An Approach to Time Series Forecasting With Derivative Spike Encoding and Spiking Neural Networks

Davide Manna, Gaetano Di Caterina, Alex Vicente Sola, Paul Kirkland
Neuromorphic Sensor Signal Processing Lab, EEE Dept, University of Strathclyde, Glasgow, UK
{davide.manna, gaetano.di-caterina, alex.vicente-sola, paul.kirkland} @strath.ac.uk

Abstract

Timely and energy-efficient time series forecasting can play a key role on edge devices, where power requirements can be stringent. Spiking Neural Networks (SNNs) are regarded as a new avenue in which to solve time series problems, but with lower SWaP (Size, Weight, and Power) needs. We propose an SNN pipeline to process and forecast time series, developing a novel data spike-encoding mechanism and two loss functions that optimise the prediction of the upcoming spikes. Our approach encodes a signal into sequences of spikes that approximate its derivative, preparing the data to be processed by the SNN, while our proposed loss functions account for the reconstruction of the output spikes into a meaningful value to promote convergence to top-level solutions. Results show that our solution can effectively learn from the encoded data and the SNN trained with our loss function can outperform the same model trained with SLAYER's default loss.

Keywords: time series, forecasting, spiking neural networks, neuromorphic, differencing, derivative

1. Introduction

Time series forecasting is a crucial task in various domains, including finance, logistics, condition monitoring 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. Time series are composed of data points indexed by time, and can be broadly classified into stationary and non-stationary categories (Box et al. (2015) and Hyndman and Athanasopoulos (2018)). Stationary time series are characterized by statistical properties such as mean, variance, and autocorrelation that remain constant over time. This consistency simplifies the analysis and forecasting processes, making stationary time series ideal for applying many statistical methods. Non-stationary time series, on the other hand, exhibit changing statistical properties over time. These changes can manifest as trends, seasonal patterns, or structural breaks, which complicate the modelling process. Non-stationary time series can lead to unreliable and misleading results if not appropriately addressed. Therefore, transforming non-stationary time series into stationary ones is a crucial step in time series analysis. Techniques such as differencing, detrending, and seasonal adjustment are commonly used to achieve stationarity (Hyndman and Athanasopoulos (2018)). Classical methods of time series forecasting are well-established and widely used due to their simplicity and interpretability. These methods include the Autoregressive Integrated Moving Average (ARIMA) (Box et al. (2015)), which combines autoregression, differencing, and moving averages to model time series data, Seasonal ARIMA (SARIMA), and Exponential Smoothing (Box et al. (2015)). These methods are based on mathematical models that aim to capture the statistical properties of the time series data and often make use of transforms to render data more stationary in the process. 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 (Lara-Benitez et al. (2021) and Sezer et al. (2020)). Notable results have been achieved by Recurrent Neural Networks (RNNs), Long-short-Term Memory (LSTM) networks, and Transformers (Bandara et al. (2019), Hua et al. (2019), and Wen et al. (2023)). However, these solutions

URI: https://hdl.handle.net/10125/109720
978-0-9981331-8-8

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 (Douglas et al. (2022), Li and Sainath (2017), Nan et al. (2020), Thompson et al. (2020), and Wang et al. (2019)).

Neuromorphic (NM) engineering is an emerging field that aims to leverage brain-like computations to create efficient systems (Christensen et al. (2022)). This has been made possible by the technological advancements in NM vision sensors (Brandli et al. (2014)), and NM chips Akopyan et al. (2015), Davies et al. (2018), Furber et al. (2014), and Orchard et al. (2021), which enable sparse, asynchronous perception and processing. Such sparsity and asynchronicity can be fully exploited by means of Spiking Neural Networks (SNNs), a type of neural network that operates on the principles of spike-based information processing. In the context of time series forecasting, however, their inherent time-based nature is what puts them forward as a potential alternative to the current approaches in the literature. SNNs, in fact, use spiking neurons as processing units that accumulate information depending on their timing of arrival and only fire when necessary, thus inherently involving time in computations and leading to lower computational demands. SNNs can thus build sparse temporal representations and propagate essential information only, and harness the potential to provide accurate responses with low latency and power requirements. Because of this, they can be 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 (Kim et al. (2022), Kirkland et al. (2020), Kirkland et al. (2022), Neftci et al. (2017), and Parameshwara et al. (2021), Vicente-Sola et al. (2022)), where they demonstrate competitive results to their counterparts in conventional DL.

In this paper, we propose to expand the field of application of NM computing to that of time series. Our approach aims to answer the question of whether it is possible to incorporate a classical time-series preprocessing step like differencing into a spike-encoding mechanism that makes data amenable to SNNs. Furthermore, we explore the possibility of leveraging biologically-inspired concepts and the information from the encoding system to devise ad-hoc loss functions that could optimize the predictions. In this effort, we show how our approach, inspired by NM vision sensors and by the differencing transform, not only manages to do this by means of a population of spiking neurons but also how the proposed encoding approximates the derivative of a signal, given the right assumptions. We incorporate this in a solution featuring a simple SNN embedding weight and delays learning through the Spike Layer Error Reassignment in Time (SLAYER) (Shrestha and Orchard (2018)), which in turn allows the network to adapt to different time scales and changing patterns in the data. Furthermore, we propose two loss functions for the SLAYER learning rule, that exploit the distance between two consecutive output spikes (inter-spike interval) and the significance of each spike as a result of the encoding respectively. Through a series of experiments, we show that SNNs are a viable option to process time series data and that our signal-reconstruction-based loss function can outperform the same model trained using the default loss function in SLAYER as well as a SARIMA-based approach, therefore successfully learning to predict the next upcoming spike in a way that optimizes the quality of the decoding into the signal's original domain.

2. Related Works

A typical neuromorphic dataset consists of a series of events indexed by time (Brandli et al. (2014)). 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 continuous data, which is ill-suited for neuromorphic computation due to its spike-based processing nature. Therefore, appropriate encoding mechanisms are required. In Yarga et al. (2022), 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 Yarga et al. (2022). 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 Sharma and Srinivasan (2010). Here, the authors focus on the

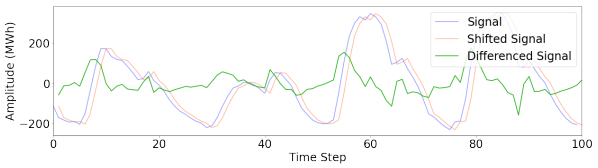

Figure 1: Excerpt from the Panama dataset. The blue and orange lines are the signal and the lag-1 version of the signal respectively (with their means subtracted). The green line is the difference signal.

interval between two spikes to perform the encoding and train their SNN using an evolutionary algorithm. In Mateńczuk et al. (2021), the authors perform a comparison of a conventional multi-layer perception (MLP) and LSTM with their spiking implementations on financial time series. In Reid et al. (2014), the authors delve into the application of a specific type of SNN, the Polychronous Spiking Network, for financial time series prediction. This demonstrates the feasibility and effectiveness of SNNs in handling time series data, particularly in the domain of financial forecasting. Concerning encoding, they employ a one-to-one encoding system between discretized time series values and encoding neurons. The authors in Gao et al. (2021) utilize an STDP-trained SNN to predict price spikes in price time series and deal with high-frequency data, demonstrating encouraging results. Here, data undergoes a Poisson encoding, a type of rate-based encoding utilized to transform real-valued data into random sequences of spikes. In Liu et al. (2024) the authors propose a Single-Modal Pulse Encoding Module, composed of an image feature extractor (which considers plots of time series) and an SNN LIF-based module. The overall approach seems to attain state-of-the-art accuracy levels, thus highlighting the potential of SNNs in the time series domain. Nevertheless, the encoding scheme could potentially introduce a non-negligible overhead, as it requires several feature extraction steps before the encoding can finally take place.

3. Data Encoding and Processing

To approach the time series forecasting problem, we used the Panama Short-term electricity load forecasting (Panama) (Madrid and Antonio (2021)) and the Electricity Transformer Temperature with one-hour resolution (ETTh1) (Zhou et al. (2021)) 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 propose a novel system for forecasting, and therefore we focus on a univariate forecasting problem for simplicity; therefore, we only consider the electricity load readings and the oil temperature (the target variable in the ETTh1 dataset) and use them as the input variable and the target variable. In Figure 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 (Brandli et al. (2014)). 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 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 Figure 1 (green line). The result is a set of spike trains, i.e. a sequence of spikes indexed by time, emitted by each neuron in the encoding layer. 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 (i.e. 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 $n \in \mathbf{Z} \setminus \{0\}$, 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 (Figure 2), we get a rough estimate of the ranges threshold could be varied in. Interestingly,

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	**274.81**	**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

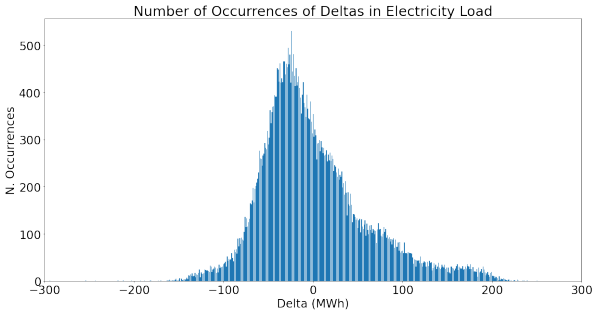

Figure 2: Bar chart of the variations in the Panama dataset. Each bar represents the number of times that amount of change is found in the data.

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.

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:

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

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 $\frac{d}{dt}x(t) \approx m$. Therefore, 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 (i.e. sampling frequency) of the datasets. However, by considering a single time step interval and a small enough threshold, such an approximation can be relatively accurate. In Figure 3, 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 highlight 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 reducing the effect of trends and seasonal patterns. As a result, the representation learning and the forecasting of

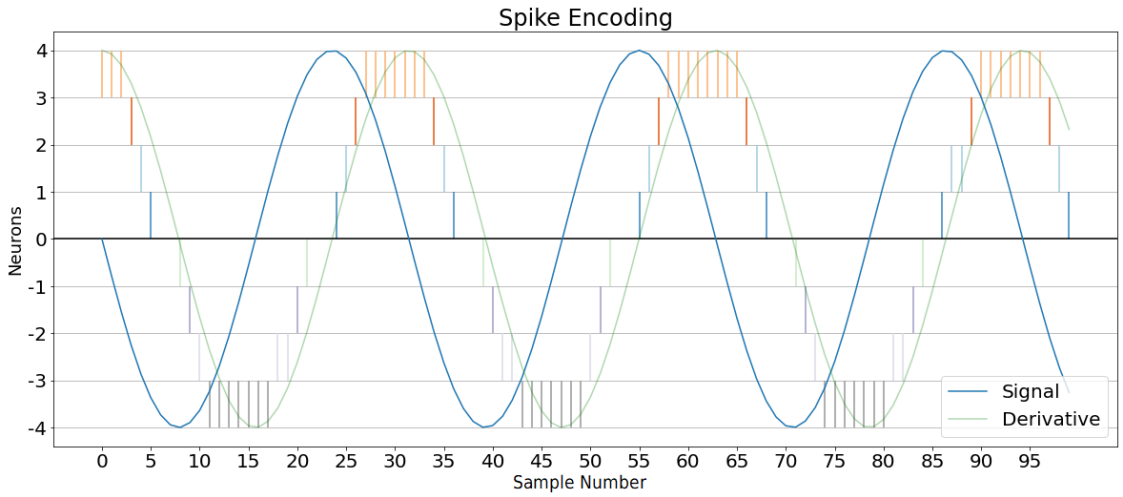

Figure 3: Example of a sinusoidal input signal with its derivative and spike-encoding. The sin (blue) and cos derivative (light green) have been scaled to match the height of the plot. Note the concurrent presence of high-grade spikes (upper and lower rows) with higher values in the cosine and the presence of lower-grade spikes when the derivative approaches zero.

the signal can be more precise and reliable (Broersen (2006), Hogenraad et al. (1997), and Xiao and Gong (2022)).

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 (Intel (2021)). More specifically, the SNN consists of two layers of fully connected Current Based (CuBa) Leaky Integrate-and-Fire (LIF) (Lapicque (1907)) neurons, representable by the following discrete system:

$$\begin{aligned} v[t] &= (1-\alpha)v[t-1] + x[t] \\ s[t] &= v[t] \geq v_{th}, \end{aligned} \tag{2}$$

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 at 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 (Manna et al. (2022)), 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) (Shrestha and Orchard (2018)) 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 (Shrestha and Orchard (2018)), 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 trains; the DecodingLoss leverages the information from the encoding paradigm to decode the signal and compare the reconstructed versions of the target and output.

3.2.1. 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. However, 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 $R = [1, 2, \ldots, N]$, 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:

$$s_R(t) = s(t) \odot R, \tag{3}$$

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.

$$s_{cs}(t) = (s_R(t) \cdot T) \odot \frac{1}{R}. \tag{4}$$

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

$$L(s_{cs}(t), \hat{s}_{cs}(t)) = \frac{1}{N} \sum_{i=0}^{N} (s_{cs}^{(i)}(t) - \hat{s}_{cs}^{(i)}(t))^2, \tag{5}$$

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.

3.2.2. 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 3.2.1. We start by defining a vector of values that correspond to each neuron's threshold:

$$V = \left[-V_{th}^{(M/2)}, \ \ldots, \ -V_{th}^{(1)}, \ V_{th}^{(1)}, \ \ldots, \ V_{th}^{(M/2)}\right]^T, \tag{6}$$

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 75% with the previous one. Overall, the initial 80% of the datasets are utilized for training, thus leaving the remaining unseen 20% available for testing. Hyper-parameter tuning of the encoding system

Table 2: Collated results on the Panama dataset with each loss function against the number of encoding neurons. In each cell, the average reconstruction MSE (rounded to the closest integer) is reported, with the best results highlighted in bold for each different number of encoding neurons.

Loss Fn/N. Neurons	4	8	12	16
DecodingLoss	302	**938**	**1719**	**2926**
ISILoss	326	1350	2669	4718
SpikeTime	**283**	1033	1988	3252

Table 3: Collated results on the ETTh1 dataset with each loss function against the number of encoding neurons. In each cell, the average reconstruction MSE is reported, with the best results highlighted in bold for each different number of encoding neurons. The reported values are to be multiplied by 10^{-3}.

Loss Fn/N. Neurons	4	8	12	16
DecodingLoss	**2.52**	**4.51**	**5.77**	**6.55**
ISILoss	3.36	6.71	8.16	8.72
SpikeTime	3.04	5.54	6.65	7.49

is performed here by considering a trade-off between overall reconstruction error (see Table 1) and number of encoding neurons. For each input segment, a target one is generated by looking at one time step ahead, effectively creating a challenge for the network to learn the upcoming spike. 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 reconstruction of the forecast signal and then compare it with the target sequence. The totality of the experiments is averaged and reported in Table 2 and 3. 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 a 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 achieve good performances and, in the 4-encoding-neurons case in the Panama dataset, even performs better than the DecodingLoss. An interesting observation is that the forecasting error increases monotonically with the number of encoding neurons for all the loss functions. This might seem counterintuitive given the higher level of representation from the encoding, however, is not surprising: due to the way the pipeline is conceived and training is performed, 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. As a matter of fact, by comparing the reconstruction of the prediction with the reconstruction of the target, and not the target itself, the quantization error that would normally be incurred by a lower number of neurons is disregarded, thus allowing for a simplified task and, conversely, for a more difficult one when considering more neurons. Motivated by the results above, the DecodingLoss-based solution is selected for further comparisons with other more classical methods like SARIMA. SARIMA is a model that is able to account for trends and seasonality in the data, effectively making it a suitable option amongst the classical methods for the data employed in this study. Through this comparison, the objective is to provide a more solid context as to what the performance of the proposed solution signifies and to provide a useful benchmarking point for future reference. In order to use SARIMA, a prior must be known regarding the periodicity of the data, which is inputted into the model prior to fitting. Furthermore, in order to ensure a fair comparison, a hyperparameter search is performed to find the best-fitting parameters. Experiments are carried out in conditions similar to the experiments above, with the task set at predicting the next time step in the series. To obtain single-time step predictions, the model is refitted on the newly observed data every time a prediction is made. Results on the Panama dataset show that the best SARIMA found via the parameter search step achieves an average MSE between predicted and actual points of 4948.31, whereas the proposed method reaches an average of 2531.03. An example of a prediction relative to these results is given in Figure 4. On the ETTh1 dataset, the SARIMA model achieves an MSE of 1.72, whereas the proposed model sits at 0.39. An example prediction of this is reported in Figure 5.

5. Conclusions

In this work, we addressed the problem of time series forecasting using SNNs. Our approach included utilising a novel encoding scheme that incorporates the concept of signal differencing, which is helpful in making time series more stationary. Given the right conditions, we also show how this methodology of information encoding approximates the derivative of an input time series (or signal). Further to this, we develop

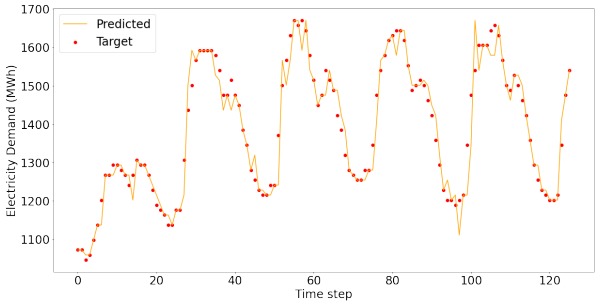

Figure 4: Example from the Panama dataset of values predicted by the proposed solution trained with the DecodingLoss function. The red dots represent the target data points, whereas the orange line represents the reconstructed prediction.

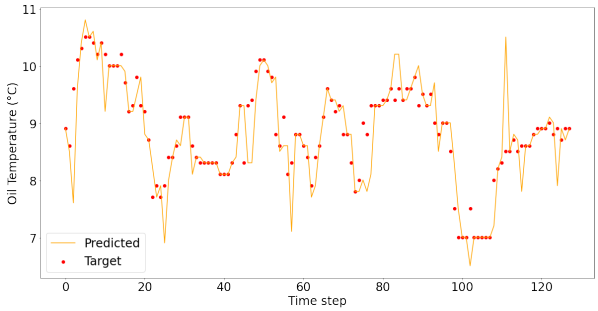

Figure 5: Example from the ETTh1 dataset of values predicted by the proposed solution trained with the DecodingLoss function. The red dots represent the target data points, whereas the orange line represents the reconstructed prediction.

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 achieve better results than the other two cases. Moved by these results, we proceed with comparing our DecodingLoss-based solution with a popular conventional approach to forecasting: the SARIMA model. The comparison of the prediction errors serves as a further demonstration that our proposed SNN can indeed be effectively applied to this field and bring, on top of the advantages that derive from the Neuromorphic technological substrate, advantages in the precision of the predictions. 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 investigate 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

Akopyan, F., Sawada, J., Cassidy, A., Alvarez-Icaza, R., Arthur, J., Merolla, P., Imam, N., Nakamura, Y., Datta, P., Nam, G.-J., Taba, B., Beakes, M., Brezzo, B., Kuang, J. B., Manohar, R., Risk, W. P., Jackson, B., & Modha, D. S. (2015). Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *34*(10), 1537–1557. https://doi.org/10.1109/TCAD.2015.2474396

Bandara, K., Shi, P., Bergmeir, C., Hewamalage, H., Tran, Q., & Seaman, B. (2019). Sales demand forecast in e-commerce using a long short-term memory neural network methodology. In *Lecture notes in computer science* (pp. 462–474). Springer International Publishing. https://doi.org/10.1007/978-3-030-36718-3_39

Box, G. E. P., Reinsel, G. C., Jenkins, G. M., & Ljung, G. M. (2015). *Time series analysis 5e*. John Wiley & Sons.

Brandli, C., Berner, R., Yang, M., Liu, S.-C., & Delbruck, T. (2014). A 240 × 180 130 dB 3 μs latency global shutter spatiotemporal vision sensor. *IEEE Journal of Solid-State Circuits*, *49*(10), 2333–2341.

Broersen, P. M. T. (2006). *Automatic autocorrelation and spectral analysis*. SPRINGER NATURE.

Christensen, D. V., Dittmann, R., Linares-Barranco, B., Sebastian, A., Le Gallo, M., Redaelli, A., Slesazeck, S., Mikolajick, T., Spiga, S., Menzel, S., et al. (2022). 2022 roadmap on neuromorphic computing and engineering. *Neuromorphic Computing and Engineering*, *2*(2), 022501.

Davies, M., Srinivasa, N., Lin, T.-H., Chinya, G., Cao, Y., Choday, S. H., Dimou, G., Joshi, P., Imam, N., Jain, S., Liao, Y., Lin, C.-K., Lines, A., Liu, R., Mathaikutty, D., McCoy, S., Paul, A., Tse, J., Venkataramanan, G., … Wang, H. (2018). Loihi: A neuromorphic

manycore processor with on-chip learning. *IEEE Micro*, *38*(1), 82–99. https://doi.org/10.1109/MM.2018.112130359

Douglas, F., Gover, H., Docherty, C., Shields, G., Leventi, K., & Di Caterina, G. An exploration of the optimal feature-classifier combinations for transradial prosthesis control [Engineering in Medicine and Biology Conference, EMBC ; Conference date: 11-07-2022 Through 15-07-2022]. English. In: Engineering in Medicine and Biology Conference, EMBC ; Conference date: 11-07-2022 Through 15-07-2022. 2022, July. https://embc.embs.org/2022/

Furber, S. B., Galluppi, F., Temple, S., & Plana, L. A. (2014). The spinnaker project. *Proceedings of the IEEE*, *102*(5), 652–665. https://doi.org/10.1109/JPROC.2014.2304638

Gao, K., Luk, W., & Weston, S. (2021). High-frequency trading and financial time-series prediction with spiking neural networks. *Wilmott*, *2021*, 18–33. https://doi.org/10.1002/wilm.10927

Hogenraad, R., McKenzie, D. P., & Martindale, C. (1997). The enemy within: Autocorrelation bias in content analysis of narratives. *Computers and the Humanities*, *30*(6), 433–439.

Hua, Y., Zhao, Z., Li, R., Chen, X., Liu, Z., & Zhang, H. (2019). Deep learning with long short-term memory for time series prediction. *IEEE Communications Magazine*, *57*(6), 114–119. https://doi.org/10.1109/mcom.2019.1800155

Hyndman, R., & Athanasopoulos, G. (2018). *Forecasting: Principles and practice* (2nd). OTexts.

Intel. (2021). Lava: A software framework for neuromorphic computing.

Kim, Y., Chough, J., & Panda, P. (2022). Beyond classification: Directly training spiking neural networks for semantic segmentation. *Neuromorphic Computing and Engineering*, *2*(4), 044015.

Kirkland, P., Caterina, G. D., Soraghan, J., & Matich, G. (2020). SpikeSEG: Spiking segmentation via STDP saliency mapping. *2020 International Joint Conference on Neural Networks (IJCNN)*.

Kirkland, P., Manna, D., Vicente, A., & Caterina, G. D. (2022). Unsupervised spiking instance segmentation on event data using STDP features. *IEEE Transactions on Computers*, *71*(11), 2728–2739.

Lapicque, L. (1907). Recherches quantitatives sur l'excitation electrique des nerfs traitée comme une polarization. *Journal de Physiologie et Pathologie General*, *9*, 620–635.

Lara-Benitez, P., Carranza-Garcia, M., & Riquelme, J. C. (2021). An experimental review on deep learning architectures for time series forecasting. *International Journal of Neural Systems*, *31*(03), 2130001.

Li, B., & Sainath, T. N. (2017). Reducing the computational complexity of two-dimensional lstms. *Interspeech 2017*. https://doi.org/10.21437/interspeech.2017-1164

Liu, C., Tao, Z., Luo, Z., & Liu, C. (2024). Mtsa-snn: A multi-modal time series analysis model based on spiking neural network. https://doi.org/10.48550/ARXIV.2402.05423

Madrid, E. A., & Antonio, N. (2021). Short-term electricity load forecasting with machine learning. *Information*, *12*(2), 50.

Manna, D. L., Vicente-Sola, A., Kirkland, P., Bihl, T., & Di Caterina, G. (2022). Simple and complex spiking neurons: Perspectives and analysis in a simple stdp scenario. *Neuromorphic Computing and Engineering*.

Mateńczuk, K., Kozina, A., Markowska, A., Czerniachowska, K., Kaczmarczyk, K., Golec, P., Hernes, M., Lutosławski, K., Kozierkiewicz, A., Pietranik, M., Rot, A., & Dyvak, M. (2021). Financial time series forecasting: Comparison of traditional and spiking neural networks. *Procedia Computer Science*, *192*, 5023–5029.

Nan, G., Wang, C., Liu, W., & Lombardi, F. (2020). Dc-lstm: Deep compressed lstm with low bit-width and structured matrices. *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. https://doi.org/10.1109/iscas45731.2020.9180869

Neftci, E. O., Augustine, C., Paul, S., & Detorakis, G. (2017). Event-driven random back-propagation: Enabling neuromorphic deep learning machines. *Frontiers in Neuroscience*, *11*.

Orchard, G., Frady, E. P., Rubin, D. B. D., Sanborn, S., Shrestha, S. B., Sommer, F. T., & Davies, M. (2021). Efficient neuromorphic signal processing with loihi 2. *2021 IEEE Workshop on Signal Processing Systems (SiPS)*, 254–259.

Parameshwara, C. M., Li, S., Fermüller, C., Sanket, N. J., Evanusa, M. S., & Aloimonos, Y. (2021). Spikems: Deep spiking neural network for motion segmentation. *2021 IEEE/RSJ*

International Conference on Intelligent Robots and Systems (IROS), 3414–3420.

Reid, D., Hussain, A. J., & Tawfik, H. (2014). Financial time series prediction using spiking neural networks (E. Ben-Jacob, Ed.). *PLoS ONE*, *9*(8), e103656.

Sezer, O. B., Gudelek, M. U., & Ozbayoglu, A. M. (2020). Financial time series forecasting with deep learning: A systematic literature review: 2005-2019. *Applied Soft Computing*, *90*, 106181.

Sharma, V., & Srinivasan, D. (2010). A spiking neural network based on temporal encoding for electricity price time series forecasting in deregulated markets. *The 2010 International Joint Conference on Neural Networks (IJCNN)*, 1–8.

Shrestha, S. B., & Orchard, G. (2018). Slayer: Spike layer error reassignment in time. *Advances in neural information processing systems*, *31*.

Thompson, N. C., Greenewald, K., Lee, K., & Manso, G. F. (2020). The computational limits of deep learning. *arXiv preprint arXiv:2007.05558*, *10*.

Vicente-Sola, A., Manna, D. L., Kirkland, P., Di Caterina, G., & Bihl, T. (2022). Evaluating the temporal understanding of neural networks on event-based action recognition with dvs-gesture-chain. *arXiv*.

Wang, M., Wang, Z., Lu, J., Lin, J., & Wang, Z. (2019). E-lstm: An efficient hardware architecture for long short-term memory. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, *9*(2), 280–291. https://doi.org/10.1109/jetcas.2019.2911739

Wen, Q., Zhou, T., Zhang, C., Chen, W., Ma, Z., Yan, J., & Sun, L. (2023). Transformers in time series: A survey. https://arxiv.org/abs/2202.07125

Xiao, Y., & Gong, P. (2022). Removing spatial autocorrelation in urban scaling analysis. *Cities*, *124*, 103600.

Yarga, S. Y. A., Rouat, J., & Wood, S. (2022). Efficient spike encoding algorithms for neuromorphic speech recognition. *Proceedings of the International Conference on Neuromorphic Systems 2022*, 1–8.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., & Zhang, W. (2021). Informer: Beyond efficient transformer for long sequence time-series forecasting. *The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Conference*, *35*(12), 11106–11115.