d$ otherwise. A

homeostatic rule is also used to avoid too dominant neurons in a column, decreasing the activation thresholds of neurons that spike below the columns' average spiking rate and increasing them for neurons with a spiking rate higher than the columns' average. These averages are computed using first-order average estimators $\bar{r}[n] = \tau^{-1}r[n] + (1 - \tau^{-1})\bar{r}[n-1]$ with τ equal to 5 times the average gait sequence length. Thresholds are updated at the end of each sequence STDP by $\Delta \theta_i^c = \beta(\bar{r}^c - C\bar{r}_i^c)$, where C is the number of neurons in column C , \bar{r}^c and \bar{r}_i^c are the average spiking rate of the column and its i th neuron, respectively. After the update, the thresholds are clamped to the interval $[1, 10]$.

3.5 | Supervised training using EBP

The main obstacle to perform EBP on SNNs is the non-differentiability of the neuron's output. Its discontinuous (it is either 0 or 1) and homogeneous (all the spikes look the same) nature require adopting a different approach to the one used in traditional ANNs. One of the most common approaches is to define a surrogate function f that is, differentiable, and substitute the output of the neuron by that surrogate during gradient computations, that is, $\delta \mathbf{y} \sim \delta f$. One sensible approach is to make this surrogate function model the probability of that neuron firing. A sensible surrogate function for the proposed neuron columns should be based on the normalised membrane potential signals $\tilde{v}_i^c[n]$, which are differentiable with respect to the neurons' input and synaptic filter weights. In this regard, it should also

1. Be monotonically increasing with its normalised membrane potential and monotonically decreasing with the normalised membrane potential of the other neurons in the column.
2. Saturate when the membrane potential of the neuron dominates the other neurons' membrane potential in the column.

Taking these into account, we propose to use the softmax function of the normalised membrane potential as a surrogate function:

$$f^c[n] = \text{softmax}(\tilde{\mathbf{v}}^c[n]) = \frac{1}{\sum_{j=1}^C e^{\tilde{v}_j^c[n]}} \begin{pmatrix} e^{\tilde{v}_1^c[n]} \\ \vdots \\ e^{\tilde{v}_C^c[n]} \end{pmatrix} \quad (5)$$

We evaluate this differentiable surrogate function during the backward phase of EBP *only at the spiking times*, that is, when one neuron in the column emits a spike. By doing this, we only take into account the spiking events, which are the ones forwarding real information, thus removing influence of instants where the neurons do not get enough evidence to spike and saving a lot of computation power and training time. Thus, spikes serve as noise removing and energy efficiency mechanisms.

3.6 | Objective function, optimiser, and boosting

Although there could be information on time dependencies among spiking events (local information), we only use the average spiking rate at the output layer neurons. We use a Siamese architecture, where a reference sequence \mathcal{S}_R , belonging to class C_R , and a probe sequence \mathcal{S}_P , belonging to class C_P and make a forward pass through the network, obtaining two average output average spiking rates \mathbf{r}_R^o and \mathbf{r}_P^o . We then maximise the following cosine similarity-based objective function:

$$O(\mathcal{S}_R, \mathcal{S}_P) = (2\delta[C_R - C_P] - 1) \frac{\mathbf{r}_R^o \cdot \mathbf{r}_P^o}{\|\mathbf{r}_R^o\| \|\mathbf{r}_P^o\|}, \quad (6)$$

where $\delta[\cdot]$ is the Kronecker delta function.

We use the Nadam optimiser as described in [65] to improve convergence, abiding the parameters suggested in this paper. We also incorporated boosting to train the neural network, grouping the pairs of sequences matching a given enrolment sequence in the same minibatch, as in tuple loss [66], and performing the updates based only on the hard negative sequences and the positive sequence (if any of the negative examples is hard, or if the positive sequence is hard itself). A hard sample is the one that gets a wrong decision or a right decision but closer to the decision threshold than a given margin. The decision threshold is updated each epoch to the one resulting in Equal Error Rate (EER) performance on the train set.

3.7 | Homoeostatic mechanisms on columns

Since columns compete in columns for spiking, some neurons may be too successful and make other neurons to never spike, what is usually called neurons death. To avoid this, two different homoeostatic mechanisms are implemented on the additive thresholding LIF neurons.

3.7.1 | Homoeostatic gradient

The first one consists of adding a term to each neurons' derivative with respect to the neurons' activity, related to the spiking rate for each neuron, in addition to the increment derived from error backpropagation. This regularisation slightly shifts the error derivative towards making all the neurons spike at the same rate, which will discourage neurons death. This is implemented as follows:

$$\Delta\theta_i = (\Delta\theta_i)_{\text{EBP}} + \gamma(\bar{r}^c - C\bar{r}_i^c) \quad (7)$$

where C is the number of neurons in column \mathcal{C} , \bar{r}^c and \bar{r}_i^c are the average spiking rates of the column and its i th neuron, respectively, and γ is small constant, which will decay geometrically when there are not dead neurons in the column.

3.7.2 | Homoeostatic threshold boosting

The second one is a reactive homoeostatic mechanism, which slightly decreases the thresholds of dead neurons and slightly increases the thresholds of too active neurons after each training batch is processed. A neuron is considered as dead when its spiking rate is less than the possible maximum spiking rate of any neuron in the same layer divided by 10 times the number of neurons in the column. A neuron is considered as too active when its spiking rate is more than the possible maximum spiking rate of any neuron in the same layer divided by two. This threshold boosting only occurs when there exist dead neurons in a column, and it modifies the thresholds of dead neurons and too active neurons as follows:

$$\Delta\theta_i^{\mathcal{C}, \text{dead}} = -\zeta \frac{\|W_i^c\|}{D} \quad (8)$$

$$\Delta\theta_k^{\mathcal{C}, \text{too active}} = \zeta \frac{\|W_k^c\|}{D} \quad (9)$$

where ζ is a small constant, $\|\cdot\|$ is the Frobenius norm, and D is the number of dead neurons in column \mathcal{C} . In addition to this, all neuron thresholds are decreased abiding Equation (8) when the column spiking rate is less than $\frac{0.95}{R+1}$, where R is the refractory period. This keeps the columns as active as possible. These two mechanisms can be combined or used separately.

3.8 | Non-column-based SNNs

For comparison purposes, we also include an SNN architecture, which does not use neuron columns. In this architecture, each layer is fed with the whole output of the previous layer (or MinMax normalised input). Each layer is constituted by a set of linear 1D filters with as many channels as inputs, each of them followed by a Max pooling layer and an FIR neuron. The last layer is followed by a Global Average Pooling. We use a sigmoid surrogate for EBP with a slope of 10 in all FIR neurons [62].

This architecture is trained by maximising the cosine similarity-based objective function in Equation (6), but adding necessary regularisation terms proportional to the norm 1 of the output at each layer, to avoid neuronal death. The weight of these regularisation terms is adapted through the training phase, increasing it when the norm goes below a given threshold, and decreasing it when the norm is stable. The inclusion of this approach will allow to highlight the actual contribution made by the neuron columns in terms of authentication performance.

4 | GAIT RECOGNITION SYSTEM USING ANNS

We build a gait recognition system based on ANNs, which serves as a baseline to adequately demonstrate the potential of SNNs to model gait. We choose an architecture that led to

competitive results for gait sequence modelling tasks [34, 67]. Specifically, we chose Temporal Convolutional Networks (TCN), which are proven to be an effective strategy to model time series data in general [68]. Furthermore, to accommodate the needs of authentication, that is, an open set assumption with limited amounts of enrolment data, we leverage metric learning to train the system, that is, we use the triplet loss [35].

We construct multiple TCN networks with varying depth and an approximately constant model complexity. We present the baseline architecture from our prior work in Figure 4. This TCN has a fairly simple architecture with only three layers. The layers are three 1D convolutional layers with dilation, where the number of kernels increases from 32 in the first layer, to 64 in the second, and to 128 for the final layer. The dilation rate is 2, 4, and 8; while the kernel sizes are 3, 3, and 5, respectively. Global average pooling is performed, which provides a final embedding of size 128. The embedding is length normalised and used to evaluate the triplet loss function:

$$\mathcal{L}(A, P, N) = \max(\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + m, 0) \quad (10)$$

where $f(A)$ is the embedding of the anchor, $f(p)$ the embedding of a gait sequence, which belongs to the same user as the anchor, and $f(N)$ the embedding of a gait sequence from a different user. m , the margin, is an additional term, which further reduces inter-cluster similarity and encourages higher intra-cluster similarity. We mine semi-hard triplets to learn on, that is, those where the negative sample is further from the anchor than the positive one, but still a positive loss is incurred. We train the network for 500 epochs with the Adam optimiser with a learning rate of 0.001.

During training, we feed fixed length sequences of size 180 to the network due to the sample rate of 100 Hz this corresponds to 1.8 s of data. We chose this value such that at least one gait cycle is captured. The starting point within the original gait sequence of this window of length 180 is chosen at random. A new choice is made every epoch. Note that the input sequences are the pre-processed augmented sequences

that contain 26 dimensions. This pre-processing is described in Section 5. During testing, the full length sequences are used to compute the embedding.

Following the same strategy, we also construct a one layer and two layer TCN with a similar amount of trainable parameters. This allows us to provide baselines for different network depths. These depths mainly influence the effective receptive field of the network, which depend on the amount of layers N , the dilation rate d , and the filter size k . The effective history of a layer can be computed as $(k - 1)d$. Thus, the effective receptive field of the whole network is then: $\sum_{n=1}^N (k_n - 1)d_n$, with k_n and d_n the filter size, and dilation rate of the n^{th} layer, respectively. Thus, we train three networks with the following parameters:

- **3-layer TCN:** the network described above with a kernel size 3, 3, 5; a dilation rate 2, 4, 8; and a number of kernels 32, 64, 128, which gives an effective receptive field of 40. This architecture is illustrated in Figure 4.
- **2-layer TCN:** with a kernel size 3, 5; a dilation rate 2, 4; a number of kernels 64, 152, which gives an effective receptive field of 20.
- **1-layer TCN:** with a kernel size 10, a number of kernels 206, and no dilation, which gives a receptive field of 10.

5 | EXPERIMENTAL SETUP AND DATA PRE-PROCESSING

To train and evaluate our systems, we use the IMU sequences contained in the OU-ISIR dataset labelled as *level walk* and captured by the centre sensor. Usually for each user, there are two sequences, of which we use one for enrolment and one for testing. We only consider the 483 users with two valid walking sequences for all three sensors, that is, left, right, and centre located sensors. Only the centre sensor is used for the experiments.

The walk sequences of the OU-ISIR dataset are represented as a six-dimensional (6D) time series: 3D linear acceleration and 3D angular velocity. We augment this signal and represent it as a 26D time sequence as described in prior work [16]:

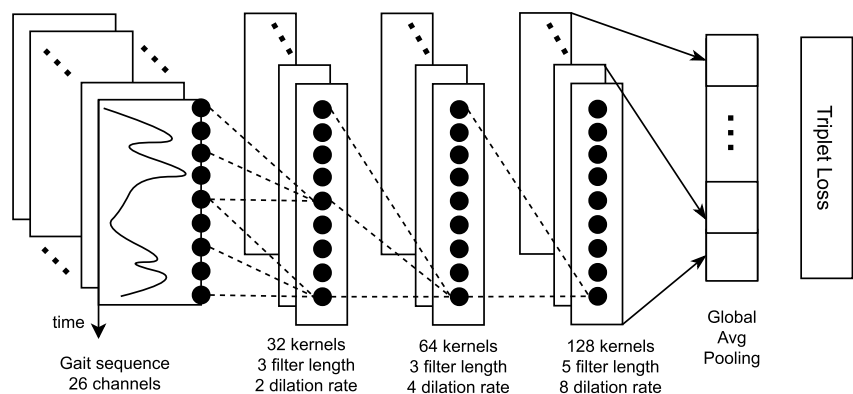


FIGURE 4 The architecture of the 3-layer artificial neural networks (ANN) system, which is based on one-dimensional convolutions with dilation and trained with the triplet loss

- Eight dimensions are derived from the *gyroscopic dynamics* (8D): the angular velocity $\vec{\omega}(n) = (\omega_x(n), \omega_y(n), \omega_z(n))$, their first-order differences $\alpha_d(n) = \omega_d(n) - \omega_d(n-1)$, and the magnitude of both vectors $\omega(n) = \|\vec{\omega}(n)\|$ and $\alpha(n) = \|\vec{\alpha}(n)\|$.
- Six dimensions are related to the *vertical and horizontal components* (6D), which are an approximation of the vertical and horizontal acceleration in the world plane, computed as [23] $v(n) = \vec{a}(n) \cdot \vec{G}$ and $b(n) = \vec{a}(n) - \left(v(n) \frac{\vec{G}}{\|\vec{G}\|} \right)$ with \vec{G} the estimate of the gravity component computed as $\vec{G} = (\text{mean}(\vec{a}_x(n)), \text{mean}(\vec{a}_y(n)), \text{mean}(\vec{a}_z(n)))$. They are complemented with their jerks (first-order differences) and velocities (integration). The constant factor introduced by integration is filtered out by applying a butter high-pass filter of order 5 with a cutoff frequency of 1 Hz.
- Twelve dimensions are related to *roll and pitch* (12D). We make two approximations of roll and pitch: the first approximation only uses linear accelerations: $r_a(n) = \text{atan2}[\dot{a}_z^g(n), \dot{a}_y^g(n)]$ and $p_a(n) = \text{atan2}[\dot{a}_x^g(n), \dot{a}_y^g(n)]$. The second approximation fuses the first with an estimation of roll and pitch from the angular velocity: $f(n) = 0.98[f(n-1) + f_g(n)] + 0.02f_a(n)$, where f is either the pitch p or the roll r ; and $f_a f_g$ are the approximations of roll and pitch from the accelerometer and gyroscope data, respectively: $p_g(n) = \sum_{k=0}^n \omega_x(n)$ and $r_g(n) = \sum_{k=0}^n \omega_z(n)$. We keep both approximations and complement them with their first-order and second-order differences: $\dot{r}_a(n)$, $\dot{p}_a(n)$, $\ddot{r}_a(n)$, and $\ddot{p}_a(n)$; and the fused approximations of roll and pitch $\dot{r}(n)$, $\dot{p}(n)$, $\ddot{r}(n)$, and $\ddot{p}(n)$.

We adopt a 5-fold cross-validation protocol, where we randomly split the 483 users into 5 disjoint sets, which contain either 96 or 97 users: $S_s = \{U_0^s, U_1^s, \dots, U_x^s\}$ with $x \in [96, 97]$ and $s \in [0, 4]$. Thus, we do 5 rounds, where during each round k , we select S_k as the test set, and the remaining 4 sets as the training set. We report the Equal Error Rate (EER) in the test set, that is, the threshold at which False Rejection Rate (FRR) is equal to False Acceptance Rate (FAR). Besides, we plot the Detection Error Trade-off curves for a visual comparison. As each set contains x users for whom there are two walking sequences, x benign authentication attempts and $x(x-1)$ malicious authentication attempts can be simulated.

In addition to the experiments performed in our prior work [17], we tested different multilayer SNNs and also performed experiments with additive thresholding columns, not only focussing on the final biometric recognition figures, but also analysing the influence of the two presented homeostatic mechanisms.

We have developed our column-based SNN framework as a standalone C++ library, which uses *armadillo* [69] for fast linear algebra computations, such as convolutions.

For the non-column-based SNN, we used the library *SNN-Torch* [63]. During the experiments done in this paper, we

set the norm 1 threshold in the regularisation strategy to 0.05 as a well-performing value.

The SNN experiments were performed in two different machines:

- CPU server: $2 \times 2 \times 10$ -core Intel[®] Xeon[®] E5-2687 W v3 3.1 GHz, 25 M Cache, 9.60 GT/s QPI, Turbo, HT, 10 C/20 T (160 W) Max Mem 2133 MHz. Memory: 256 GB.
- Workstation: AuthenticAMD[®] AMD Ryzen[®] 9 3900XT 12-Core Processor x24. Memory 4×32 GB, DDR4 3000 MT/s.

The TCN architecture was implemented using tensorflow v2.3.0. The implementation of the triplet loss function of the tensorflow addons package [70] is used. The models are trained on a workstation with the following specifications: Intel[®] Core[®] i5-9500 CPU with 6 cores at 3.00 GHz, a GeForce[®] RTX[®] 2060 Rev. A graphics card with 6 GB ram and memory 32 GB.

6 | EXPERIMENTAL RESULTS

In our experiments, we tested different SNN topologies, with one, two, and three layers, some with additive and some with multiplicative thresholding, using max pooling between them. We present here the performance obtained by the TCN architectures presented in Section 4, together with the following SNNs:

- **N1**: 1 layer, 4 columns, 32 neurons/column, 1 coefficient filter, 32D MinMax column input, multiplicative thresholding, refractory period $R = 1$.
- **N2**: 1 layer, 4 columns, 32 neurons/column, 4 coefficient filter, 32-D MinMax column input, multiplicative thresholding, refractory period $R = 1$.
- **N3**: 2 layers, each with 4 columns, 32 neurons/column, and 4 coefficient filters, multiplicative thresholding. The first layer has 32-D and the second layer has 96-D MinMax column inputs. Max pooling with window and stride 2 is used between layers, and refractory period $R = 2$ in both layers.
- **N4**: 1 layer, 32 columns, 32 neurons/column, 2 coefficient filter, 32-D MinMax column input, multiplicative thresholding, refractory period $R = 1$.
- **N5**: 1 layer, 4 columns, 32 neurons/column, 2 coefficient filter, 32-D MinMax column input, additive thresholding, refractory period $R = 1$. This network has been included to analyse the role of homeostatic rules.
- **N6**: 1 layer, 16 columns, 16 neurons/column, 8 coefficient filter, 32-D Gaussian column input, 4-Gaussian input, refractory period $R = 3$, additive thresholding.
- **N7**: 3 layers, additive thresholding. *Input layer* with 16 neurons/column, 16 columns, 32-D Gaussian column input, 3 coefficient filter, refractory period $R = 1$; *second*

layer with 16 neurons/column, 16 columns, 3 coefficient filter, refractory period $R = 1$, 32-D input; *output layer* with 64 columns, 16 neurons/column, 7 filter coefficient, refractory period $R = 2$, 32-D input.

- **N8:** 3 layers, additive thresholding. *Input layer* with 16 neurons/column, 16 columns, 32-D Gaussian column input, 3 coefficient filter, refractory period $R = 1$; *second layer* with 16 neurons/column, 16 columns, 5 coefficient filter, refractory period $R = 2$, 32-D input; *output layer* with 64 columns, 16 neurons/column, 5 filter coefficient, refractory period $R = 2$, 32-D input.
- **N9:** 3 layers, additive thresholding. *Input layer* with 16 neurons/column, 16 columns, 32-D Gaussian column input, 3 coefficient filter, refractory period $R = 1$; *second layer* with 16 neurons/column, 16 columns, 3 coefficient filter, refractory period $R = 2$, 32-D input; *output layer* with 64 columns, 16 neurons/column, 5 filter coefficient, refractory period $R = 2$, 32-D input.
- **N10:** 1 layer, 64 columns, 16 neurons/column, 8 coefficient filter, 32-D Gaussian column input, 4-Gaussian input, refractory period $R = 3$, additive thresholding.
- **N11:** Non-column-based SNN with 3 layers using Super-Spike. *Input layer* with a filter kernel size 16 and a pool

kernel size and stride 3; *second layer* with a filter kernel size 8, a pool kernel size and stride 3; *output layer* with a filter kernel size 4, a pool kernel size and stride 3.

It has to be noted that the different SNN and ANN methods have their own optimal hyperparameters, making it difficult to perform performance comparisons using the same architectures. For instance, multiplicative thresholding SNNs do not perform well in multilayer experiments due to stability issues, which appear when the threshold gets close to 0. Different architectures have been explored for all the different systems, and the architectures used in this paper are good examples for each approach.

Table 1 shows the performance of the tested neural networks. The conventional TCN outperforms the SNN approach, although the best performing ones, that is, N6, and especially N4, get only slightly worse results. The performance of N6 is remarkable taking into account the small complexity of this network in comparison to N4 or the TCN. Also, shallow spiking neural networks perform better than the 2-layered N3. The performance of the best performing networks for each depth can be visually compared in Figure 5, where the Detection Error Trade-off curve is shown.

TABLE 1 Authentication performance in terms of test HTER% of the different networks

Label	Algorithm	HTER% for each test fold					$\mu_{\text{HTER}\%} \pm \sigma_{\text{HTER}\%}$
N1	STDP	7.97	4.88	6.19	10.66	6.25	7.19 ± 2.23
	EBP	3.98	2.51	3.09	6.25	5.21	4.21 ± 1.53
N2	STDP	9.28	5.96	8.25	10.71	11.46	9.13 ± 2.17
	EBP	4.16	2.13	4.12	7.27	4.17	4.37 ± 1.84
N3	STDP	10.53	7.22	11.16	15.62	11.46	11.20 ± 3.00
	EBP	7.62	2.80	4.99	6.33	8.33	6.01 ± 2.20
N4	STDP	5.62	4.12	5.15	9.38	6.25	6.10 ± 1.99
	EBP	2.49	1.03	2.06	3.12	2.08	2.16 ± 0.76
N6	STDP	7.08	5.94	6.04	9.81	6.78	7.13 ± 1.57
	EBP	4.33	2.10	2.45	6.24	2.42	3.51 ± 1.76
N7	STDP	13.31	15.86	12.50	18.21	17.98	15.57 ± 2.62
	EBP	3.14	2.97	3.89	9.63	7.07	5.34 ± 2.91
N8	STDP	17.21	18.43	17.82	20.61	20.66	18.55 ± 1.60
	EBP	4.16	3.26	6.30	8.85	5.68	5.65 ± 2.16
N9	STDP	18.29	13.65	14.31	21.04	18.01	17.06 ± 3.06
	EBP	4.86	2.58	3.21	8.73	4.10	4.70 ± 2.42
N10	STDP	6.78	5.42	5.92	9.65	8.27	7.21 ± 1.74
	EBP	3.44	2.93	2.12	7.52	2.77	2.76 ± 2.16
N11 (SuperSpike)	EBP	8.35	4.54	5.57	8.33	8.85	7.13 ± 1.94
1-layer TCN	EBP	2.06	1.03	2.06	3.00	1.12	1.88 ± 0.76
2-layer TCN	EBP	1.19	0.79	1.03	2.08	0.84	1.19 ± 0.47
3-layer TCN	EBP	1.03	1.92	1.92	2.08	1.06	1.60 ± 0.46

Regarding the comparison between multiplicative and additive thresholding methods for SNNs, our experiments do not show significant differences on HTER% between them. Also, both MinMax, represented by N1-4, and 4-Gaussian, represented by N5-10, both input transformations obtain comparable results.

In both SNN and TCN experiments, we did not find better results for the deepest networks. The gait database size is an important factor. A bigger dataset might lead to different results with respect to depth. In the case of SNNs, 2-layered and 3-layered networks obtain very good results in the training set (not shown here), but performance is worse in the test set when compared to 1-layered networks.

Regarding the non-column-based SNN system, its performance lies between STDP and EBP performance of the column-based SNN systems proposed in this work. This is motivated by a higher instability shown by the non-column-based SNN, which makes the regularisation term more important with respect to the discriminative gradient term during optimization than in the column-based case. Without regularisation, there would be nothing avoiding the spiking rate to arbitrarily decrease in our distance learning the objective function and eventually produce dead neurons, thus stopping effective learning. Although this is true for both column- and non-column-based approaches, this effect is more outspoken in the non-column-based approach.

There is another phenomenon that shows up in the results: fold heterogeneity. Fold 4 appears to be especially difficult for

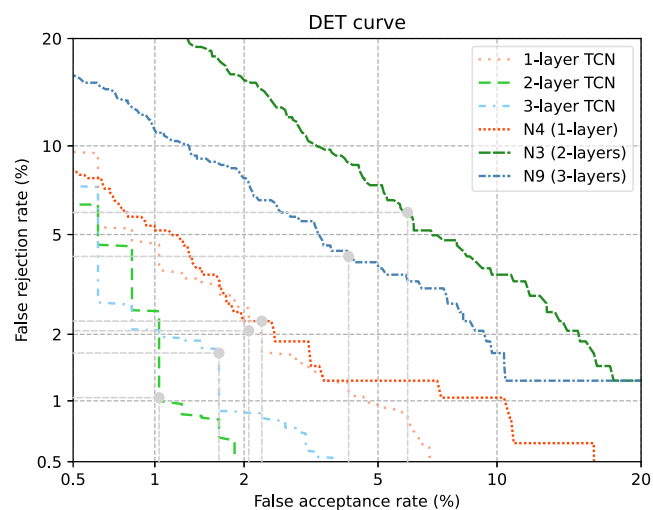


FIGURE 5 DET curve for the best performing EBP-trained SNNs and Temporal Convolutional Networks (TCN) for each network depth

TABLE 2 N5 performance in terms of a priori HTER% (using Equal Error Rate (EER) threshold) when using the different homoeostasis strategies

Homoeostatic rule	HTER% for each test fold					Average \pm
None	4.56	4.01	3.93	6.83	3.10	4.49 \pm 1.41
Homoeostatic Gradient	7.02	3.96	4.99	7.31	10.59	6.77 \pm 2.55
Homoeostatic Thresholding Boosting	5.18	3.43	4.83	6.69	2.53	4.53 \pm 1.61
Both	5.22	2.43	4.67	4.95	2.97	4.05 \pm 1.26

all the tested systems. Since fold division has been done randomly, without fairly distributing important factors, such as age and gender throughout the different folds, this may affect the performance of a fold. Since these side factors are not disclosed in the OU-ISIR database, we must accept these random nuisances.

To analyse the influence of the homoeostatic rules during training, and its performance on test set performance, we run experiments where these rules are switched on or off using N5 as a benchmark. This is the reason why the performance of this network is reported in Table 2. The training error evolution during training when no homoeostatic mechanisms are used is compared in Figure 6 with the training error when homoeostatic mechanisms are used for network N5 in fold 5 as a representative example. The best performing network during training is chosen, so the performance seems comparable attending to the results shown in Table 2. However, it can be seen that when no homoeostatic mechanisms are used, the performance in the training set eventually degrades. This is due to the increasing number of dead neurons in the SNN columns, which eventually thwarts their discrimination ability. Homoeostatic threshold boosting is a necessary mechanism in our experiments. When comparing its training set performance during training while also incorporating homoeostatic gradient, there are no significant differences as can be observed in detail

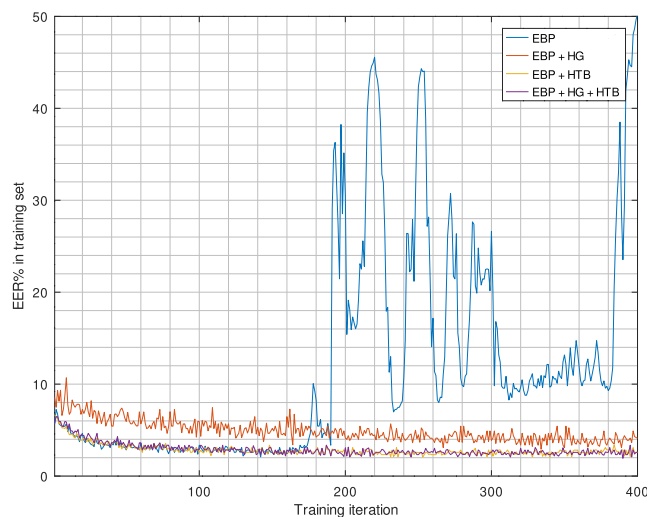


FIGURE 6 EER% in training set during training in fold 5 when no homoeostatic rules are applied error backpropagation (EBP), and when homoeostatic gradient (HG) or homoeostatic threshold boosting (HTB) is used

in Figure 7. However, using both mechanisms simultaneously results in a better performance for the test set, as shown in Table 1, due to the even lower number of dead neurons.

7 | CONCLUSIONS

In this work, we presented a novel column-based SNN architecture and derived a novel EBP approach for supervised learning for this architecture, two different LIF neuron models that can be alternatively employed, and two homeostatic rules that when applied during discriminative training in the proposed SNN architectures help to ensure that all neurons keep spiking at an optimal rate. We tested this approach on the challenging task of authenticating persons using IMU gait signals in an open set protocol and compared it to state-of-the-art ANNs and SNNs.

Although the results obtained by EBP on SNNs are yet slightly behind from the ones shown by state-of-the-art ANN architectures, a clear improvement over the generative approach STDP is demonstrated, resulting in closing the performance gap with respect to state-of-the-art ANNs. Also, the neuron columns show better performance when compared to non-column-based SNNs due to the higher instability of spiking rates exhibited by the non-column-based case.

The low power consumption shown by SNN hardware implementations, together with the reduced performance gap with respect to ANNs, encourages further research on the application of SNN for biometrics. This makes this technology especially suitable for continuous authentication solutions, where power consumption on the user device may be a limiting factor, but also for a wider range of applications related to data stream processing, which may include smart cameras, autonomous driving, drone autopilot and navigation, etc.

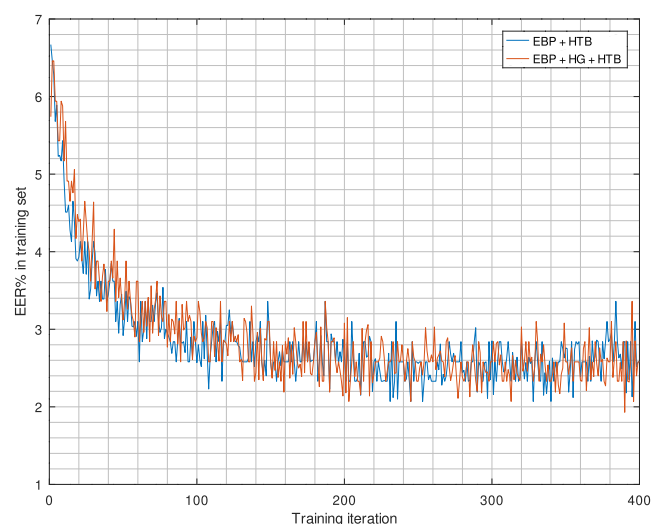


FIGURE 7 EER% in the training set during training in fold 5 when homeostatic threshold boosting (HTB) is used by itself or when the homeostatic gradient (HG) is also employed

This work constitutes a first approach to biometrics using the SNN technology. Of course, this work cannot be understood as a full exploration of the possibilities that this technology brings to the biometrics community, and even less to the signal processing and artificial intelligence communities. However, in this work, we have shown that this technology has great potential for data stream processing, using gait recognition as a challenging working example.

8 | FUTURE WORK

From the conclusions of this work, we can envision different research topics that are enabled by this work. These include testing the proposed SNN architecture in other biometric modalities and more generally in other artificial intelligence tasks. SNNs are stateful systems, and as such, very well suited to process temporal information. At a first glance, the techniques proposed in this paper can be immediately applied to multidimensional signals (understood as temporal streams), and given the generality of the EBP learning technique, the biometric's *metric learning* objective function is only one of the many possibilities, but the proposed framework and the EBP technique are not limited to this specific use case.

Furthermore, the synaptic input structure can be further generalised in future works to be able to process 2D multi-channel streams (one case can be video signals). Other research lines could also be devoted to explore new neuron models, where, for instance, alternative membrane potential update strategies are chosen. Last, but not least, new column-based architectures should be explored, and incorporating back-propagation through time (BPTT) and recurrent architectures should also be a goal in future works.

NOMENCLATURE

In the following figures and equations, we will use the following symbols whose meaning is briefly explained in this section:

D	Input dimensionality.
C	Number of neurons in a neurons column.
$\mathbf{x}^C[n]$	input of neurons column C at time n .
$\hat{v}_i^C[n]$	normalised membrane potential of neuron i in neurons column C at time n .
$\mathbf{y}^C[n]$	output (spike signals) of neuron column C at time n .
α	neurons memory factor.
R	refractory period.

ACKNOWLEDGEMENTS

This research is partially funded by the Research Fund KU Leuven, by the Flemish Research Programme Cybersecurity with reference number VR20192203, and by imec through the Security and Privacy Centre projects on Biometrics and Authentication.

CONFLICT OF INTEREST

The authors declare that they have no conflict of interests.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from Department of Intelligent Media, The Institute of Scientific and Industrial Research, Osaka University. Restrictions apply to the availability of these data, which were used under license for this study. Data are available <http://www.am.sanken.osaka-u.ac.jp/BiometricDB/InertialGait.html> with the permission of Department of Intelligent Media, The Institute of Scientific and Industrial Research, Osaka University.

ORCID

Enrique Argones Rúa  <https://orcid.org/0000-0002-4241-0134>

REFERENCES

- Tirumala, S.S., Shahamiri, S.R.: A review on deep learning approaches in speaker identification. In: Proceedings of the 8th International Conference on Signal Processing Systems. ICSPS 2016, pp. 142–147. Association for Computing Machinery, New York (2016). <https://doi.org/10.1145/3015166.3015210>
- Swapna, M., Sharma, Y.K., Prasad, B.M.G.: A survey on face recognition using convolutional neural network. In: Raju, K.S., et al. (eds.) Data Engineering and Communication Technology, pp. 649–661. Springer, Singapore (2020)
- Arrieta, A.G., Estrada, G.C., Romero, L.A., Lancho, Á.L.S.L.Y.B.P.: Neural networks applied to fingerprint recognition. In: Omatu, S., et al. (eds.) Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living, pp. 621–625. Springer Berlin Heidelberg, Berlin (2009)
- Chantaf, S., Hilal, A., Elsaleh, R.: Palm vein biometric authentication using convolutional neural networks. In: Bouhleh, M.S., Rovetta, S. (eds.) Proceedings of the 8th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT'18), vol. 1, pp. 352–363. Springer International Publishing, Cham (2020)
- Faúndez Zanuy, M., et al.: Authentication of individuals using hand geometry biometrics: a neural network approach. *Neural Process. Lett.* 26(3), 201–216 (2007). <https://doi.org/10.1007/s11063-007-9052-y>
- Salloum, R., Kuo, C.J.: Ecg-based biometrics using recurrent neural networks. In: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 2062–2066 (2017)
- Tolosana, R., et al.: Deepsign: deep on-line signature verification. *IEEE Trans. Biometrics, Behav. Iden. Sci.* 3(2), 1–239 (2021). <https://doi.org/10.1109/tbiom.2021.3054533>
- Lopez, F., et al.: Recurrent neural network for inertial gait user recognition in smartphones. *Sensors* 19(18), 4054 (2019). <https://doi.org/10.3390/s19184054>. <https://www.mdpi.com/1424-8220/19/18/4054>
- Maass, W.: Networks of spiking neurons: the third generation of neural network models. *Neural Network.* 10(9), 1659–1671 (1997). [https://doi.org/10.1016/s0893-6080\(97\)00011-7](https://doi.org/10.1016/s0893-6080(97)00011-7). <https://www.sciencedirect.com/science/article/pii/S0893608097000117>
- Monroe, D.: Neuromorphic computing gets ready for the (really) big time. *Commun. ACM* 57(6), 13–15 (2014). <https://doi.org/10.1145/2601069>
- Balaji, A., et al.: Mapping spiking neural networks to neuromorphic hardware. *IEEE Trans. Very Large Scale Integr. Syst.* 28(1), 76–86 (2020). <https://doi.org/10.1109/tvlsi.2019.2951493>
- Maganda, N., et al.: Implementation of spiking neural network classifiers based on backpropagation-based learning algorithms. In: 2009 International Joint Conference on Neural Networks, pp. 2294–2301 (2009)
- Blouw, P., et al.: Benchmarking keyword spotting efficiency on neuromorphic hardware. In: Proceedings of the 7th Annual Neuro-Inspired Computational Elements Workshop. NICE 19. Association for Computing Machinery, New York (2019). <https://doi.org/10.1145/3320288.3320304>
- Markram, H., et al.: Regulation of synaptic efficacy by coincidence of postsynaptic aps and EPSPS. *Science* 275(5297), 213–215 (1997). <https://doi.org/10.1126/science.275.5297.213>
- Lodato, S., Arlotta, P.: Generating neuronal diversity in the mammalian cerebral cortex. *Annu. Rev. Cell Dev. Biol.* 31(1), 699–720 (2015). <https://doi.org/10.1146/annurev-cellbio-100814-125353>
- Van Hamme, T., et al.: Gait template protection using HMM-UBM. In: Brömme, A., et al., editors. 2018 International Conference of the Biometrics Special Interest Group, BIOSIG 2018, Darmstadt, September 26–28, 2018. vol. P-282 of LNI. GI/IEEE. 1–8 (2018). Available from: <https://doi.org/10.23919/BIOSIG.2018.8553315>
- ArgonesRúa, E., van Hamme, T., Preuveneers, D., Joosen, W.: Gait authentication based on spiking neural networks. In: Brömme, A., et al. (eds.) BIOSIG 2021 - Proceedings of the 20th International Conference of the Biometrics Special Interest Group, pp. 51–60. Gesellschaft für Informatik e.V., Bonn (2021)
- Rong, L., et al.: A wearable acceleration sensor system for gait recognition. In: 2007 2nd IEEE Conference on Industrial Electronics and Applications, pp. 2654–2659 (2007)
- Derawi, M.O., Bours, P., Holien, K.: Improved cycle detection for accelerometer based gait authentication. In: 2010 Sixth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, pp. 312–317 (2010)
- Gafurov, D., Snekenes, E., Bours, P.: Improved gait recognition performance using cycle matching. In: 24th IEEE International Conference on Advanced Information Networking and Applications Workshops, WAINA 2010, pp. 836–841 (2010)
- Trung, N.T., et al.: Phase registration in a gallery improving gait authentication. In: 2011 International Joint Conference on Biometrics (IJCB), pp. 1–7 (2011).
- Ngo, T.T., et al.: The largest inertial sensor-based gait database and performance evaluation of gait-based personal authentication. *Pattern Recogn.* 47(1), 228–237 (2014). <https://doi.org/10.1016/j.patcog.2013.06.028>
- Lu, H., et al.: Unobtrusive gait verification for mobile phones. In: Proceedings of the 2014 ACM International Symposium on Wearable Computers. ISWC '14, pp. 91–98. ACM, New York (2014)
- Zhong, Y., Deng, Y.: Sensor orientation invariant mobile gait biometrics. In: IEEE International Joint Conference on Biometrics, pp. 1–8 (2014)
- SanSegundo, R., et al.: Frequency features and GMM-UBM approach for gait-based person identification using smartphone inertial signals. *Pattern Recogn. Lett.* 73, 60–67 (2016). <https://doi.org/10.1016/j.patrec.2016.01.008>
- SanSegundo, R., et al.: I-vector analysis for gait-based person identification using smartphone inertial signals. *Pervasive Mob. Comput.* 38, 140–153 (2017). <https://doi.org/10.1016/j.pmcj.2016.09.007>
- Nickel, C., et al.: Using hidden Markov models for accelerometer-based biometric gait recognition. In: 2011 IEEE 7th International Colloquium on Signal Processing and its Applications, pp. 58–63 (2011)
- Van hamme, T., et al.: Gait template protection using HMM/UBM. In: 2018 International Conference of the Biometrics Special Interest Group (BIOSIG), pp. 1–8 (2018)
- Nguyen, K.T., et al.: Gait recognition with multi-region size convolutional neural network for authentication with wearable sensors. In: Dang, T.K., et al. (eds.) Future Data and Security Engineering, pp. 197–212. Springer International Publishing, Cham (2017)
- Gadaleta, M., Rossi, M.: Idnet: smartphone-based gait recognition with convolutional neural networks. *Pattern Recogn.* 74, 25–37 (2018). <https://doi.org/10.1016/j.patcog.2017.09.005>
- Zou, Q., et al.: Deep learning-based gait recognition using smartphones in the wild. *IEEE Trans. Inf. Forensics Secur.* 15, 3197–3212 (2020). <https://doi.org/10.1109/tifs.2020.2985628>
- Lopez, F., et al.: Recurrent neural network for inertial gait user recognition in smartphones. *Sensors* 19(18), 4054 (2019). <https://doi.org/10.3390/s19184054>
- Delgado-Escañó, R., et al.: An end-to-end multi-task and fusion CNN for inertial-based gait recognition. *IEEE Access* 7, 1897–1908 (2019). <https://doi.org/10.1109/access.2018.2886899>

34. hamme, V., et al.: On the security of biometrics and fuzzy commitment cryptosystems: a study on gait authentication. *IEEE Trans. Inf. Forensics Secur.* 16, 5211–5224 (2021). <https://doi.org/10.1109/tifs.2021.3124735>
35. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: a unified embedding for face recognition and clustering. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 815–823 (2015)
36. Deng, J., et al.: Additive angular margin loss for deep face recognition. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4690–4699 (2019)
37. Filipi Gonçalves dos Santos, C., et al.: Gait recognition based on deep learning: a survey. *ACM Comput. Surv.* 55(2), 1–34 (2022). <https://doi.org/10.1145/3490235>
38. Tavanaei, A., et al.: Deep learning in spiking neural networks. *Neural Network.* 111, 47–63 (2019). <https://doi.org/10.1016/j.neunet.2018.12.002>
39. Tavanaei, A., Maida, A.S.: Multi-layer unsupervised learning in a spiking convolutional neural network. In: *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 2023–2030 (2017)
40. Tavanaei, A., Maida, A.S.: Bio-inspired Spiking Convolutional Neural Network Using Layer-wise Sparse Coding and Stdp Learning (2017)
41. Tavanaei, A., Maida, A.: Bp-stdp: approximating backpropagation using spike timing dependent plasticity. *Neurocomputing* 330, 39–47 (2019). <https://doi.org/10.1016/j.neucom.2018.11.014>
42. Diehl, P., Cook, M.: Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Front. Comput. Neurosci.* 9, 99 (2015). <https://doi.org/10.3389/fncom.2015.00099>
43. Tavanaei, A., Kirby, Z., Maida, A.S.: Training spiking convnets by stdp and gradient descent. In: *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8 (2018)
44. Kheradpisheh, S.R., et al.: Stdp-based spiking deep convolutional neural networks for object recognition. *Neural Network.* 99, 56–67 (2018). <https://doi.org/10.1016/j.neunet.2017.12.005>
45. Diehl, P.U., et al.: Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In: *2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8 (2015)
46. Esser, S.K., et al.: Backpropagation for energy-efficient neuromorphic computing. In: Cortes, C., et al. (eds.) *Advances in Neural Information Processing Systems*, vol. 28. Curran Associates, Inc. (2015). <https://proceedings.neurips.cc/paper/2015/file/10a5ab2db37feedfcaeb192ead4ac0e-Paper.pdf>
47. Rueckauer, B., et al.: Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Front. Neurosci.* 11, 682 (2017). <https://doi.org/10.3389/fnins.2017.00682>
48. Neil, D., Pfeiffer, M., Liu, S.C.: Learning to be efficient: algorithms for training low-latency, low-compute deep spiking neural networks. In: *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pp. 293–298. Association for Computing Machinery, New York (2016). <https://doi.org/10.1145/2851613.2851724>
49. Hunsberger, E., Eliasmith, C.: Spiking Deep Networks with LIF Neurons (2015)
50. Neil, D., Liu, S.: Effective sensor fusion with event-based sensors and deep network architectures. In: *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2282–2285 (2016)
51. Hunsberger, E., Eliasmith, C.: Training Spiking Deep Networks for Neuromorphic Hardware. *CoRR* (2016). [abs/1611.05141](http://arxiv.org/abs/1611.05141), Available from: <http://arxiv.org/abs/1611.05141>
52. Cao, Y., Chen, Y., Khosla, D.: Spiking deep convolutional neural networks for energy-efficient object recognition. *Int. J. Comput. Vis.* 113(1), 54–56 (2015). <https://doi.org/10.1007/s11263-014-0788-3>
53. O'Connor, P., et al.: Real-time classification and sensor fusion with a spiking deep belief network. *Front. Neurosci.* 7, 178 (2013). <https://doi.org/10.3389/fnins.2013.00178>
54. Stamatias, E., et al.: Robustness of spiking deep belief networks to noise and reduced bit precision of neuro-inspired hardware platforms. *Front. Neurosci.* 9, 222 (2015). <https://doi.org/10.3389/fnins.2015.00222>
55. Martinelli, F., et al.: Spiking neural networks trained with backpropagation for low power neuromorphic implementation of voice activity detection. In: *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, May 4-8, 2020*. (IEEE, 8544–8548 (2020)). <https://doi.org/10.1109/ICASSP40776.2020.9053412>
56. Lee, J.H., Delbruck, T., Pfeiffer, M.: Training deep spiking neural networks using backpropagation. *Front. Neurosci.* 10, 508 (2016). <https://doi.org/10.3389/fnins.2016.00508>
57. Mostafa, H.: Supervised learning based on temporal coding in spiking neural networks. *IEEE Transact. Neural Networks Learn. Syst.* 29(7), 3227–3235 (2018)
58. PatiñoSaucedo, A., et al.: Event-driven implementation of deep spiking convolutional neural networks for supervised classification using the spinnaker neuromorphic platform. *Neural Network.* 121, 319–328 (2020). <https://doi.org/10.1016/j.neunet.2019.09.008>
59. Anwani, N., Rajendran, B.: Training multi-layer spiking neural networks using normad based spatio-temporal error backpropagation. *Neurocomputing* 380, 67–77 (2020). <https://doi.org/10.1016/j.neucom.2019.10.104>
60. Shrestha, S.B., Orchard, G.S.: Spike layer error reassignment in time. In: Bengio, S., et al. (eds.) *Advances in Neural Information Processing Systems*, vol. 31. Curran Associates, Inc. (2018). <https://proceedings.neurips.cc/paper/2018/file/82f2b308c3b01637c607ce05f52a2fed-Paper.pdf>
61. Wu, Y., et al.: Spatio-temporal backpropagation for training high-performance spiking neural networks. *Front. Neurosci.* 12, 331 (2018). <https://doi.org/10.3389/fnins.2018.00331>
62. Zenke, F., Ganguli, S.: SuperSpike: supervised learning in multilayer spiking neural networks. *Neural Comput.* 30(6), 1514–1541 (2018). Available from: https://doi.org/10.1162/neco_a_01086
63. Eshraghian, J.K., et al.: Training Spiking Neural Networks Using Lessons from Deep Learning. *arXiv preprint arXiv:210912894*, 2021
64. Abbott, L.F.: Llapicque's introduction of the integrate-and-fire model neuron (1907). *Brain Res. Bull.* 50(5), 303–304 (1999). [https://doi.org/10.1016/s0361-9230\(99\)00161-6](https://doi.org/10.1016/s0361-9230(99)00161-6)
65. Dozat, T.: Incorporating nesterov momentum into ADAM. In: Bengio, Y., LeCun, Y. (eds.) *4th International Conference on Learning Representations, ICLR 2016* (2016). San Juan, Puerto Rico, May 2–4, 2016, Conference Track Proceedings. (2016). <https://iclr.cc/archive/www/doku.php%3Fid=iclr2016:accepted-main.html>
66. Yu, B., Tao, D.: Deep metric learning with triplet margin loss. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6489–6498 (2019)
67. hamme, V., et al.: A systematic comparison of age and gender prediction on imu sensor-based gait traces. *Sensors* 19(13), 2945 (2019). Available from: <https://doi.org/10.3390/s19132945>. <https://www.mdpi.com/1424-8220/19/13/2945>
68. Bai, S., Kolter, J.Z., Koltun, V.: An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling (2018)
69. Sanderson, C., Curtin, R.: Armadillo: a template-based c++ library for linear algebra. *J. Open Source Software* 1(2), 26 (2016). Available from: <https://doi.org/10.21105/joss.00026>
70. Morgan, S., et al.: Tensorflow Addons Losses: Tripletsemihardloss. last updated: 2021-11-19. https://www.tensorflow.org/addons/tutorials/losses_triplet

How to cite this article: Argones Rúa, E., et al.: Discriminative training of spiking neural networks organised in columns for stream-based biometric authentication. *IET Biome.* 11(5), 485–497 (2022). <https://doi.org/10.1049/bme2.12099>