Neuromorphic Computing with Address-Event-Representation using Time-to-Event Margin Propagation

Madhuvanthi Srivatsav R, Shantanu Chakrabartty, Chetan Singh Thakur

***Abstract*—Address-Event-Representation (AER) is a spike-routing protocol that allows the scaling of neuromorphic and spiking neural network (SNN) architectures. However, in conventional neuromorphic architectures, the AER protocol and in general, any virtual interconnect plays only a passive role in computation, i.e., only for routing spikes and events. In this paper, we show how causal temporal primitives like delay, triggering, and sorting inherent in the AER protocol itself can be exploited for scalable neuromorphic computing using our proposed technique called Time-to-Event Margin Propagation (TEMP). The proposed TEMP-based AER architecture is fully asynchronous and relies on interconnect delays for memory and computing as opposed to conventional and local multiply-and-accumulate (MAC) operations. We show that the time-based encoding in the TEMP neural network produces a spatio-temporal representation that can encode a large number of discriminatory patterns. As a proof-of-concept, we show that a trained TEMP-based convolutional neural network (CNN) can demonstrate an accuracy greater than 99% on the MNIST dataset and 91.2% on the Fashion MNIST Dataset. Overall, our work is a biologically inspired computing paradigm that brings forth a new dimension of research to the field of neuromorphic computing.**

***Index Terms*—Neuromorphic Computing, Algorithms, Address Event Representation, Spiking Neural Networks**

I. Introduction

Address-Event-Representation (AER) is a widely-used event-based asynchronous protocol used commonly in the design of large-scale and re-configurable neuromorphic hardware [1]–[3].A typical implementation of AER uses packet-based switching and time-division-multiplexing to achieve brain-scale connectivity on 2-dimensional and 2.5-dimensional hardware platforms, which are limited by the number of physical interconnects and routing pathways. In literature, different variants of the AER communication protocol have been proposed to improve channel capacity [2] and system scalability by reducing memory requirements [3, 4]. In most previous implementations, AER and other interconnect mechanisms (virtual and physical) have only played a passive role, i.e., they only transmit signals. Spike-routing latency is viewed as a nuisance or a source of system uncertainty in these architectures. However, recent research has shown that neuronal dendrites, which can be viewed as the neurobiological equivalent of interconnects, exhibit a range of linear and nonlinear mechanisms that allow them to implement elementary computations [5]–[7]. These findings have inspired spiking neural networks (SNNs) architectures using active interconnects or interconnects with computational capabilities [4, 7]–[9]. For instance, in [9], the researchers present a bio-plausible columnar learning network (CLN) that is inspired by the excellent spatio-temporal computing capabilities of the dendrites of the cortical neurons. Also, a recently proposed dendro-centric computing framework [10] extends the concept of active interconnects further and proposes to encode information spatio-temporally in the pulse or spike sequences. These sequences can then be decoded using nano dendrites, and this modality can be used to address specific neurons using the sequences as addresses. From an energetic point of view, this kind of information processing has been proposed to scale linearly with the number of neurons, thus enabling energy-efficient AI applications [10]. Another major argument for incorporating processing-in-interconnects through axonal or dendritic delays is the premise that both spatial and temporal encoding can produce different groups of neurons that fire in specific temporal sequences. Such networks with axonal delays can exhibit an enormous memory capacity as they have more groups than neurons (due to combinatorial factors) and thus could exhibit a huge diversity in network responses [6].

In this paper, we present a spike computing framework called time-to-event margin-propagation (TEMP) that exploits the computational primitives inherent in AER and other spike-routing or interconnect architectures. Margin-Propagation (MP) is an approximate computing technique that has been previously used for designing hardware-efficient neural networks using analog and digital circuits [11]–[15]. However, the MP paradigm has not been applied to time-domain processing where the MP primitives translate to delay, trigger and sorting. This paper is the first attempt in combining these two concepts to develop a novel AER architecture using causal primitives like delay, trigger, and sorting operations, as shown in Fig. 1(a)-(c). These operations can be easily implemented using time-division-multiplexing and packet-switching networks. For instance, the triggering operation illustrated in Fig. 1(b) passes an input spike only if it arrives before a specific time instant denoted by T. Similarly, the sorting operation illustrated in Fig. 1(c) is naturally implemented because of the temporal ordering of spikes. Using TEMP, we show that these fundamental operations can be used to demon-

Madhuvanthi Srivatsav R, and Chetan Singh Thakur are with the NeuRonICS Lab, Department of Electronic Systems Engineering, Indian Institute of Science, Bengaluru 560012, India (e-mail: csthakur@iisc.ac.in). Shantanu Chakrabartty is with the Department of Electrical and Systems Engineering, Washington University in St. Louis, St. Louis, MO 63130 USA.

All correspondence for this work should be addressed to *shantanu@wustl.edu* and *csthakur@iisc.ac.in*

strate non-linear classification abilities producing comparable results to traditional multi-layer neural networks. Furthermore, TEMP models the information in the precise timing of the spikes, with the help of TTFS (Time to First Spike) encoding Fig. 1(d). TTFS coding leads to a significant reduction in network spiking activity, and hence reducing the overall energy consumption. A hyper-parameter in the TEMP formulation controls the network's sparsity, latency, and accuracy, thus ensuring its adaptability to diverse applications. As highlighted in Fig. 1(e), by tuning the hyper-parameter γ, a TEMP network can control the number of output spikes/sparsity of a layer. This formulation can be used to realize a much richer M-of-N spike encoding [8] or K-based encoding strategies [10]. Such rank order encoding strategies are easy to implement on neuromorphic hardware, have higher information content, and promote energy-efficient sparse coding [10], [16]. Coding information in spike order has been proven to be useful in applications such as image recognition and reconstruction. The first spike is generally considered to carry more information, leading to the possibility of faster recognition and restoration with few spikes [8]. Additionally, the asynchronous nature of TEMP allows the network to encode information using temporal dynamics that results in the spatio-temporal encoding of features with potentially enormous memory capacity. This is illustrated in Fig. 1(f), where a TEMP network trained to discriminate digits exploits different spike-timing patterns involving different groups of neurons for images of the same digit.

II. Event-based Model for a TEMP Neuron

At the core of TEMP is margin propagation which is a piece-wise-linear approximate computing technique introduced in [11, 12] and extended in [13, 14]. TEMP extends margin propagation into the time-domain where a TEMP neuron generates a spike/event at time instant t when the following condition is satisfied

$$\sum_j [t - t_j]_+ = \gamma \tag{1}$$

Here t_j denotes the arrival time of the j^{th} pre-synaptic spike/event, $\gamma > 0$ denotes the firing threshold, and $[.]_+$ denotes a ReLU function. Fig. 2(a,b) shows a possible mechanism for implementing equation 1. An internal state variable (for example, a counter or a capacitor) stores the membrane potential, which starts increasing as soon as a pre-synaptic spike/event occurs. However, at the j^{th} event, the state variable or the counter update rate is increased by j. Thus, as more events arrive, the state-variable increases at a faster rate. When the state variable reaches the threshold value γ, say at time instant t_z, the TEMP neuron emits a spike. The ReLU operation in equation 1 is naturally implemented due to time-causality - that is, any spikes that arrive after t_z are ignored during the computation as shown in Fig. 2(a). Also, every neuron is associated with a $time_{out}$ factor at which it will reset its counter or state variable. If the neuron's potential does not reach the threshold or γ before the $time_{out}$, the neuron will no longer spike, and will be reset. Note that while there could be several techniques to implement TEMP on digital, analog, electronic, and non-electronic hardware, this paper focuses on the system architecture and not on specific implementation details.

III. TEMP-based spiking neural network

Like other SNN architectures, two TEMP neurons i and j can be connected to each other using a synaptic weight w_{ij}. However, unlike the conventional SNN formulations, the role of synaptic weights in the TEMP network is to delay the input spikes. Following a differential margin-propagation architecture proposed in [14], [13] to approximate inner-products, a similar mapping is also applied to equation 1. The output of an i^{th} neuron in a TEMP network is two spikes/events denoted by their respective time of occurrence t_i^+ and t_i^-. These occurrences are computed according to the following:

$$\begin{aligned} \sum_j [t_i^+ - (t_j^+ + w_{ij}^+)]_+ + [t_i^+ - (t_j^- + w_{ij}^-)]_+ &= \tau_m \\ \sum_j [t_i^- - (t_j^+ + w_{ij}^-)]_+ + [t_i^- - (t_j^- + w_{ij}^+)]_+ &= \tau_m \end{aligned} \tag{2}$$

Here, the synaptic weights are represented as differential quantities as $w_{ij} = w_{ij}^+ - w_{ij}^-$, with $w_{ij}^+, w_{ij}^- \geq 0$. Both the positive quantities w_{ij}^+, w_{ij}^- are time-delays which ensures that the equation (2) is causal. Similarly t_j^+ and t_j^- represent the pre-synaptic arrival times in differential form. The occurrence times t_i^+ and t_i^- are then processed according to a differential ReLU operator, which is given by

$$(t_i^+, t_i^-) = \begin{cases} (t_i^+, t_i^-) & \text{if } t_i^+ \geq t_i^- \\ (t_i^-, t_i^-) & \text{otherwise} \end{cases} \tag{3}$$

Note that in TEMP-based networks, there is no distinction between spike delays and synaptic modulation, as a result, the formulation allows incorporating phase (or spike timing) information into the overall network dynamics. In fact, there exists an equivalence between TEMP-based networks and Integrate and Fire (IF) spiking neural networks. In Supplementary Appendix I, we show the mathematical equivalence for both leaky and non-leaky IF neurons.

TEMP can exhibit both leaky as well as non-leaky behavior. In the non-leaky model, input is retained until the neuron spikes, unlike the leaky version, where the potential starts to decay after reaching its peak. Fig. 3 demonstrates the leaky and non-leaky dynamics of the TEMP neuron which is subjected to input spike train $\delta(t - t_i)$ of varying frequencies. The top two plots resemble the non-leaky approach for an input pre-synaptic spiking frequency of 50Hz and 20Hz, respectively, whereas the bottom two plots depict the leaky approach for an input pre-synaptic spiking frequency of 10Hz and 5Hz, respectively. The threshold γ is varied for each of the figures in the 4 subsections. The threshold γ decides the potential at which the neuron fires. It can also be observed that if the γ value is higher, the time the membrane potential takes to reach that threshold increases, leading to a decrease in the firing frequency. Also, note that the leakage dynamics are

This article has been accepted for publication in IEEE Journal on Emerging and Selected Topics in Circuits and Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JETCAS.2023.3328916

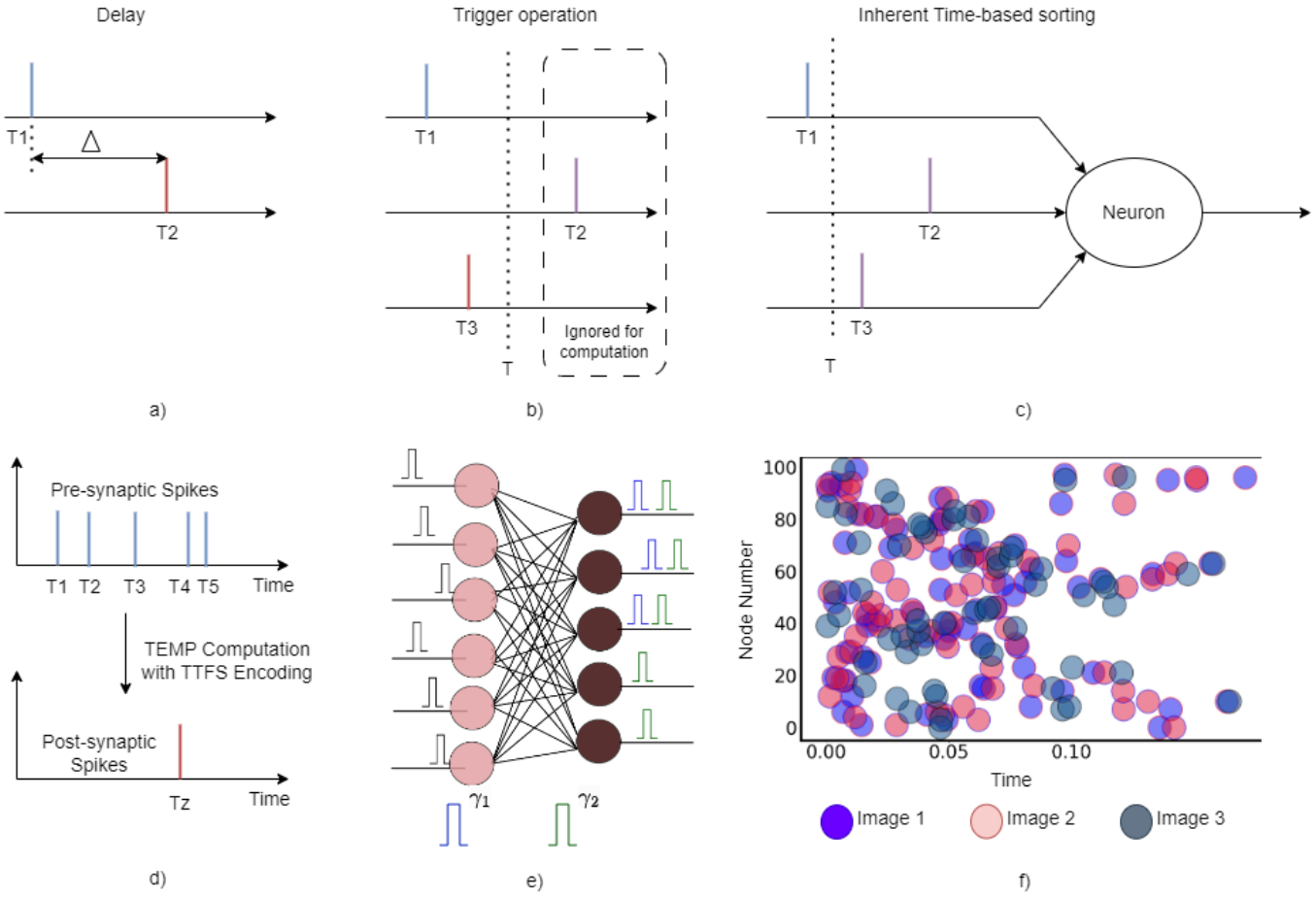


Fig. 1. **Proposed TEMP-based computing paradigm:** Virtual interconnects (AER) can only a) delay pulses (spikes) or b-c) determine causal relationships between pulses (triggering and time-based sorting). d) The formulation of TEMP incorporates the TTFS encoding scheme, where the post-synaptic neuron processes the information from the pre-synaptic spikes and encodes its output in the precise spiking time of one spike. e) The application-specific tunable γ parameter controls the sparsity in encoding information. Here $\gamma_1 > \gamma_2$, and it can be observed that as γ is increased, sparsity is enhanced. f) The time-domain computations in TEMP bring forth spatio-temporal encoding of input patterns. Diverse encoding patterns can be observed for images of the same class in a network trained on the MNIST dataset. This diversity can be attributed to the fact that the information is encoded in the group of neurons that fire and the order in which they fire.

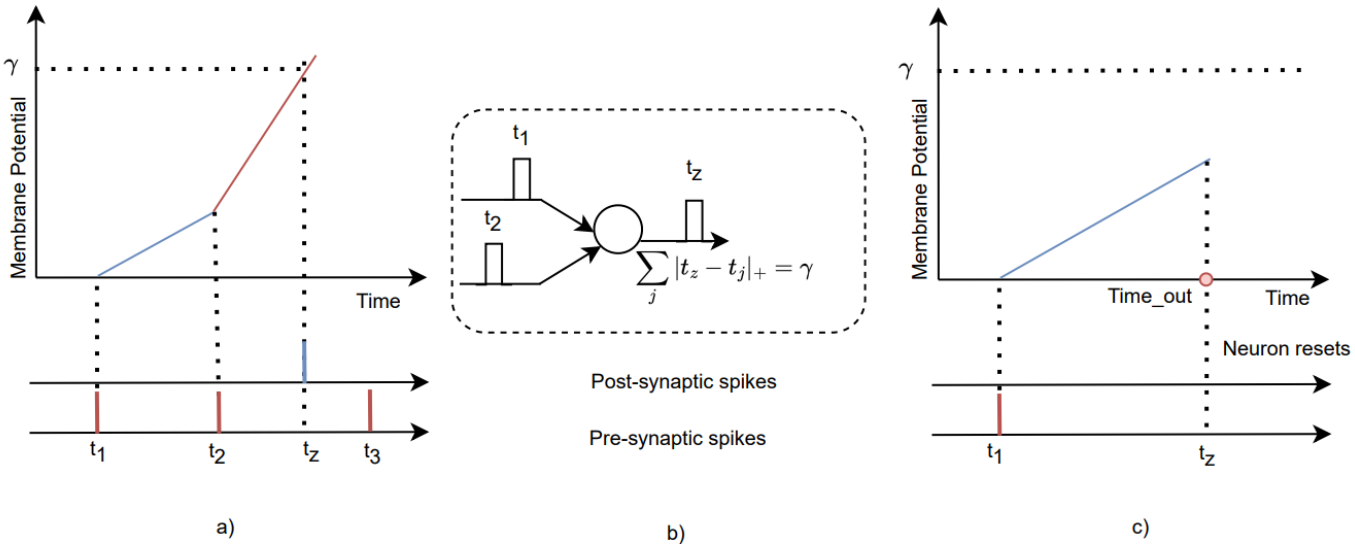


Fig. 2. **Computational model of TEMP-based neurons:** a) As the post-synaptic neuron receives pre-synaptic spikes at time t_1 and t_2, its membrane potential starts rising with increasing slopes determined by the number of pre-synaptic neurons that it encounters. b) a network representation of TEMP based neurons. c) Every TEMP neuron is associated with a time-out, after which it will no longer spike and will reset.

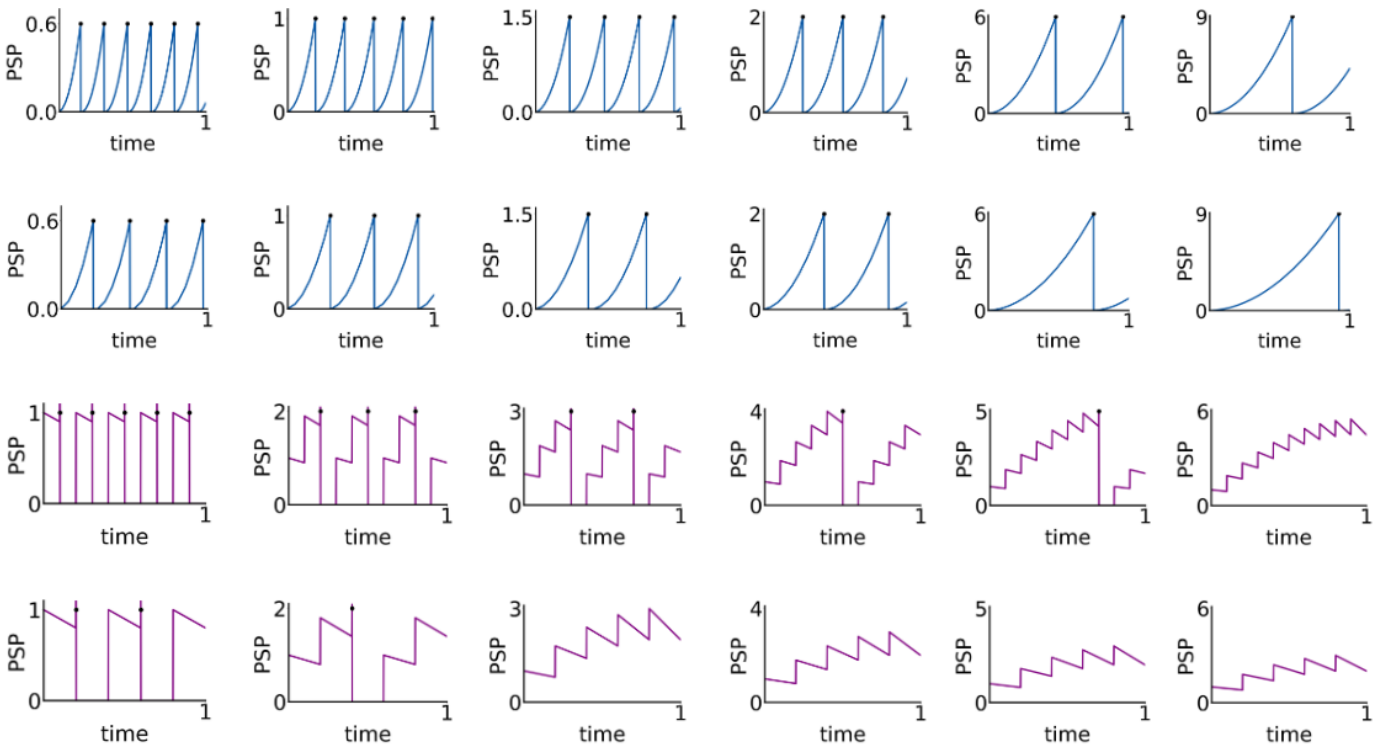


Fig. 3. **Dynamics of a single TEMP neuron** in non-leaky (a,b) and leaky (c,d) mode for a pre-synaptic spike train generated at a frequency of 50 Hz (a), 20 Hz (b), 10 Hz (c) and 5 Hz (d). When the membrane potential reaches the threshold γ, an output spike is generated. The sub-plots show the variability of the output spike rate (output spike generation is marked as black dots) with respect to γ and input spike rate. PSP represents the postsynaptic potential of a neuron.

linear in TEMP-based neurons. This can be attributed to the fact that the derivation of the leaky dynamics in TEMP is based on the piece-wise linear approximation of the exponential function as depicted by Eq 3 in Appendix 1. In this work, we have shown the working of the non-leaky implementation of the TEMP neuron model, in which the spikes that arrive early have a higher contribution. However, similar analysis is possible for the leaky implementation too, where the neurons that fire further in the past will have lesser contribution.

IV. AER Realization of TEMP networks

Here we describe a possible mechanism to implement TEMP networks using the AER protocol. The trained parameters w_{ij} will be assumed to be quantized or can assume only specific values. It has been observed that post-training quantized weights (at a precision equal to or greater than 8 bits) provide the same level of recognition accuracy as a network with full-precision weights. Therefore, for q-bit quantization, each of the weights can assume 2^q possible values. For the proposed implementation, we will instantiate 2^q routing tables of size $N \times M$, where N is the number of post-synaptic/receiver neurons and M is the number of pre-synaptic/sender neurons. This is shown in Fig. 4(a), where an entry in each of the routing tables is a 1 or 0 entry indicating if a sender neuron emits a spike, the event is routed to the destination neuron after a fixed delay. Note that the delay corresponding to each routing table is fixed, and all routing tables share a common output bus/interconnect. The AER protocol is then used by the destination (or post-synaptic) TEMP neurons to receive the event, which then process information according to equation 1. The fixed delay in Fig. 4(a) could be implemented using physical interconnects or using time-outs. Specific implementation details will be a topic for another paper.

V. Experimental Results

To demonstrate the advantages achieved with the proposed TEMP framework when applied to machine learning tasks, a population of TEMP neurons are connected with each other in a feed-forward fashion. The results are based on the implementation of TEMP as given by Eq. 2 in the spiking network using a standard deep learning framework. Spatio-temporal input stimuli are interfaced to the network through a population of neurons which we call sensory neurons. The sensory layer is projected onto the subsequent layer through learnable conduction delays, which enable learning multiple tasks.

A. Non-linear Classification using TEMP Networks

1) XOR Classification task: We begin by validating the proposed TEMP with a classic linearly non-separable XOR task (Illustrated in Fig. 5). This has been done to verify the nonlinear classification capability of the proposed solution. XOR data has been generated from the uniform distribution $\mathcal{U}(-1, 1)$ and encoded as differential spikes. The architecture we have set up has a dense layer with ten TEMP and

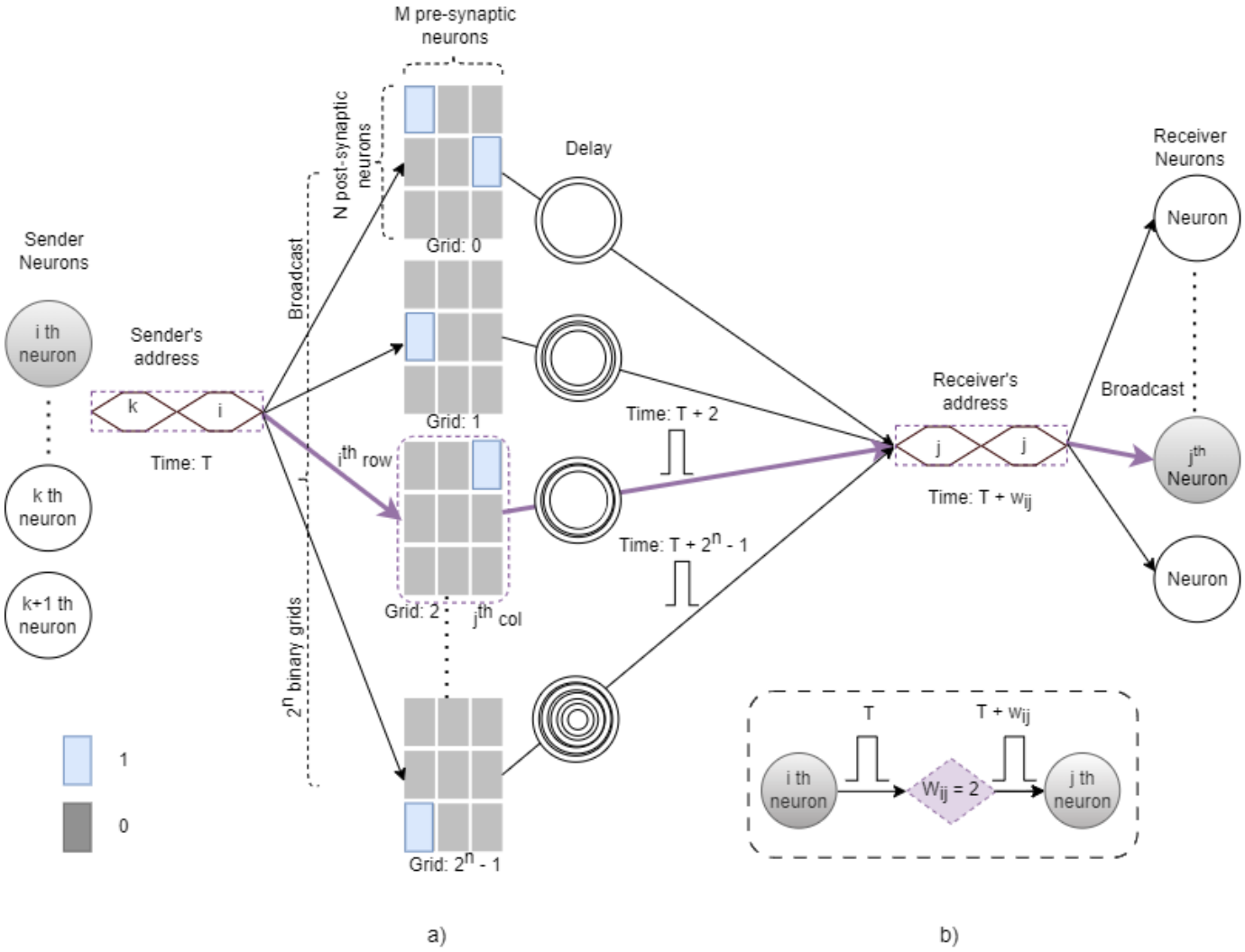


Fig. 4. **Realization of TEMP in AER:** a) As represented in the connectivity representation figure (b), the i^{th} sender neuron with address i, sends out a spike, which needs to be transmitted to the j^{th} receiver neuron with a delay of 2-time units. The sender's spike event is broadcasted to the 2^q (N x M) binary routing tables, which contain the connectivity information. In the second routing table (Grid:2), a 1 is present at the grid (i,j), which indicates a connection between the i^{th}sender neuron and the j^{th} receiver neuron (0 indicates no connection). Accordingly, a spike event with the receiver's address is sent to the digital bus after a delay of 2-time units. b) Connectivity information representing the connectivity between the i^{th} and j^{th} neuron with an interconnect delay of 2-time units.

a classification layer with two TEMP, which is tasked to signal the true class. The network is initialized with values drawn from a normal distribution. Learning loss is defined such that TEMP belonging to true class fires t^- far earlier than t^+. Loss is binary cross entropy loss between softmax activation of the classification layer and true class labels. After training 20 epochs with 60000 samples (batch size of 128) with Adam optimizer and an initial learning rate of 0.001, the network successfully learned the XOR classification task with an accuracy of 99.6%. The good classification accuracy proves that the non-linearity induced by the TEMP enables the classification of non-linearly separable XOR data (The results are presented in Fig. 5).

2) MOON classification task: To further understand the effect of adding layers to classify non-linearly separable data, we have investigated the classification performance of the proposed TEMP solution with a synthetic two-dimensional binary classification dataset popularly known as the MOON dataset (Illustrated in Fig 5). We have implemented the following spiking network for this purpose: $2 \rightarrow 10 \rightarrow 20 \rightarrow 2$. The weights were initialized by drawing from the normal distribution.

Training samples of 20000 were presented to the network as mini-batches of size 128 for 20 epochs. The optimizer was set to Adam with a learning rate of 0.03. It was necessary to include standardization ($\frac{x-\mu}{\sigma}$) as part of the loss function to sustain the propagation of gradients across the network during training. The proposed TEMP demonstrated success in fitting the data with a test accuracy of $99.25 \pm 0.03\%$, thereby emphasizing the capability of the proposed TEMP to estimate optimal nonlinear boundary on test data (The results are presented in Fig. 5).

3) MNIST and Fashion-MNIST Classification tasks: To investigate the credibility of the proposed TEMP network

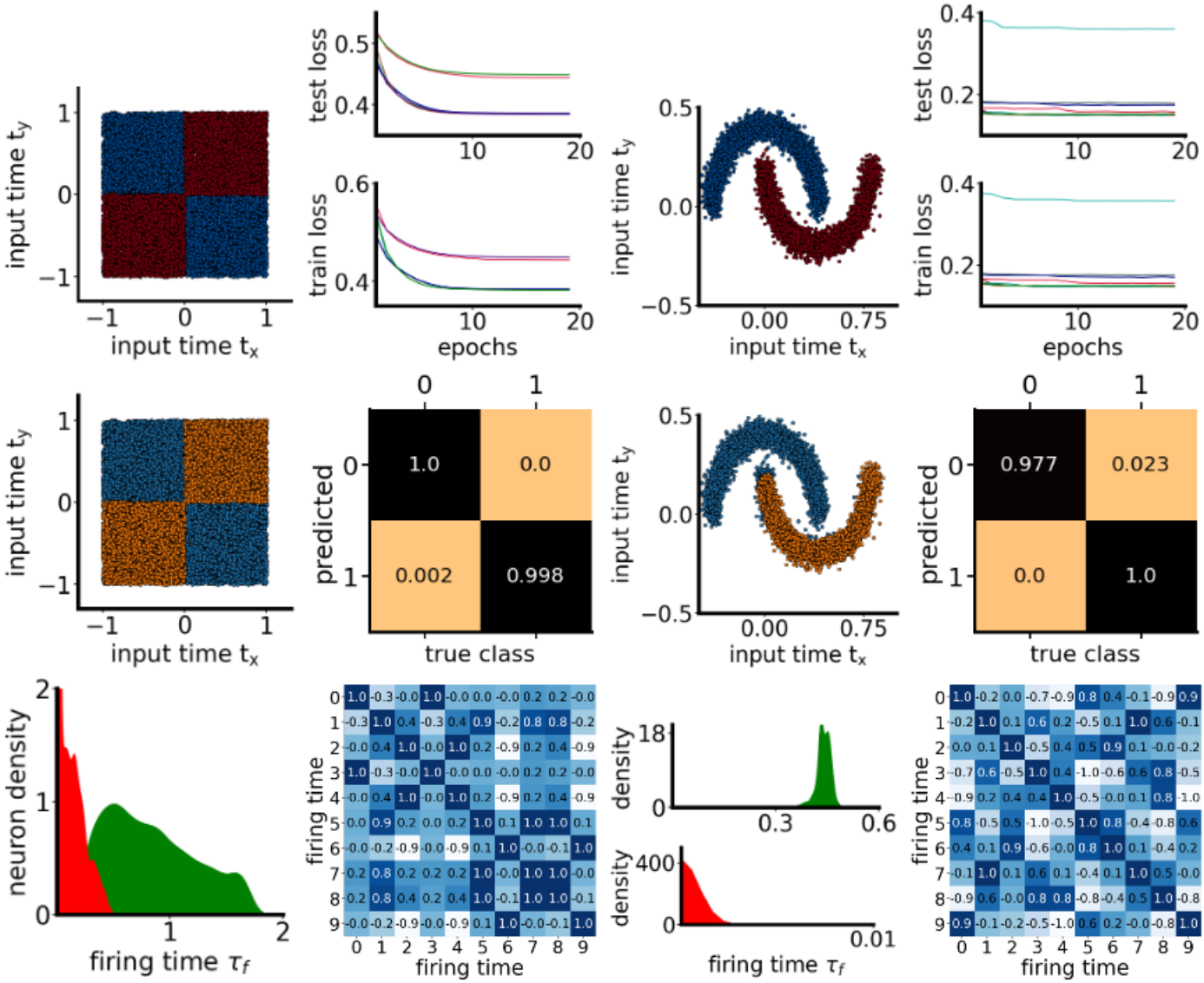


Fig. 5. **Classification of XOR and MOON data set.** Top row: plots 1 and 3 - Illustration of XOR and MOON dataset. The input times t_x and t_y correspond to input spike train time corresponding to the x and y axes. Plot 2 and 4 - training progress in terms of loss on train and validation dataset for different runs (with different initializations) (XOR and MOON). Middle row: plots 1 and 3 - Classification result on XOR and MOON datasets. The color of each sample indicates the class predicted by the proposed solution. Plots 2 and 4 - confusion matrix of XOR and MOON. Bottom row: plots 1 and 3 - spike times of the two different class neurons. Our learning has induced a clear separation between the firing time of the two classes. Plots 2 and 4 - correlation between the firing time of 10 hidden neurons of XOR and MOON dataset estimated across test samples. Very less value in non-diagonal elements indicates nearly zero correlation between the firing time of hidden neurons.

in terms of generalization capability, we applied it to the prevalent MNIST [17] and Fashion MNIST [18] classification tasks. The network architectures used consist of TEMP-based fully-connected layers and convolution layers. Pixel intensities were translated to differential spike trains.

For the architecture implementing a $784 \rightarrow 100 \rightarrow 10$ network, the dense layer was trained in an end-to-end supervised fashion with a batch size of 32 using the ADAM optimizer with an initial learning rate of 0.001 to minimize the standardized categorical entropy loss. Training converges in 30 epochs, reaching a best test accuracy of 97.7% on the MNIST dataset. Fig. 6b gives insight into the distribution of delay learned by the synapses connecting 100 hidden nodes and 10 class nodes for MNIST data. It brings out the class-specific distribution of delay.

Further, we implemented a TEMP-based CNN network with two convolutions consisting of 16 and 32 channels and two fully-connected layers with 500 and 10 nodes. Each convolution layer was implemented using 3x3 kernels and was followed by a max pooling layer. Further, batch normalization was introduced between every successive layer for faster convergence. The network was trained with mini-batches of size 16 for 30 epochs with Adam optimizer. We were able to achieve 99.1% accuracy on the MNIST Dataset and 91.27% accuracy on the Fashion MNIST dataset. A time-based learning rate decay scheduler was used for all the training runs. A brief description of the network architectures describing the type of layer and its corresponding gamma value is presented

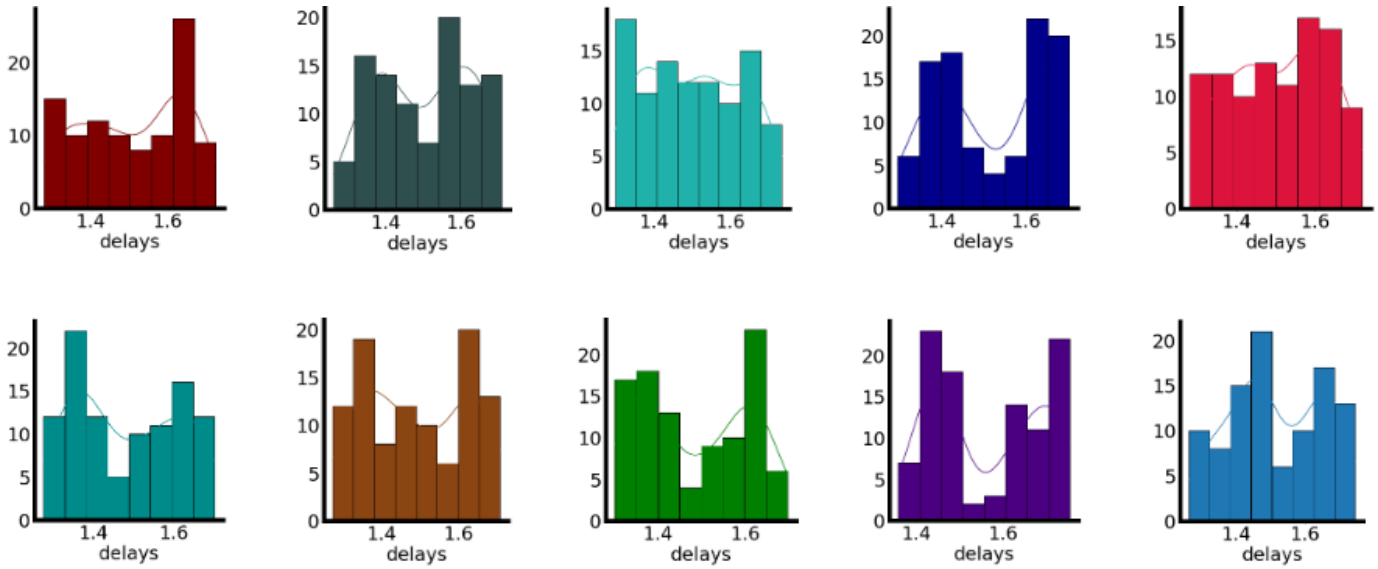


Fig. 6. **Class-specific learning of trainable delays**: The histogram plot of the synaptic weights/delay (after learning) between hidden neurons to the 10 class nodes of the network trained on the MNIST dataset. The difference in the delay distributions depicts the class-specific learning of delays.

in Table I.

Comparison with state-of-the-art: There exists an extensive body of literature in designing and training SNNs that encode information in the relative timing of individual neuron spikes [19]–[24]. These works focus on training multi-layered LIF and n-LIF neuron based SNN models using back-propagation and STDP techniques, and emphasize on the evident advantages of temporal coding schemes in terms of sparsity, energy efficiency, and the ease of implementation on neuromorphic hardware. For comparison with TEMP, we consider the results obtained with other spiking architectures which follow temporal and rate-based encoding. Table II displays MNIST classification accuracy at par with that of state-of-the-art spiking architectures. Additionally, a recently published work on columnar learning networks [9], which incorporates dendritic dynamics and promotes spatio-temporal data processing, achieved an accuracy of 95% on the MNIST dataset. Further, [25] follows a rate-based coding approach and achieves a classification accuracy of 94.5 % using a 6-layer CNN architecture on the Fashion-MNIST dataset, while [26] achieves an accuracy of 86.5 % using a 2 convolution layer SNN with TTFS encoding. Note that rate-based approaches display better accuracy but with sacrifice in latency, sparsity, and energy efficiency when compared to TTFS-based approaches.

TABLE I
NETWORK ARCHITECTURES FOR DIFFERENT DATASETS

Dataset	Network Architecture
XOR	Input_layer (2) - FC_MP (15,1) - FC_MP (2,0.5)
MOON	Input_layer (2) - FC_MP (10,1) - FC_MP (20,0.5) - FC_MP (2,1)
MNIST	Input_layer (784) - FC_MP (100,22) - FC_MP (10,15)
Fashion MNIST and MNIST CNN model	Input_layer (28×28×1) - Conv_MP (3×3×16,8) - MaxPool - BatchNorm - Conv_MP (3×3×32,8) - MaxPool - BatchNorm - FC_MP (500,11) - BatchNorm - FC_MP (10,50)

Key: Input layer - Input_layer (no of nodes), Fully connected, dense TEMP layer - FC_MP (no of nodes, gamma), TEMP convolutional layer - Conv_MP (kernel size x kernel size x no of channels, gamma), Batchnorm - batch normalization layer, MaxPool - Max-pooling layer

TABLE II
COMPARISON OF PROPOSED TEMP WITH EXISTING SPIKING ARCHITECTURES ON MNIST CLASSIFICATION TASK. TEMP[1] AND TEMP[2] ARE SPIKING NETWORKS WITH FULLY-CONNECTED AND CONVOLUTIONAL LAYERS, RESPECTIVELY.

Method	Accuracy	Method	Accuracy
Rate coding			
[27]	0.91	[28]	0.975
[29]	0.984	[30]	0.95
TTFS coding			
[19]	0.97	[20]	0.99
[21]	0.99	[22]	0.984
[23]	0.979	[24]	0.97
TEMP[1]	**0.977**	TEMP[2]	**0.991**

Key: [27] - LIF neurons trained using STDP
[28] -N-LIF rate encoding using DFA/SDFA
[29] - N-LIF trained using Backpropagation
[30] - LIF trained using unsupervised Contrastive Divergence rule
[19] - N-LIF rate encoding using Backpropagation
[20] - LIF trained using DNN-SNN conversion
[21] - N-LIF using backpropagation
[22] - N-LIF trained using STDP
[23] - Spike Response model trained using backpropagation
[24] - LIF trained using backpropagation

B. Effects of Weight Quantization

The weights of the TEMP-based MLP and CNN networks trained on the MNIST dataset were subject to different levels of quantization and the results are reported in Fig. 7. It can be observed that up to 4 bits of quantization, there is no significant drop in the test accuracy for the MLP and CNN architectures. This property can be exploited in reducing the routing table overheard in the proposed AER architecture.

This article has been accepted for publication in IEEE Journal on Emerging and Selected Topics in Circuits and Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JETCAS.2023.3328916

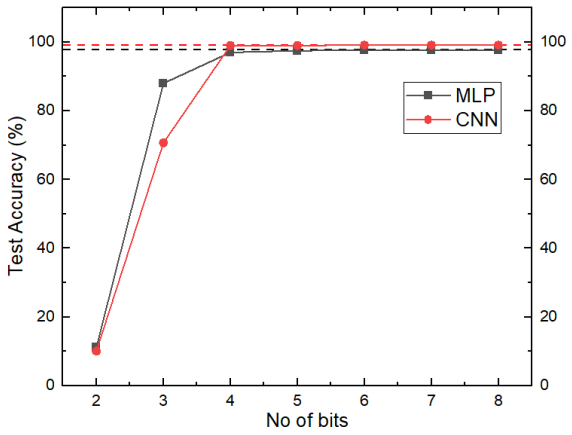


Fig. 7. Results of post-training quantization on the TEMP based MLP and CNN networks trained on the MNIST dataset. The dashed lines represent the floating-point baseline accuracies.

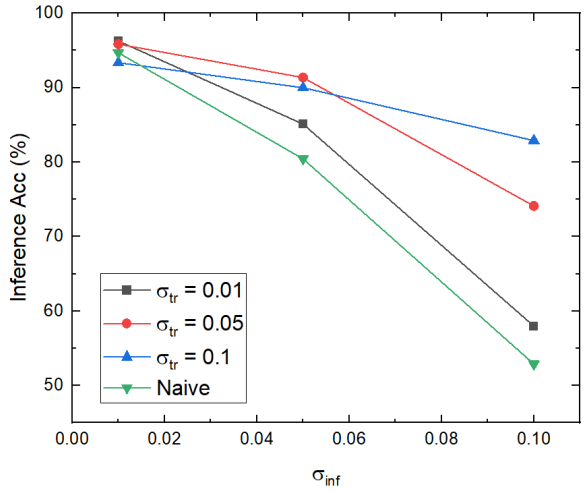


Fig. 8. Effect of noise on inference Accuracy of TEMP based networks: When a naively trained network is subject to a random 10% packet loss and Gaussian noise of varying standard deviations σ_{inf}, the inference accuracy of the network drops (As shown by the **Naive** plot). Test accuracy on MNIST obtained for the networks trained with dropout layer and noise with different standard deviations (σ_{tr}) as a function of the noise injected during inference characterized by σ_{inf} is plotted. On performing variation aware training to compensate for packet loss and stochastic delay, the robustness of the network can be enhanced. VAT is performed on top of the trained network with a dropout layer (of probability = 0.1) and Gaussian noise of standard deviation varying between 0.01,0.05 and 0.1. The average inference accuracy over 10 inference runs is reported.

C. Robustness Analysis

In this paper, we have proposed an AER network, implemented using the TEMP formulation, where the interconnect delays play a major role in computation. However, in the current communication networks, the interconnect delay is not deterministic. The performance degradation in networks can be mainly attributed to packet loss [31] and delay variability or jitter which is usually modeled using Gaussian, exponential and Weibull probability density functions [32]–[34].

In an attempt to understand the effect of the stochastic delay and packet loss on the inference accuracy of TEMP-based networks, we consider the 3-layer MLP architecture and add a Gaussian noise and dropout layer between the hidden and output layers. The dropout layer is added to model random packet loss, and the noise layer is added to model stochastic delay in the arrival time of the spikes that are transmitted to the output layer.

Assuming 10 % of the spikes are dropped randomly, and the standard deviation (σ_{inf}) of the Gaussian noise varies between 0.01, 0.05 and 0.1, the drop in the inference accuracy of a naively trained network can be observed in Fig. 8. However, this drop in inference accuracy can be compensated by a variation aware training (VAT) technique, where the network is re-trained with the dropout and the noise layer of varying σ_{tr}, thus making it more robust to packet losses, and stochastic delays.

D. Analysis of Spatio-Temporal encoding in TEMP Networks

By virtue of axonal delays, TEMP-based networks exhibit rich spatio-temporal encoding, which is well-known for their enhanced combinatorial representational capabilities. As the network becomes structured with learning, certain stimuli-specific patterns emerge as portrayed in Fig. 9, validating the combinatorial representational capability of these encoding patterns.

As noticed in Fig. 9a, though there exists a similarity between patterns that belong to the same class, patterns that emerged in response to different stimuli do exhibit variance.

Fig. 9b shows the spike-raster plot (Time of firing vs. Neuron Number) of a dense layer. This experiment is an indication of the fact that there will be no ambiguity even if a set of neurons is shared across multiple spatio-temporal patterns. This is owing to the fact that a single neuron can fire with different patterns at different times, and these patterns are not only defined by their constituent neurons but also by their precise firing time.

E. The Effect of the Hyper-parameter γ on Computation

Sparsity in causal spikes: The majority of energy is consumed by signaling, which could be regulated by sparse coding, where M out of total N neurons ($M \leq N$) are active. Sparse coding is an energy-saving neural coding along the lines of neural signal transmission theory and energy utilization rate theory.

The formulation of TEMP neuron includes a hyper-parameter named γ, which regulates the number of post-synaptic neurons that fires (Fig. 10a). Thus, TEMP exhibits an adaptive sparse coding strategy, which enables it to generate sparse M of N spike codes. The proposed TEMP has adaptive delay plasticity, thus introducing a scheme where the order of arrival of pre-synaptic spikes at a post-synaptic neuron is governed by their information content. In this study, we highlight the effect of delay plasticity and γ on the processing ability of TEMP (Fig. 10 b,c).

Fig. 10d displays the distribution of pre-synaptic spikes required to fire each of the hidden layer neurons. It could be verified that, an average neuron fires with 16% of total pre-synaptic spikes, thereby validating that the spikes representing critical information reach the neuron early and thus achieve the desired result with as few spikes as possible.

This article has been accepted for publication in IEEE Journal on Emerging and Selected Topics in Circuits and Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JETCAS.2023.3328916

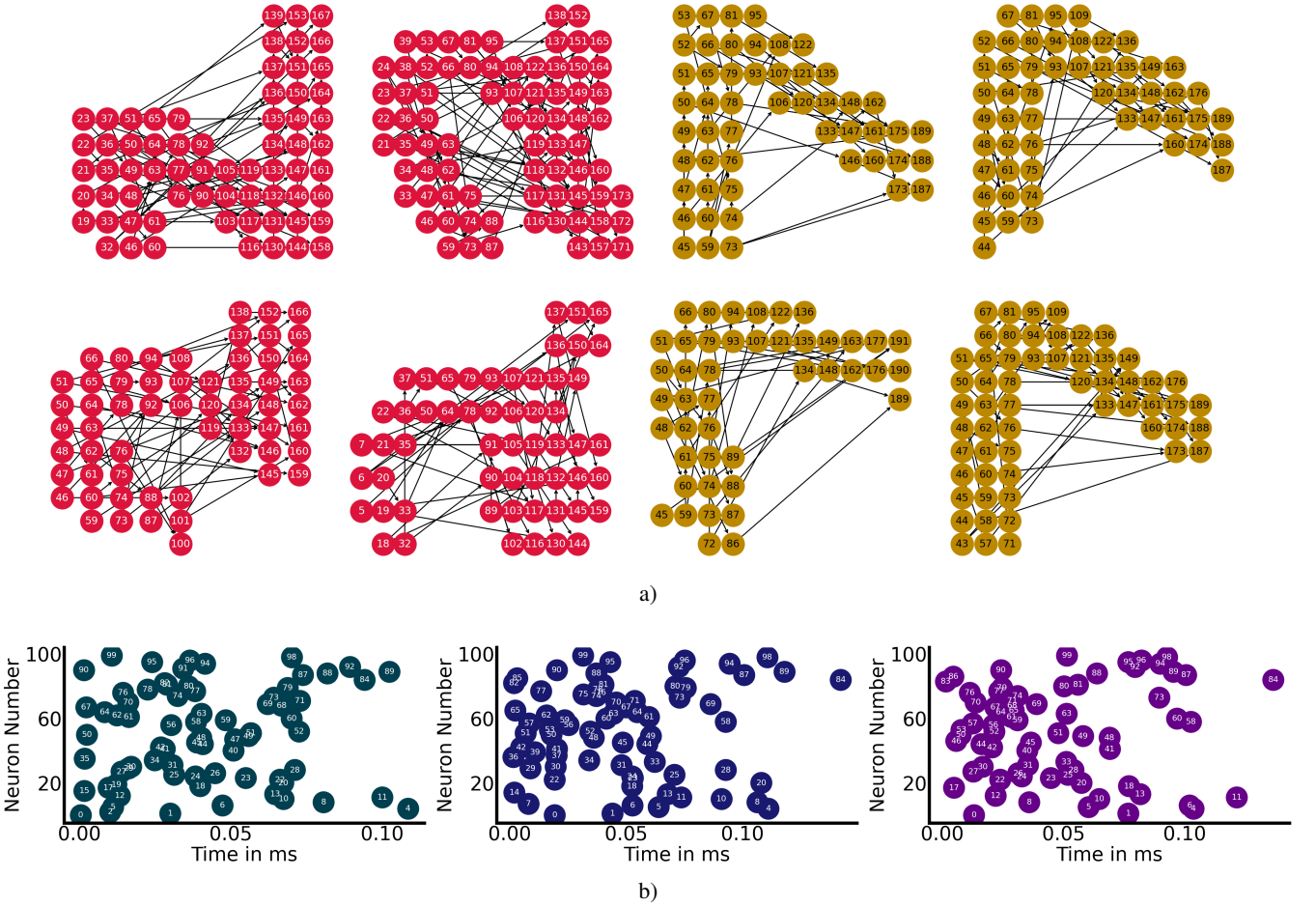


Fig. 9. **Combinatorial representational capability of spatio-temporal patterns.** (a) Spatio-temporal patterns emerged from the TEMP-based convolution layer in response to samples from input stimuli representing classes 2 and 7. This shows the intra-class similarity and inter-class variability learned by the patterns. Despite the intra-class similarity, unique spatio-temporal firing patterns can be observed for each stimulus, thus bringing out the combinatorial representational capability of TEMP. (b) Spike raster plot of the learned spatio-temporal patterns. It could be seen that patterns share neurons, but the neuronal firing order differs across the patterns. This adds to the combinatorial representational capability of TEMP-based networks.

F. Tunable latency (TT scaling)

The hyperparameter γ plays a notable role in tuning the tradeoff between latency and accuracy. We could achieve a notable reduction in latency (γ) with a slight degradation in performance as provided in Fig. 10(f). With increase in latency (γ), the recognition capability of TEMP improves. However, as γ is increased further, we notice the drop in accuracy, owing to the fact that the non-linearity exhibited by the TEMP neuron is causality-induced (γ dependent) non-linearity. An increase in γ is limited by the fact that the causality should induce sufficient non-linearity to generate separable patterns. However, there is an optimal value of γ, where we can obtain both competitive accuracies as well as reasonable latency.

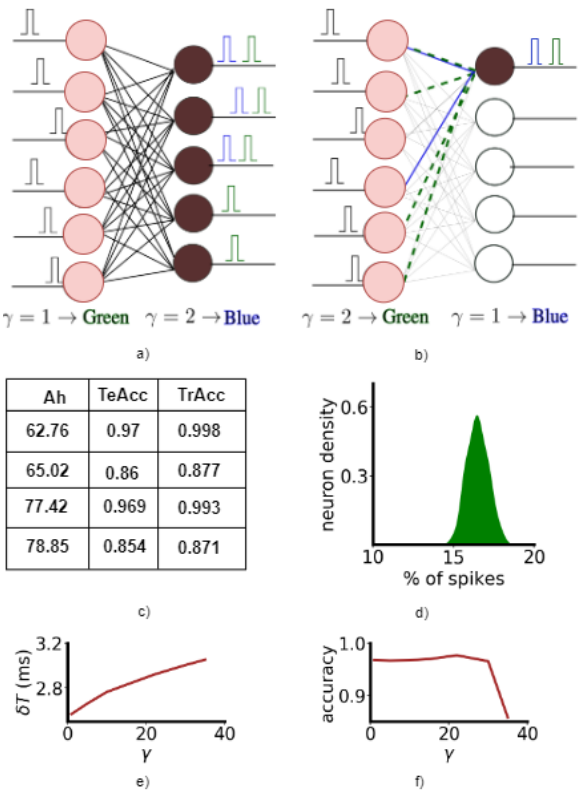
VI. Discussion & Conclusion

We have proposed a time-based spike computing paradigm TEMP. This builds on a principle known as Margin Propagation (MP), which has been introduced to approximate log-sum-exp as a piecewise linear function. We remark that TEMP involves only primitive operations such as time-based addition, subtraction, threshold operation, etc.

TEMP is built on delay plasticity, which contributes towards a unique implementation of the popular AER protocol. By modeling synaptic strength as interconnect delay, TEMP reduces the demands on neuromorphic hardware by completely eliminating synaptic circuits, thus favoring the construction of highly reconfigurable large-scale neuromorphic spiking architectures.

The property of learnable delay, inherent to TEMP, has led to the emergence of spatio-temporal patterns in the network. The combinatorial representational capability of these patterns has been demonstrated for different classes of stimuli. Further, through experiments, the benefits of the tunable hyperparameter γ inherent to TEMP have been demonstrated. γ can be used to control the latency, accuracy, and sparsity in TEMP-based networks. This application-specific tunable property of γ enables widespread application of TEMP from ultra-fast differential sensing systems to highly accurate visual recognition systems.

Many novel approaches for training SNNs encoding information temporally have been explored extensively. Some of the gradient-based learning methods to train LIF or n-LIF based SNNs are presented in [19, 23, 24]. By incorporating



Ah	TeAcc	TrAcc
62.76	0.97	0.998
65.02	0.86	0.877
77.42	0.969	0.993
78.85	0.854	0.871

Fig. 10. Effect of hyperparameter γ on computation (a,b,c) Sparse coding: (a) The hyperparameter γ determines the number of nodes that fire, which is a crucial factor in achieving desirable sparsity. (b) Based on γ, the causal set of pre-synaptic spikes varies, which determines the number of pre-synaptic spikes involved in the firing of the post-synaptic neuron. The delay plasticity ensures that most informative pre-synaptic neurons get to deliver their spikes early to post-synaptic neurons. This results in minimal computation. (c) Sparse coding vs. accuracy analysis. A_h is the percentage of active nodes, which is controlled by the hyperparameter γ. TrAcc and TeAcc are training and test accuracy at various sparsity coefficients. (d) This gives the plot of the distribution of the number of causal pre-synaptic spikes required by the TEMP neurons to fire. The total number of pre-synaptic spikes was 784. It could be seen that, on average, the hidden layer TEMP neurons require only $\sim 16\%$ of total pre-synaptic spikes to get triggered. This proves that delay plasticity results in the early arrival of most informative spikes, hence reducing the computations to as minimal as possible. (e,f) Tunable latency vs. accuracy: Analysis of latency and accuracy of TEMP network trained over different values of $\gamma : [1 \ldots 35]$. As γ increases, latency increases as expected. However, classification accuracy reaches its maximum for a particular value of γ as the latter is the critical factor in tuning the non-linearity induced by the TEMP. Thus, γ acts as a hyperparameter for the latency vs. accuracy tradeoff.

temporal coding into differentiable non-leaky TEMP formulation, we showed that we were able to define a continuously differentiable expression between input and output spike times. This enabled exact error backpropagation through a network of neurons. The derivations of the gradients is very similar to the equations derived for the margin propagation (MP) networks as presented in [14].

We showed that the network constructed with the proposed spike computing model could solve non-linear classification tasks with accuracy comparable to that of the state-of-the-art. For training TEMP-based networks, we used TensorFlow libraries and ran the models on Nvidia A100 GPU. We observed the training process to be highly memory intensive. As portrayed in Eq. 2, the large memory requirements of TEMP-based networks can be attributed to the presence of differential weights and inputs, and the inherent computation necessary to find the post synaptic spiking times (t_i^+ and t_i^- in Eq. 2). These variables must also be stored for gradient calculation. With these implementation intricacies, the GPU's utilization reaches 100% even for a 3-layer network with 1000 hidden nodes for the MNIST dataset trained with a batch size of 128. For CNN networks trained on the MNIST and Fashion-MNIST datasets we had to use a batch size of 16 to circumvent this issue. Thus, we could not experiment well on large and complex datasets due to these hardware resource constraints, and we are currently working towards developing sustainable training solutions to overcome these challenges, such as directly training in the spike domain using custom libraries and designing custom accelerators for training.

Data Availability

The code for implementing the TEMP inference engine is available at https://github.com/NeuRonICS-Lab/temp-framework.

Acknowledgment

The authors would like to acknowledge Lakshmi Annamalai (PhD Student, NeuRonICS Lab, IISc) and the joint IISc-WashU Memorandum of Understanding, to facilitate the collaboration between the two institutions. At IISc, this work was supported in part by the SPARC Funds (SP/MHRD-18-0006), INAE (SP/INAE-22-2106), Pratiksha Trust and BCD Funds (FG/SMCH-22-2106).

References

[1] K. A. Boahen, "Point-to-point connectivity between neuromorphic chips using address events," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 47, no. 5, pp. 416–434, 2000.
[2] S. A. Bamford, A. F. Murray, and D. J. Willshaw, "Large developing receptive fields using a distributed and locally reprogrammable address–event receiver," *IEEE transactions on neural networks*, vol. 21, no. 2, pp. 286–304, 2010.
[3] N. Rathi, I. Chakraborty, A. Kosta, A. Sengupta, A. Ankit, P. Panda, and K. Roy, "Exploring neuromorphic computing based on spiking neural networks: Algorithms to hardware," *ACM Computing Surveys*, 2022.
[4] P. Jongkil, Y. Theodore, J. Siddharth, and et. al., "Hierarchical address event routing for reconfigurable large-scale neuromorphic systems," *IEEE transactions on neural networks and learning systems*, 2017.
[5] L. Michael and H. Michael, "Dendritic computation," *Annual Review of Neuroscience*, vol. 28, no. 1, pp. 503–532, 2005.
[6] E. M. Izhikevich, "Polychronization: computation with spikes," *Neural computation*, vol. 18, no. 2, pp. 245–282, 2006.
[7] P. Sun, L. Zhu, and D. Botteldooren, "Axonal delay as a short-term memory for feed forward deep spiking neural networks," pp. 8932–8936, 2022.
[8] F. Galluppi and S. Furber, "Representing and decoding rank order codes using polychronization in a network of spiking neurons," pp. 943–950, 2011.
[9] S. Yoo, Y. Park, Z. Wang, Y. Wu, S. Medepalli, W. Thio, and W. D. Lu, "Columnar learning networks for multisensory spatiotemporal learning," *Advanced Intelligent Systems*, vol. 4, no. 11, p. 2200179, 2022.
[10] K. Boahen, "Dendrocentric learning for synthetic intelligence," *Nature*, vol. 612, no. 7938, pp. 43–50, 2022.
[11] S. Chakrabartty and G. Cauwenberghs, "Margin propogation and forward decoding in analog vlsi," *Proc. IEEE Int. Symp. Circuits and Systems.*, 2004.

[12] G. Ming and C. Shantanu, "Synthesis of bias-scalable cmos analog computational circuits using margin propagation," *IEEE Transactions on Circuits and Systems.*, 2012.

[13] A. R. Nair, C. S, and C. S. Thakur, "In-filter computing for designing ultra light acoustic pattern recognizers," *IEEE internet of Things Journal.*

[14] A. R. Nair, P. K. Nath, C. S, and C. S. Thakur, "Multiplierless mp-kernel machine for energy efficient edge devices," *IEEE transactions on very large scale integration systems.*, 2022.

[15] P. Kumar, A. Nandi, S. Chakrabartty, and C. S. Thakur, "Process, bias, and temperature scalable cmos analog computing circuits for machine learning," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 70, no. 1, pp. 128–141, 2023.

[16] S. B. Furber, G. Brown, J. Bose, J. M. Cumpstey, P. Marshall, and J. L. Shapiro, "Sparse distributed memory using rank-order neural codes," *IEEE Transactions on Neural Networks*, vol. 18, no. 3, pp. 648–659, 2007.

[17] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

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

[19] H. Mostafa, "Supervised learning based on temporal coding in spiking neural networks," *IEEE transactions on neural networks and learning systems*, pp. 3227–3235, 2017.

[20] L. Zhang, Z. S, T. Zhi, and a. et, "Tdsnn: From deep neural networks to deep spike neural networks with temporal-coding," *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 1319–1326, 2019.

[21] Zhou, Shibo, and a. et, "Temporal-coded deep spiking neural network with easy training and robust performance," *Proc. AAAI Conf. Artif. Intell.*, 2021.

[22] S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, and a. et, "Stdp-based spiking deep convolutional neural networks for object recognition," *Neural Networks*, 2018.

[23] I. M. Comsa and a. et, "Temporal coding in spiking neu- ral networks with alpha synaptic function," *International Conference on Acoustics, Speech and Signal Processing*, 2020.

[24] Goltz, Julian, and a. et, "Fast and energy-efficient neuromorphic deep learning with first-spike times," *Nature machine intelligence*, pp. 823–835, 2021.

[25] S. Haghiri, A. Zahedi, A. Naderi, and A. Ahmadi, "Multiplierless implementation of noisy izhikevich neuron with low-cost digital design," *IEEE Transactions on Biomedical Circuits and Systems*, 2018.

[26] M. Yu, T. Xiang, P. Srivatsa, K. T. N. Chu, B. Amornpaisannon, Y. Tavva, V. P. K. Miriyala, and T. E. Carlson, "A ttfs-based energy and utilization efficient neuromorphic cnn accelerator," *Frontiers in Neuroscience*, vol. 17, 2023.

[27] C. Lee, G. Srinivasan, P. Panda, and a. et, "Deep spiking convolutional neural network trained with unsupervised spike-timing-dependent plasticity," *IEEE Transactions on Cognitive and Developmental Systems*, pp. 384–394, 2018.

[28] S. Bang, D. Lew, S. Choi, and J. Park, "An energy-efficient snn processor design based on sparse direct feedback and spike prediction," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–8.

[29] J. Wu, Y. Chua, M. Zhang, and a. et, "Deep spiking neural network with spike count based learning rule," *International Joint Conference on Neural Networks (IJCNN)*, 2019.

[30] E. Stromatias and a. et, "Scalable energy-efficient, low- latency implementations of trained spiking deep belief networks on spinnaker," *International Joint Con- ference on Neural Networks*, 2015.

[31] Á. C. Rumín and E. I. C. Guy, "Establishing how many voip calls a wireless lan can support without performance degradation," in *Proceedings of the 2nd ACM international workshop on Wireless multimedia networking and performance modeling*, 2006, pp. 61–66.

[32] J.-S. Kim, J. Lee, E. Serpedin, and K. Qaraqe, "A robust estimation scheme for clock phase offsets in wireless sensor networks in the presence of non-gaussian random delays," *Signal Processing*, vol. 89, no. 6, pp. 1155–1161, 2009.

[33] G. Hooghiemstra and P. Van Mieghem, "Delay distributions on fixed internet paths," *Delft University of Technology, report*, vol. 20011020, 2001.

[34] N. Ou, T. Farahmand, A. Kuo, S. Tabatabaei, and A. Ivanov, "Jitter models for the design and test of gbps-speed serial interconnects," *IEEE Design & Test of Computers*, vol. 21, no. 4, pp. 302–313, 2004.

VII. Biography Section

Madhuvanthi Srivatsav R :

Madhuvanthi Srivatsav R (Student Member, IEEE) received the B.Tech (Hons) and M.Tech in VLSI and Electronics Systems Design from the Indian Institute of Information Technology, Design, and Manufacturing, Kancheepuram, in 2021. She is currently pursuing a Ph.D. degree with the Indian Institute of Science (IISc), Bengaluru, India. She is associated with the NeuRonICS Laboratory, Department of Electronic Systems Engineering, Indian Institute of Science. Her research interests include Machine Learning, Neuromorphic computing, Analog and Digital IC design, and hardware implementation of AI.

Shantanu Chakrabartty :

Shantanu Chakrabartty (Senior Member, IEEE) received the B.Tech. degree from the Indian Institute of Technology Delhi, India, in 1996, and the M.S. and Ph.D. degrees in electrical engineering from Johns Hopkins University, Baltimore, MD, USA, in 2002 and 2004, respectively. From 1996 to 1999, he was at Qualcomm Inc., San Diego, CA, USA. From 2004 to 2015, he was an Associate Professor at the Department of Electrical and Computer Engineering, Michigan State University (MSU), East Lansing, MI, USA. He is currently a Clifford W. Murphy Professor and the Vice-Dean for Research and Graduate Education with the McKelvey School of Engineering, Washington University in St. Louis, St. Louis, MO, USA. He was a Catalyst Foundation Fellow from 1999 to 2004. He is a fellow of the American Institute of Medical. He was a recipient of the National Science Foundation's CAREER Award, the University Teacher-Scholar Award from MSU, and the 2012 Technology of the Year Award from MSU Technologies. He has previously served as an Associate Editor for the IEEE Transactions of Biomedical Circuits and Systems.

Chetan Singh Thakur :

Chetan Singh Thakur (Senior Member, IEEE) received the Ph.D. degree in neuromorphic engineering from the MARCS Research Institute, Western Sydney University, in 2016, and the M.Tech degree from the Indian Institute of Technology, Bombay, India, in 2007. He was a Post-Doctoral Researcher with Johns Hopkins University, Baltimore, MD, USA. He also worked as a Senior Integrated Circuit Design Engineer at Texas Instruments Singapore for few years. In 2017, he joined the Indian Institute of Science, Bangalore, and he is now working as an Associate Professor. His research interests include the computing principles of the brain and apply those to build novel intelligent VLSI systems. He received several awards, such as the Pratiksha Trust Young Investigator Award, the Abdul Kalam Innovation Award from the Indian National Academy of Engineering (INAE) and the Inspire Faculty Award for brain-inspired computing from the Department of Science and Technology (DST).