AFRL-RY-WP-TP-2023-0150

AN APPROACH TO TIME SERIES FORECASTING WITH SPIKING NEURAL NETWORKS (Preprint)

Trevor J. Bihl
Sensing Management Branch
Multi-Domain Sensing Autonomy Division

Davide L. Manna, Alex Vicente-Sola, Paul Kirkland, and Gaetano Di Caterina
University of Strathclyde

JULY 2023
Final Report

AIR FORCE RESEARCH LABORATORY
SENSORS DIRECTORATE
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7320
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE

REPORT DOCUMENTATION PAGE

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION.

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
July 2023	Journal Article Preprint	17 July 2023	17 July 2023

4. TITLE AND SUBTITLE

AN APPROACH TO TIME SERIES FORECASTING WITH SPIKING NEURAL NETWORKS (Preprint)

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
FA8655-20-1-7037	N/A	N/A
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**
N/A	N/A	N/A

6. AUTHOR(S)

Trevor J. Bihl (AFRL/RYAR)

Davide L. Manna, Alex Vicente-Sola, Paul Kirkland, and Gaetano Di Caterina (University of Strathclyde)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)		8. PERFORMING ORGANIZATION REPORT NUMBER
Sensing Management Branch Multi-Domain Sensing Autonomy Division Air Force Research Laboratory, Sensors Directorate Wright-Patterson Air Force Base, OH 45433-7320 Air Force Materiel Command, United States Air Forces	University of Strathclyde (UK) 16 Richmond Street Glasgow G1 1XQ	

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'SREPORT NUMBER(S)
Air Force Research Laboratory, Sensors Directorate Wright-Patterson Air Force Base, OH 45433-7320 Air Force Materiel Command, United States Air Force	Air Force Office of Scientific Research (AFOSR) 875 North Randolph Street Arlington, VA 22201	AFRL/RYAR	AFRL-RY-WP-TP-2023-0150

12. DISTRIBUTION/AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

PAO case number AFRL-2023-3024, Clearance Date 17 July 23. The U.S. Government is joint author of this work and has the right to use, modify, reproduce, release, perform, display, or disclose the work. Report contains color.

14. ABSTRACT

Analyzing time series data is critical in applications as di-verse as demand forecasting and speech recognition. Timely and energy-efficient predictions can play a key role on edge devices, where power requirements can be stringent, but rapid reactions are required. In re-cent years, several Deep Learning (DL) algorithms have been successfully applied to solve time series data problems; however, they inherently lack the ability to process time as a dimension and often have high SWaP (Size, Weight, and Power) demands. Spiking Neural Networks (SNNs) are regarded as a new avenue in which to solve time series problems but with lower-SWaP needs due to their more closely biologically inspired algorithmic architecture. In this work, we propose an SNN pipeline to process and forecast time series. We develop a novel data spike-encoding mechanism, and two loss functions that optimise the prediction of the time series in their original domain. Our data encoding system is inspired by NM event sensors as a means to increase the interoperability of our algorithm with NM technology as well as to prepare data to be processed by the SNN; our loss function takes into account the number of events transmitted throughout the network to keep a lower power consumption while ensuring convergence to top-level solutions. We develop an SNN in such a way that is compatible for hardware deployment on NM chips. We utilize the Panama short-term electricity load forecasting dataset and the electricity transformer temperature dataset to evaluate our solution on a univariate forecasting task. Collectively, our results show that SNNs can outperform the baseline model on the forecasting task, while also ensuring low power consumption and fast information processing. This underlines how SNNs can be a perfect fit for other time series forecasting tasks, including real-time signal processing and tasking on edge devices.

15. SUBJECT TERMS

neural network, time series, forecasting, spiking neural network, encoding loss functions, slayer, lava, backpropagation

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
Unclassified	Unclassified	Unclassified	SAR	14

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
Trevor Bihl	N/A

PREVIOUS EDITION IS OBSOLETE.

STANDARD FORM 298 (REV. 5/2020)

Prescribed by ANSI Std. Z39.18

An Approach to Time Series Forecasting With Spiking Neural Networks

Davide L. Manna[1], Alex Vicente-Sola[1], Paul Kirkland[1], Trevor J. Bihl[2], and Gaetano Di Caterina[1]

[1] Neuromorphic Sensor Signal Processing (NSSP) Lab,
Department of Electronic and Electrical Engineering,
University of Strathclyde, G1 1XW, Glasgow, UK
davide.manna@strath.ac.uk
http://nssp.eee.strath.ac.uk
[2] Air Force Research Laboratory (AFRL),
Wright-Patterson AFB, Ohio, USA

Abstract. Analyzing time series data is critical in applications as diverse as demand forecasting and speech recognition. Timely and energy-efficient predictions can play a key role on edge devices, where power requirements can be stringent, but rapid reactions are required. In recent years, several Deep Learning (DL) algorithms have been successfully applied to solve time series data problems; however, they inherently lack the ability to process time as a dimension and often have high SWaP (Size, Weight, and Power) demands. Spiking Neural Networks (SNNs) are regarded as a new avenue in which to solve time series problems but with lower-SWaP needs due to their more closely biologically inspired algorithmic architecture. In this work, we propose an SNN pipeline to process and forecast time series. We develop a novel data spike-encoding mechanism, and two loss functions that optimise the prediction of the time series in their original domain. Our data encoding system is inspired by NM event sensors as a means to increase the interoperability of our algorithm with NM technology as well as to prepare data to be processed by the SNN; our loss function takes into account the number of events transmitted throughout the network to keep a lower power consumption while ensuring convergence to top-level solutions. We develop an SNN in such a way that is compatible for hardware deployment on NM chips. We utilize the Panama short-term electricity load forecasting dataset and the electricity transformer temperature dataset to evaluate our solution on a univariate forecasting task. Collectively, our results show that SNNs can outperform the baseline model on the forecasting task, while also ensuring low power consumption and fast information processing. This underlines how SNNs can be a perfect fit for other time series forecasting tasks, including real-time signal processing and tasking on edge devices.

Keywords: time series · forecasting · spiking neural network · encoding · loss functions · slayer · lava · backpropagation

1 Introduction

Time series forecasting is a crucial task in various domains, including finance, meteorology, logistics, and many others. The goal of time series forecasting is to make predictions about the future values of a given time series based on past observations. Classical methods for time series forecasting include ARIMA, SARIMA, and Exponential Smoothing [1]. These methods are based on mathematical models that capture the statistical properties of the time series data, such as trends, seasonality, and residual errors. While these methods have been widely used for many years, they can struggle to accurately capture complex non-linear dependencies in the data. Recent advancements in the Deep Learning (DL) field have demonstrated promising results on a range of time series datasets [2, 3]. However, these solutions often employ complex systems to overcome their lack of ability to process time-dimensional data, and often have high demands in terms of memory and power.

Neuromorphic (NM) engineering is an emerging field that aims to leverage brain-like computations to create efficient systems [4]. In this context, spiking neural networks (SNNs) are a type of neural network that operate on the principles of spike-based information processing, which is thought to be more biologically plausible and energy-efficient than traditional, rate-based neural networks. SNNs use spiking neurons as processing units that only fire when necessary, leading to lower computational demands. Their time-based nature is ideal for time series data, allowing them to build sparse representations and propagate essential information. SNNs thus have the potential to provide accurate responses with low latency and power requirements, making them an excellent choice for time-series forecasting problems that heavily rely on time as a defining element of the data. Further to this, recent advances in the field have shown how SNNs can be successfully applied to a range of other tasks such as image classification and segmentation [5–10], where they demonstrate competitive results to their counterparts in conventional DL.

In this paper, we propose a novel NM approach for time series forecasting based on SNNs. Our approach incorporates concepts of spike-encoding and learning of weights and delays through back-propagation, which allow the network to adapt to different time scales and changing patterns in the data. Specifically, we develop an NM (vision sensor-inspired) spike-encoding system based on a population of neurons that can approximate the derivative of the input signal. Furthermore, we propose two novel loss functions that exploit the inter-spike interval and the significance of each spike as a result of the encoding. Through a series of experiments, we demonstrate that the SNN trained with our loss function can outperform the baseline model trained with vanilla SLAYER [11] learning rule, and successfully learns to emit spikes in such a way that optimises the quality of the reconstructed signal.

2 Related Works

A typical neuromorphic dataset consists of a series of events indexed by time [12]. As such, tasks derived from the use of these datasets can easily be regarded as a type of time series application. However, what is commonly regarded as time series is limited to specific datasets where the interest is in predicting future trends in data, especially in contexts such as finance, energy, and transportation, to name a few. These normally contain real-valued data, which is ill-suited for neuromorphic computation due to its spike-based processing nature. Therefore, encoding mechanisms are required. In [13], the authors do an excellent job of reviewing and testing several different real-to-spike encoding systems. The encoding system we present in this work is similar to the Send-on-Delta (SOD) method presented in [13]. However, they apply the encoding to the result of a spectrogram and cochleagram, and they consider the difference between the signal at the current timestep and the signal at the timestep when a spike was emitted last. In our case, we encode the signal itself and always consider two consecutive timesteps. Another difference is in the thresholding system. Because they use Short-Time Fourier Transforms (STFTs) and cochleagrams, they use two encoding neurons (positive and negative change) for each frequency bin, each with the same threshold. In our case, we define a baseline threshold and a set of multiplicative thresholds based on this. A further spike-encoding mechanism is developed in [14]. Here, the authors focus on the interval between two spikes to perform the encoding, and train their SNN using an evolutionary algorithm.

In [15], the authors perform a comparison of a conventional multi-layer perception (MLP) and long-short term memory (LSTM) with their spiking implementations on financial time series. In [16], the authors also consider financial data and compare a polychronous spiking network with conventional DL systems. However, the two works above do not perform any spike-encoding on the real-valued data.

3 Data Encoding and Processing

To approach the time series forecasting problem, we used the Panama Short-term electricity load forecasting (Panama) [17] and the Electricity Transformer Temperature with one hour resolution (ETTh1) [18] datasets. Both these datasets contain energy readings coupled with other information, such as temperature at the time of the reading and wind speed. In this work, we focus on a univariate forecasting problem; therefore, we only consider the electricity load readings and use them as the input variable and the target variable. In Fig. 1, we report an excerpt of the electricity load from the Panama dataset. The datasets above were collected using conventional (non-NM) sensors; therefore, they are composed of real-valued data points. Hence, for conversion into a spike representation, we design an encoding mechanism that draws inspiration from NM vision sensors [12]. These produce a positive or negative event whenever the change in lighting in the scene is above a certain value. As a result, information can be encoded in

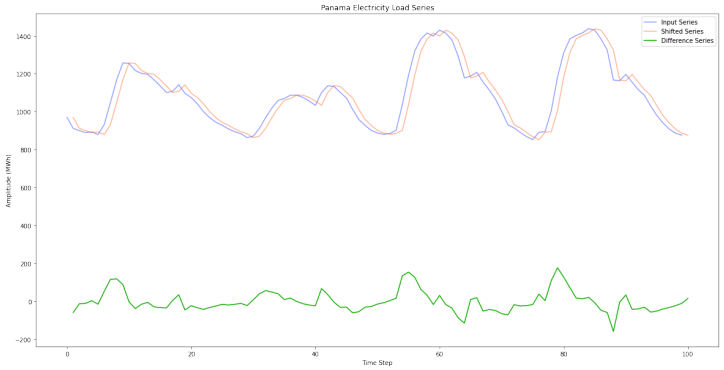


Fig. 1: Excerpt from the Panama dataset. The blue and orange lines are the signal and the lag-1 version of the signal respectively. The green line is the difference signal.

the timing of such events and a higher degree of sparsity is achieved. Drawing inspiration from this, our encoding mechanism transforms a real-valued input into a sequence of spikes emitted by a population of neurons. Each neuron in such a population is responsible for emitting a spike whenever the change in the signal is above a certain threshold $V_{th}^{(i)}$, both in a positive and negative direction. In other words, if the input signal has a strong enough variation from time step t to time step $t+1$, one of the neurons will emit a spike; if no variation happens, or if the variation is too small, no spike will be emitted, hence increasing sparsity. This is achieved by considering the difference of the input signal with a lagged (delayed) version of itself by one time step. A representation of this can be found in Fig. 1 (green line).
The so-obtained encoding can be easily reversed by considering the threshold each neuron is associated with. Similarly to a quantization problem, the fidelity of the reconstruction of the original signal from the encoding is proportional to the granularity (number of neurons and values of thresholds) of the encoding.

In order to quantify the information loss from the encoding, in Table 1, we report an exploratory search of the reconstruction mean squared error (MSE) on the Panama dataset using a different number of encoding neurons with different thresholds. We used different sets of multiplicative thresholds for our encoding, i.e. each neuron was assigned a value $n \in Z \setminus \{0\}$, then multiplied by the base threshold, so that the actual threshold would be $V_{th}^{(i)} = n \cdot V_{th}$. The selection of the base thresholds was aided by the analysis of the variations present in the dataset. By visualising the variations that take place and their number of occurrences (Fig. 2), we get a rough estimate of the ranges threshold could be varied in. Interestingly, as highlighted in Table 1, the minimum MSE is not achieved by using the largest number of neurons and lowest thresholds, thus highlighting how threshold selection can be a crucial step. Fig. 3 shows an excerpt using the parameters relative to the minimum MSE value obtained. As can be

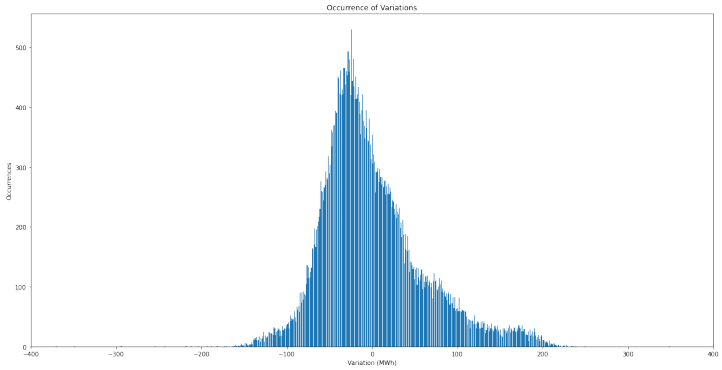


Fig. 2: Bar chart of the variations in the Panama dataset. Each bar represents the number of times that amount of change is found in the data.

Table 1: Mean Squared Error between the original signal and reconstructed signal using different encodings. Neuron multipliers should be interpreted as referring to two neurons each (positive and negative versions of each). For instance, (1,2) with base threshold 9 refers to neurons with thresholds (-18, -9, 9, 18). The range(1, 60, step=2) indicates a range of values starting from 1 and increasing by two up to 60. In bold, are the best MSE values from the reconstructions.

Neuron Multipliers	**Base Threshold**				
	9	**13**	**23**	**33**	**53**
(1)	2852.88	2578.11	2033.40	1690.30	1476.04
(1, 2)	2259.65	1846.16	1187.73	910.15	944.43
(1, 2, 3)	1788.29	1328.03	731.51	576.37	814.01
(1, 2, 3, 4)	1417.18	962.62	478.96	433.10	800.42
(1, 2, 3, 4, 5)	1126	704.65	337.55	380.05	791.72
(1, 2, 3, 4, 5, 10)	494.50	**313.76**	**311.93**	354.15	773.62
(1, 2, 3, 4, 5, 10, 20, 30)	347.37	**274.67**	**283.07**	338.37	767.10
range(1, 60, step=2)	509.33	938.30	2530.13	4449.87	6885.43

observed, the reconstructed signal follows rather faithfully the original signal, except from it incurring in some information loss for certain ranges of values.

3.1 Approximation of the Derivative

Before the application of the spiking function, our encoding method can be formally expressed as the average variation of the amplitude of the signal between two points in time:

$$m = \frac{x(t + \Delta t) - x(t)}{\Delta t}, \tag{1}$$

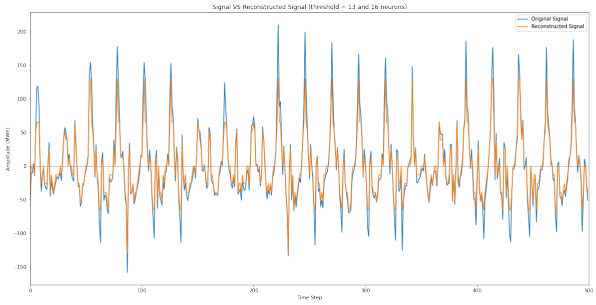


Fig. 3: Extract from the Panama data after differencing versus its reconstruction.

where $x(t)$ is the input signal and Δt is the time step. We observe that equation (1) denotes the slope of signal x around time t. Interestingly, as Δt becomes smaller, (1) becomes an increasingly better approximation of the derivative of signal over time:

$$\frac{d}{dt}x(t) \approx \frac{x(t+\Delta t) - x(t)}{\Delta t}. \tag{2}$$

Thus, assuming that $x(t)$ is smooth around time t, our encoding method approximates the instantaneous rate of change, or the derivative, of $x(t)$. In practical terms, the goodness of such approximation is constrained by the choice of thresholds for the neurons, and by the time resolution (sampling frequency) of the datasets. However, considering a single time step interval and a small enough threshold, such an approximation can be relatively accurate. In Fig. 4, we intuitively show the goodness of the approximation when the input signal is a sine wave (hence with a cosine derivative). In the figure, the spikes closely follow the evolution of the derivative (green line) over time, with each spike representing a different amount of variation in the original signal (blue line).

By means of our encoding system, we obtain two major advantages. Firstly, by taking the (approximate) derivative of the signal, we are looking at its rate of change. This means that we can determine how fast the signal is changing and in which direction, regardless of its absolute value at a given time. The rate of change of a signal can be a crucial indicator of underlying patterns, and analyzing it can provide us with deeper insights into the behavior of the signal. In addition, explicitly using the derivative can also expose information about the second derivative of the signal for the SNN to learn. This can help highlighting the presence of inflection points, where the rate of change of the signal changes from increasing to decreasing, or vice versa, hence possibly allowing learning of more robust representations. Secondly, by performing this operation, we are effectively applying a differencing transform to the input signal. This, in the context of time series analysis, helps increase stationarity in the signal, hence re-

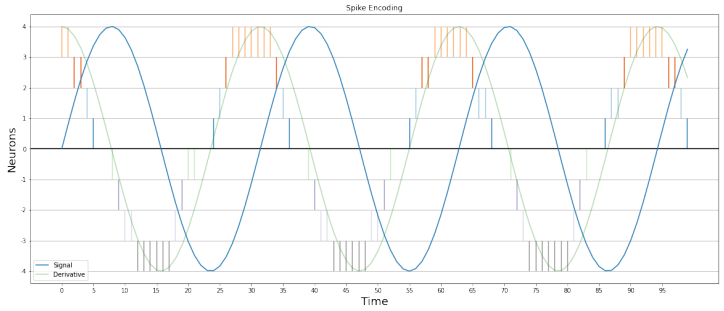


Fig. 4: Example of a sinusoidal input signal with its derivative and spike-encoding.

ducing the effect of trends and seasonal patterns. As a result, the representation learning and the forecasting of the signal can be more precise and reliable [19–21].

3.2 Learning and Loss Functions

To leverage the spiking signal obtained through the encoding mechanism described in the previous section, we develop a simple two-layer fully connected neural network architecture using Intel's LAVA framework [22]. More specifically, the SNN consists of two layers of fully connected Current Based (CuBa) Leaky Integrate-and-Fire (LIF) [23] neurons, representable by the following discrete system:

$$\begin{aligned} v[t] &= (1-\alpha)v[t-1] + x[t] \\ s[t] &= v[t] \geq v_{th}, \end{aligned} \tag{3}$$

where $v[t]$ is the discrete-time internal state (voltage) of the neuron, α is a leakage factor, $x[t]$ is the discrete-time input, $s[t]$ is the spiked value a time t, and v_{th} is the threshold of the neuron. The neuron is also paired with a reset mechanism that resets the voltage to zero whenever a spike is emitted. While it has been shown that better-performing spiking neuron model alternatives might exist [24], the LIF neuron is arguably the most widely used in the literature, hence making it a good choice for future benchmarking purposes. Other than this, it can represent a valid option to increase efficiency due to its reduced computational complexity.

We adopt an advanced version of the Spike Layer Error Reassignment (SLAYER) [11] learning rule to train our network in a supervised manner. The standard way of utilising SLAYER is with the loss function defined by equations (6) and (7) in [11], named SpikeTime in the LAVA framework. Through this, the trains of target spikes and output spikes are convolved in the time dimension with a Finite Impulse Response (FIR) exponential kernel and then compared using MSE. By convolving with the FIR kernel, the loss aims to aid the resolution of the credit assignment problem through time. Further to this, we also experiment with two

different loss functions that we design specifically for this task. The ISILoss draws inspiration from the concept of minimizing the differences in the inter-spike intervals (ISI) in the target and output spike train; the sDecodingLoss leverages the information from the encoding paradigm to decode the signal and compare the reconstructed versions of the target and output.

ISILoss Function As outlined above, the ISILoss function is inspired by the concept of minimizing the differences in inter-spike intervals between two spike trains. Ideally, the ISIs should be computed for each spike train and then compared to each other. However, this is non-trivial in a back-propagation environment for several reasons. Two spike trains may have different numbers of spikes and, hence, different numbers of ISIs to compare. A possible solution for this is to adopt placeholders with pre-assigned values where the number of spikes differs. But this can be time-consuming and sensitive to the chosen pre-assigned value for the interval. Furthermore, this will likely include non-derivable operations, which could break the gradient calculation chain and result in unstable behaviours. For this reason, we adopted a heuristic method based on transforming the spike trains by means of derivable operations only. Specifically, we define a time-step vector R such that:

$$R = [1, 2, \ldots, N],$$

where N is the number of time steps in the spike trains. We use R to re-scale each spike in the spike train by the time step at which it occurred:

$$s_R(t) = s(t) \odot R, \tag{4}$$

where $\odot$ denotes the element-wise multiplication (Hadamard product) between the spike train $s(t)$ and R. Finally, we perform a cumulative sum operation on the re-scaled spike train. To do this, we define a unitary upper triangular matrix T of size $N \times N$ and perform matrix multiplication with $s_R(t)$. By doing this, we obtain a vector of size N that contains the cumulative sum of $s_R(t)$. The result is then normalized with respect to R.

$$s_{cs}(t) = (s_R(t) \cdot T) \odot \frac{1}{R}. \tag{5}$$

The final loss is calculated as a mean squared error of the resulting vectors:

$$L(s_{cs}(t), \hat{s}_{cs}(t)) = \frac{1}{N} \sum_{i=0}^{N} (s_{cs}^{(i)}(t) - \hat{s}_{cs}^{(i)}(t))^2, \tag{6}$$

where $\hat{s}_{cs}$ denotes the target spike train, transformed as discussed. The idea behind such a heuristic is that by utilizing a cumulative sum, all the time steps present in the spike train contribute to the final loss calculation by some value (likely) different than zero. This allows carrying along some information about the cumulative time of the last spikes with each time step, hence assigning some weight to their role in the spike train, even if no spike was present.

DecodingLoss Function The DecodingLoss is a cost function that builds on top of an other piece of knowledge from the time series encoding: the meaning of each neuron's firing. As a matter of fact, by means of our encoding, each neuron will be assigned a specific threshold and encode values that fall in the range $V_{th}^{(i)} \leq x(t) < V_{th}^{(i+1)}$. We use this to reconstruct a signal starting from some output spike train $s(t)$, and compare it with the decoded target spike train $\hat{s}(t)$. This is possible by following a similar paradigm as in Section 3.2. We start by defining a vector of values that correspond to each neuron's threshold:

$$V = \left[-V_{th}^{(M/2)}, \ldots, -V_{th}^{(1)}, V_{th}^{(1)}, \ldots, V_{th}^{(M/2)}\right]^T,$$

where $V_{th}^{(i)}$ denotes the (positive) threshold of neuron i. We also define a convenience unitary vector A of size M that we can use to perform a neuron-wise addition per each time step. In our experiment, we test both with and without this step in the DecodingLoss function. Finally, we utilised the unary upper triangular matrix T that we previously defined to perform a cumulative sum. The final reconstructed output is thus obtained:

$$s_{rec}(t) = (A \cdot (s(t) \odot V)) \cdot T, \tag{7}$$

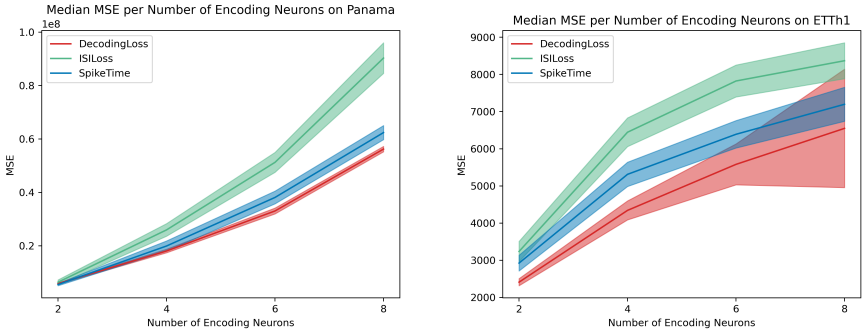
where $s_{rec}(t)$ is the reconstructed output and can be either of size N or $M \times N$ depending on whether the matrix multiplication by A was performed. Finally, the MSE is computed on the reconstructed output and target output:

$$L(s_{rec}(t), \hat{s}_{rec}(t)) = \frac{1}{N}\sum_{i=0}^{N}(s_{rec}^{(i)}(t) - \hat{s}_{rec}^{(i)}(t))^2, \tag{8}$$

where $\hat{s}_{rec}(t)$ denotes the reconstructed target signal. In this case, the cumulative sum has a more physical meaning than the ISILoss, as each value thus obtained directly corresponds to the value of the reconstructed signal. As a result, we can compare the reconstructed signal with the original signal in the original domain and compute the MSE loss on the final outcome of the obtained spikes. This approach is in line with the metric utilized to evaluate the quality of the results. Consequently, it theoretically allows for the learning of spike trains that, even if not identical to the target spikes, can nevertheless be an equivalently good approximation of the target signal, hence easing the optimization problem by expanding the landscape of acceptable solutions.

4 Results

To evaluate our system, we design a set of experiments using different settings. In each experiment, the data is encoded into spikes through the methodology described in Section 3. The encoded data is then split into smaller segments of length 128 or 256, with each consecutive segment having an overlapping of the 75% with the previous one. Hyper-parameter tuning of the encoding system is



(a) Error on Panama Dataset (b) Error on ETTh1 Dataset

Fig. 5: Reconstruction Mean Squared Error (MSE) using different loss functions and the number of encoding neurons. The solid lines represent the median calculated over at least 150 experiments, whereas the shaded areas represent the interquartile range.

performed here by considering a trade-off between overall reconstruction error (see Table 1) and number of encoding neurons. We design experiments in order to consider different numbers of encoding neurons, different segment sizes and different loss functions. To attain a higher robustness in our results, we run each experiment setting at least 150 times, and train the SNN for a total of 500 epochs. Collectively, the amount of performed experiments exceeds 15000. To evaluate the quality of the generated output, we decode the output spikes to obtain a re-construction of the signal and then compare it with the target sequence. The totality of the experiments is averaged and conveyed, for ease of representation and brevity, in Fig. 5a and 5b. From our results, we can observe that, overall, the SNNs trained using the DecodingLoss function achieve lower reconstruction MSE than the SpikeTime and ISILoss. The ISILoss shows considerably higher error with respect to its counterparts, except in the case of few-neurons encoding in the Panama dataset where the overall error diminishes. The SpikeTime loss manages to obtain good performances and, in some cases (see overlapping shaded areas in Fig. 5b), even perform better than the DecodingLoss. An interesting observation is a difference in the spread of the MSE using the DecodingLoss in the two datasets. While in the Panama dataset, the results seem to be rather consistent in terms of MSE, in the ETTh1, the spread widens abruptly as the number of neurons increases. We believe that this could be linked to the DecodingLoss assigning increasing values to the spikes emitted by the neurons. Because of this, when the number of neurons increases, the reconstruction error for some of them will reflect such higher values. Finally, we observe that the forecasting error increases monotonically with the number of encoding neurons for all the loss functions. This, however, is not surprising, because a higher number of

encoding/decoding neurons translates into more spike trains to be learnt by the SNN, which arguably represents a more complex task to solve.

5 Conclusions

In this work, we approached the problem of time series forecasting using SNNs. Our approach included utilising a novel encoding scheme that approximates the derivative of an input time series (or signal) with a population of spiking neurons. Further to this, we develop a simple, yet effective, SNN trained with the SLAYER learning rule, and define two novel loss functions, the ISILoss, and the DecodingLoss functions. We perform over 15000 experiments employing different combinations of hyperparameters, dataset splits and number of encoding neurons. We train our SNN using our novel loss functions and SLAYER's own loss function, SpikeTime. Our results show that SNNs trained using the DecodingLoss function in combination with our encoding scheme achieves better results than the other two cases. Thanks to the use of the LAVA framework, our implementation can also be easily deployed on NM chips and be evaluated on real hardware to achieve full SWaP gain from NM technologies. Future work will be the extension of our system to multi-variate time series forecasting, and the comparison of our solutions with other state-of-the-art approaches in conventional deep learning.

References

1. Gregory C. Reinsel Gwilym M. Jenkins Greta M. Ljung Box, George E. P. Box. *Time Series Analysis 5e*. John Wiley Sons, June 2015.
2. Pedro Lara-Benítez, Manuel Carranza-García, and José C. Riquelme. An experimental review on deep learning architectures for time series forecasting. *International Journal of Neural Systems*, 31(03):2130001, feb 2021.
3. Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning : A systematic literature review: 2005–2019. *Applied Soft Computing*, 90:106181, may 2020.
4. Dennis V Christensen, Regina Dittmann, Bernabe Linares-Barranco, Abu Sebastian, Manuel Le Gallo, Andrea Redaelli, Stefan Slesazeck, Thomas Mikolajick, Sabina Spiga, Stephan Menzel, et al. 2022 roadmap on neuromorphic computing and engineering. *Neuromorphic Computing and Engineering*, 2(2):022501, may 2022.
5. Alex Vicente-Sola, Davide L. Manna, Paul Kirkland, Gaetano Di Caterina, and Trevor Bihl. Evaluating the temporal understanding of neural networks on event-based action recognition with dvs-gesture-chain. September 2022.
6. Emre O. Neftci, Charles Augustine, Somnath Paul, and Georgios Detorakis. Event-driven random back-propagation: Enabling neuromorphic deep learning machines. *Frontiers in Neuroscience*, 11, jun 2017.
7. Chethan M. Parameshwara, Simin Li, Cornelia Fermüller, Nitin J. Sanket, Matthew S. Evanusa, and Yiannis Aloimonos. Spikems: Deep spiking neural network for motion segmentation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3414–3420, 2021.

8. Paul Kirkland, Gaetano Di Caterina, John Soraghan, and George Matich. SpikeSEG: Spiking segmentation via STDP saliency mapping. In *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, jul 2020.
9. Paul Kirkland, Davide Manna, Alex Vicente, and Gaetano Di Caterina. Unsupervised spiking instance segmentation on event data using STDP features. *IEEE Transactions on Computers*, 71(11):2728–2739, nov 2022.
10. Youngeun Kim, Joshua Chough, and Priyadarshini Panda. Beyond classification: directly training spiking neural networks for semantic segmentation. *Neuromorphic Computing and Engineering*, 2(4):044015, dec 2022.
11. Sumit B Shrestha and Garrick Orchard. Slayer: Spike layer error reassignment in time. *Advances in neural information processing systems*, 31, 2018.
12. Christian Brandli, Raphael Berner, Minhao Yang, Shih-Chii Liu, and Tobi Delbruck. A 240 $\times$ 180 130 dB 3 μs latency global shutter spatiotemporal vision sensor. *IEEE Journal of Solid-State Circuits*, 49(10):2333–2341, oct 2014.
13. Sidi Yaya Arnaud Yarga, Jean Rouat, and Sean Wood. Efficient spike encoding algorithms for neuromorphic speech recognition. In *Proceedings of the International Conference on Neuromorphic Systems 2022*, pages 1–8, 2022.
14. V. Sharma and D. Srinivasan. A spiking neural network based on temporal encoding for electricity price time series forecasting in deregulated markets. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2010.
15. Karolina Mateńczuk, Agata Kozina, Aleksandra Markowska, Kateryna Czerniachowska, Klaudia Kaczmarczyk, Paweł Golec, Marcin Hernes, Krzysztof Lutosławski, Adrianna Kozierkiewicz, Marcin Pietranik, Artur Rot, and Mykola Dyvak. Financial time series forecasting: Comparison of traditional and spiking neural networks. *Procedia Computer Science*, 192:5023–5029, 2021.
16. David Reid, Abir Jaafar Hussain, and Hissam Tawfik. Financial time series prediction using spiking neural networks. *PLoS ONE*, 9(8):e103656, aug 2014.
17. Ernesto Aguilar Madrid and Nuno Antonio. Short-term electricity load forecasting with machine learning. *Information*, 12(2):50, jan 2021.
18. Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Conference*, volume 35, pages 11106–11115. AAAI Press, 2021.
19. Robert Hogenraad, Dean P. McKenzie, and Colin Martindale. The enemy within: Autocorrelation bias in content analysis of narratives. *Computers and the Humanities*, 30(6):433–439, 1997.
20. Petrus M. T. Broersen. *Automatic Autocorrelation and Spectral Analysis*. SPRINGER NATURE, April 2006.
21. Yixiong Xiao and Peng Gong. Removing spatial autocorrelation in urban scaling analysis. *Cities*, 124:103600, may 2022.
22. Lava: A software framework for neuromorphic computing. https://github.com/lava-nc/lava, 2021.
23. L. Lapicque. Recherches quantitatives sur l'excitation electrique des nerfs traitée comme une polarization. *Journal de Physiologie et Pathologie General*, 9:620–635, 1907.
24. Davide Liberato Manna, Alex Vicente-Sola, Paul Kirkland, Trevor Bihl, and Gaetano Di Caterina. Simple and complex spiking neurons: perspectives and analysis in a simple stdp scenario. *Neuromorphic Computing and Engineering*, 2022.