

This paper has been accepted for the 34<sup>th</sup> International Conference on Artificial Neural Networks (ICANN 2025).

# A Unified Platform to Evaluate STDP Learning Rule and Synapse Model using Pattern Recognition in a Spiking Neural Network

Jaskirat Singh Maskeen<sup>1</sup>[0009–0001–6781–8152] and  
Sandip Lashkare<sup>2</sup>[0000–0003–2018–1681]

Indian Institute of Technology Gandhinagar, India  
{23110146, sandip.lashkare}@iiitgn.ac.in

**Abstract.** We develop a unified platform to evaluate Ideal, Linear, and Non-linear  $\text{Pr}_{0.7}\text{Ca}_{0.3}\text{MnO}_3$  memristor-based synapse models, each getting progressively closer to hardware realism, alongside four STDP learning rules in a two-layer SNN with LIF neurons and adaptive thresholds for five-class MNIST classification. On MNIST with small train set and large test set, our two-layer SNN with ideal, 25-state, and 12-state non-linear memristor synapses achieves 92.73 %, 91.07 %, and 80 % accuracy, respectively, while converging faster and using fewer parameters than comparable ANN/CNN baselines.

**Keywords:** Neuromorphic Computing · Spiking Neural Networks · Spike-Timing-Dependent-Plasticity · Pattern Recognition · MNIST Classification · Synapse Models

## 1 Introduction

Spiking neural networks (SNNs) promise brain-inspired efficiency by co-locating memory and computation, overcoming the memory-bandwidth bottleneck inherent to Von-Neumann processors [6,16]. Neuromorphic hardware such as memristor crossbars implements synaptic updates locally and in parallel, yielding orders of magnitude energy and speed improvements over CPUs/GPUs [16]. Memristors like Resistive Random-Access Memory (RRAM) have positioned them as leading on-chip synapse candidates due to high density, nonvolatility, and precise conductance control [18].

In this work, we target pattern recognition under tight data constraints. Unlike SNN works using surrogate-gradient backpropagation [22,20,19], which require global error signals and centralized weight updates (complicating deployment on neuromorphic hardware), we adopt a supervised Hebbian rule derived from local Spike-Timing-Dependent-Plasticity (STDP) [11] for entirely lo-

cal synaptic plasticity. Our custom Python SNN simulator<sup>1</sup> supports arbitrary finite-state synapse models and Leaky Integrate-and-Fire (LIF) neurons with adaptive thresholds for spike-frequency adaptation as observed in biology [17]. We use the MNIST dataset [15] to evaluate how hardware-realistic synapse models and local STDP rules affect accuracy and efficiency. While Convolutional Neural Networks (CNNs) can surpass 99% on MNIST [15], they need extensive labeled data and significant computational resources for backpropagation. In contrast, SNNs leverage sparse, event-driven spikes and unsupervised STDP to learn features with far fewer samples and compute resources [4].

Several recent studies achieve over 97% on MNIST by using more complex STDP variants in deeper SNNs, for example, Stabilized Supervised STDP (S2-STDP) in multi-layer convolutional SNNs and reward-modulated STDP (R-STDP) in deep spiking networks [7,8,23]. However, those approaches rely on multi-layer architectures and do not evaluate how a minimal two-layer SNN would perform. Moreover, the analysis of the impact of hardware-realistic synapse quantization or finite-state nonlinearity on classification accuracy, along with different STDP models, is limited.

Our work bridges this gap by presenting a unified platform to evaluate the SNN performance by comparing four STDP variants: **Conventional** [2], **Cosine-shaped**, **Sinusoidal-shaped**, **Gaussian-shaped**, paired with three synapse models: **Ideal** (theoretical upper bound for accuracy), **Linear** (with finite states), and **Non-linear** (with finite states) based on  $\text{Pr}_{0.7}\text{Ca}_{0.3}\text{MnO}_3$  data [14], in a simple two-layer network. We use only local STDP (no backpropagation), which allows us to systematically quantify accuracy, convergence speed, and hardware-relevant trade-offs under tight data constraints on the MNIST dataset. We also track synaptic weight changes to identify emerging patterns. By focusing on a minimal architecture and realistic synapse behavior, our study provides a reference to the neuromorphic community for deploying on-chip SNNs where deep architectures or backpropagation are impractical.

## 2 Methodology

### 2.1 Network Architecture

We use a two-layer SNN, with no hidden layers. Layer 1 has 784 neurons, and Layer 2 has 60 neurons (with Non-linear Synapse) or 80 neurons (with Ideal Synapse), for five-class classification, and 80 neurons for ten-class classification. Excitatory synapses connect each input neuron to every output neuron, using models described in subsequent sections. All neurons in the output layer are connected amongst themselves using lateral inhibitory synapses, and utilise the *Winner-Takes-All* rule [9]. So, after a training image has been passed through the network for a fixed duration, the output neuron that fires the most is assigned the label of the training image. So, at the end of training, each output neuron corresponds to a particular label.

<sup>1</sup> <https://github.com/jsmaskeen/nervos>

## 2.2 Neuron Model

We use a modified Leaky Integrate-and-Fire (LIF) model, which incorporates a biologically inspired mechanism of adaptive thresholding, which prevents rapid firing of neurons by increasing the threshold after a spike, which decays back to the initial threshold potential with some time constant  $\tau_0$  [13,5].

It is a simple resistor-capacitor circuit that models the dynamics of neuronal membrane potential. The neuron fires a spike once the membrane potential crosses a threshold ( $V_{th}$ ). After each spike, the membrane potential is reset to  $V_{reset}$ , while the threshold adapts dynamically based on recent neuronal activity. This adaptive process momentarily elevates the threshold to inhibit rapid consecutive firing, then gradually decays to its baseline value with a time constant  $\tau_0$ .

**Membrane Dynamics:** Equation 1 describes the dynamics of our neuron's membrane [13].

$$\frac{dV}{dt} = \frac{1}{C_m}(-g(V - E_L) + I) \quad (1)$$

where  $V$  is membrane potential,  $C_m$  is capacitance of membrane,  $g$  is conductance,  $E_L$  is resting potential and  $I$  is the input current to the neuron. Values used:  $C_m = 8 \text{ pF}$ ,  $g = 0.8 \text{ nS}$ ,  $E_L = -70 \text{ mV}$ ,  $V_{reset} = -90 \text{ mV}$ .

**Threshold Adaptation:** Equation 2 describes the adaptation of threshold voltage  $V_{th}$  [5].

$$\frac{dV_{th}}{dt} = -\frac{V_{th} - V_{th_0}}{\tau_0} \quad (2)$$

where  $V_{th}(t)$  is threshold voltage at any time  $t$ ,  $V_{th_0}$  is the baseline threshold voltage, and  $\tau_0$  is the threshold decay time constant. Values used:  $V_{th_0} = -55 \text{ mV}$ ,  $\tau_0 = 15 \text{ ms}$ .

## 2.3 Synapse Model

Every input and output neuron is connected by a synapse (which is modelled by RRAM in hardware). It represents the strength of the connection between the two neurons. We normalise the synapse weight (conductance of RRAM) so that it is between  $10^{-3}$  (minimum synapse weight) and 1 (maximum synapse weight). When we apply Long Term Potentiation (LTP) pulses, the synapse weight increases, and when we apply Long Term Depression (LTD) pulses, the synapse weight decreases.

1. **Ideal Synapse:** This synapse has an infinite number of states, and there is no non-linearity involved. Subsequent LTD (LTP) pulses uniformly decrease (increase) the synapse weight (Fig. 1, initial weight is set to 1).
2. **Linear Synapse:** This is similar to an Ideal Synapse but is closer to a real RRAM. This synapse offers uniform weight change with LTD or LTP pulses, but there are a finite number of states (Fig. 2, initial weight is set to 1 and the number of states is  $n = 23$ ).

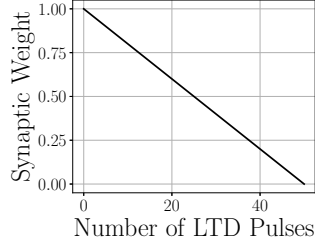


Fig. 1: Ideal synapse: weight vs. LTD pulse count.

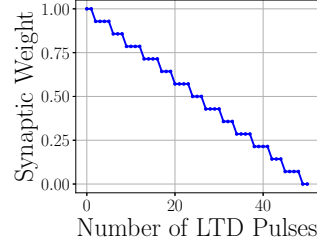


Fig. 2: Linear synapse: weight vs. LTD pulse count.

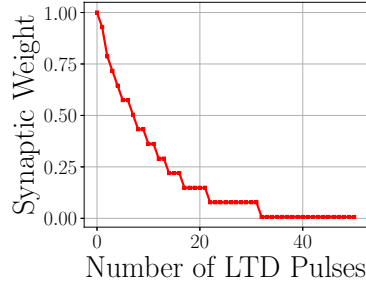
3. **Non-linear Synapse:** This synapse model accurately represents the real RRAM device ( $\text{Pr}_{0.7}\text{Ca}_{0.3}\text{MnO}_3$  memristor), as subsequent pulses give a nonlinear weight change [14]. We normalise the experimental RRAM conductance values (Table 1), and fit an exponential function, described by equation 3 [12], which gives  $w_i$ , the weight of  $i^{\text{th}}$  state for a  $n$  state synapse.

$$w_i = w_{\max} - \frac{w_{\max} - w_{\min}}{1 - e^{-\nu}} \left[ 1 - \exp \left( -\nu \left( 1 - \frac{i}{n} \right) \right) \right] \quad (3)$$

$\nu$  is used to control the shape of the curve, and  $3.5 \leq \nu \leq 3.7$ , models our experimental data most accurately. Fig. 3 shows the weight change of this synapse.

Table 1: Experimental data from RRAM [14].

State	Conductance $G$ ( $\mu\text{S}$ )
1	316.228
2	199.526
3	125.893
4	63.096
5	25.119
6	12.589
7	5.754
8	3.981

Fig. 3: Non-linear synapse: weight vs. LTD pulse count. Initial weight is 1 and number of states are  $n = 23$ .

Initially, all synapses are assigned the maximum weight of 1. This allows the network to learn more easily, as all the connections are initially strong. If the weights are assigned randomly, then certain neurons might never have a good connection to facilitate learning [3].

## 2.4 Learning Rule

The learning rule specifies how much the weight should change between a pair of pre- and post-synaptic spikes. We denote the time when a pre-synaptic neuron fires by,  $t_{pre}$  and when a post-synaptic neuron fires by,  $t_{post}$ . We define  $\Delta t$  as  $t_{post} - t_{pre}$ . The STDP function takes  $\Delta t$  as input and returns the weight change for the synapse between a corresponding pair of neurons. The learning rule is given by the equation 4 [11].

$$\Delta w = \begin{cases} \eta \cdot F(\Delta t) \cdot (w - w_{\min})^\gamma, & \text{if } F(\Delta t) < 0 \\ \eta \cdot F(\Delta t) \cdot (w_{\max} - w)^\gamma, & \text{if } F(\Delta t) > 0 \end{cases} \quad (4)$$

Where,  $\eta$  is the learning rate (we take  $\eta \in [0.03, 0.13]$ ),  $F(\Delta t) < 0$  indicates depression,  $F(\Delta t) > 0$  indicates potentiation and  $\gamma$  is a constant (we take  $\gamma = 0.9$ ). The maximum synapse weight is  $w_{\max} = 1$ , and the minimum synapse weight is  $w_{\min} = 10^{-3}$ . We describe four models of STDP ( $F(\Delta t)$ ) [1,2,21], as follows:

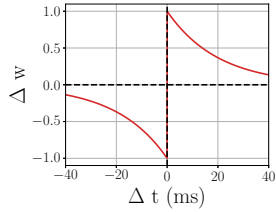


Fig. 4: Conventional STDP curve (Eq. 5).

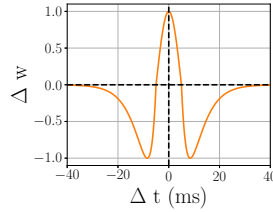


Fig. 5: Cos STDP curve (Eq. 6).

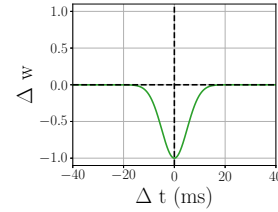


Fig. 6: nGauss STDP curve (Eq. 7).

### Conventional STDP:

$$F_{\text{conventional}}(\Delta t) = \begin{cases} A_{\text{up}} \cdot e^{-\frac{\Delta t}{\tau_{\text{up}}}}, & \text{if } \Delta t > 0 \text{ (pre before post)} \\ A_{\text{down}} \cdot e^{\frac{\Delta t}{\tau_{\text{down}}}}, & \text{if } \Delta t < 0 \text{ (post before pre)} \end{cases} \quad (5)$$

Where  $A_{\text{up}} = 0.8$ ,  $A_{\text{down}} = -0.3$ ,  $\tau_{\text{up}} = 5$ ,  $\tau_{\text{down}} = 5$ . Fig. 4 shows the potentiation and depression curve for the Conventional STDP. The lesser the time difference between post- and pre-neuron, the higher the weight increases. which indicates causation, the pre-neuron *causes* the post-neuron to fire. Similarly, in the other case, if a post-neuron fires before the pre-neuron, then the synapse weight decreases.

### Cos STDP:

$$F_{\text{cos}}(\Delta t) = \begin{cases} A_{\text{in}} \cos\left(\frac{\pi \Delta t}{2\tau_0}\right), & |\Delta t| \leq \tau_0, \\ -A_{\text{out}} \left( e^{-\alpha_1(\Delta t - \tau_0)} - e^{-\alpha_2(\Delta t - \tau_0)} \right), & \text{Otherwise,} \end{cases} \quad (6)$$

Where  $\tau_0 = 1.5$ ,  $A_{\text{in}} = 1$ ,  $A_{\text{out}} = 4$ ,  $\alpha_1 = 0.2$ ,  $\alpha_2 = 0.4$ . Fig. 5 shows the potentiation and depression curve for the Cos STDP, It is symmetric, and if the time delay between the firing of two neurons is between  $-\tau_0$  to  $\tau_0$ , the synapse weight increases, else it decreases.

**nGauss STDP:**

$$F_{\text{nGauss}}(\Delta t) = -A \exp\left(-\frac{\Delta t^2}{2\sigma^2}\right), \quad (7)$$

Where  $A = 1$ ,  $\sigma = 5$ . Fig. 6 consists only of the depression region. There is no potentiation. Hence, this rule can be used to unlearn the weights.

**Sin STDP:**

$$F_{\text{sin}}(\Delta t) = \begin{cases} -A_{\text{out}}(e^{\alpha_1 \Delta t} - e^{\alpha_2 \Delta t}), & \Delta t < 0, \\ A_{\text{in}} \sin\left(\frac{\pi \Delta t}{2\tau_0}\right), & 0 \leq \Delta t \leq 2\tau_0, \\ -A_{\text{out}}(e^{-\alpha_1(\Delta t - 2\tau_0)} - e^{-\alpha_2(\Delta t - 2\tau_0)}), & \Delta t > 2\tau_0, \end{cases} \quad (8)$$

Where  $\tau_0 = 5$ ,  $A_{\text{in}} = 1$ ,  $A_{\text{out}} = 4$ ,  $\alpha_1 = 0.2$ ,  $\alpha_2 = 0.4$ . Fig. 7 shows the potentiation and depression curve for the Sin STDP, If the pre-neuron fires after the post-neuron, or the post-neuron fires more than  $2\tau_0$  after the pre-neuron, then the synapse weight decreases. Otherwise, it increases.

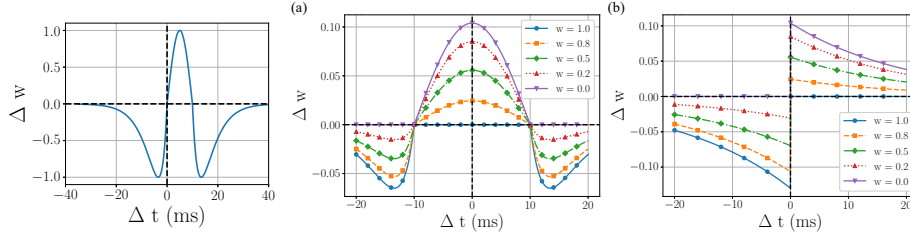


Fig. 7: Sin STDP curve Fig. 8: (a) LTD-LTP curve for Cos STDP, (b) LTD-LTP curve for Sin STDP. (Eq. 8).

Fig. 8 shows the comparison between the weight change of a synapse that follows the learning rule given by equation 4, and the two STDPs, Cos STDP (equation 6) and Conventional STDP (equation 5). We assume five initial weights of the synapse and see how the weights change. In both the figures, Fig. 8(a) and Fig. 8(b), the maximum weight synapse, can only decrease and the minimum weight synapse can only increase.

## 2.5 MNIST Dataset and Input Encoding

The MNIST dataset consists of images of handwritten digits [15] used to evaluate image classification models. Each image is a  $28 \times 28$  grayscale matrix with pixel

intensities from 0 (black) to 255 (white). We flatten each image to a 784-vector and normalize it to the range  $[0, 1]$ . So, each entry  $p \in [0, 1]$  is then converted to a firing frequency  $f$  by:

$$f = p \times (f_{\max} - f_{\min}) + f_{\min}, \quad f_{\max} = 70, \quad f_{\min} = 5 \quad (9)$$

Brighter pixels result in higher spiking frequencies. The encoded frequencies generate spike trains over a fixed duration, *training\_duration*, which is set to 100 by default. So this produces a binary spike-train matrix  $M \in \{0, 1\}^{784 \times 100}$ , where  $M_{ij} = 1$  if  $i^{th}$  neuron fired at the  $j^{th}$  millisecond, and  $M[i][j] = 0$  otherwise.

## 2.6 Evaluation

For five-class classification, our two-layer SNN (784 input  $\rightarrow$  80 output neurons) achieves an accuracy of 92.7% when trained on 100 images, and tested on 1500 unseen images. As illustrative baselines, we train the following models on the same data:

1. **ANN:** Input layer (784 units)  $\rightarrow$  Dense(128 units, ReLU activation)  $\rightarrow$  Dense(5 units, softmax activation).
2. **CNN:** Input  $28 \times 28 \times 1 \rightarrow$  Conv2D(32 filters,  $3 \times 3$  kernel window, ReLU activation)  $\rightarrow$  MaxPooling2D( $2 \times 2$ )  $\rightarrow$  Conv2D(64 filters,  $3 \times 3$  kernel window, ReLU activation)  $\rightarrow$  MaxPooling2D( $2 \times 2$ )  $\rightarrow$  Flatten  $\rightarrow$  Dense(128 units, ReLU activation)  $\rightarrow$  Dense(5 units, softmax activation).

For both models, training is done using the Adam optimizer, with a 0.001 learning rate, a categorical cross-entropy loss function, a 64 batch size, and 10 epochs.

## 3 Results

### 3.1 Changing STDP Rule

We evaluate our SNN on MNIST with ideal synapses using Conventional, Cos, Sin, and nGauss STDP across five-, seven-, and ten-class tasks. The best-performing rule is selected based on accuracy. For five classes we use 100 training images (200 for seven and ten). As nGauss alone always reduces synapse weights, we apply it briefly (with small parameters) to a random 10% of training images each epoch to promote unlearning, then switch back to the primary rule. Initial weights are 1 for Conventional and Sin STDP, but random for Cos STDP to allow any weight change otherwise all-1 weights would stay fixed unless spike time differences are huge.

Table 2 shows the accuracies achieved using the respective STDP rules. So, Cos STDP can be ruled out, as it performs poorly. The visualization of synapse weights for Cos STDP (Fig. 9(a)) shows a ghosting effect of expected digits, indicating unlearning. Since for Cos STDP, the initial synapse weights are assigned randomly, for  $\Delta t > 0$ , most causal neuron synapse weights increase. However,

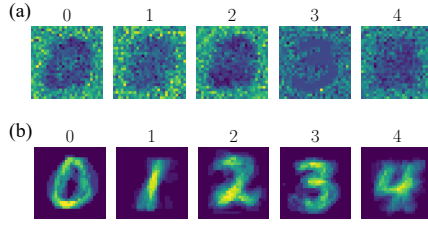


Fig. 9: Synapse weights visualised for (a) Cos STDP and (b) Sin STDP

Table 2: Accuracies (%) observed in the classification of 1500 test images.

Classes	Conventional STDP (%)	Cos STDP (%)	Sin STDP (%)
5	92.73	25.72	86.40
7	87.32	17.60	80.44
10	76.27	13.45	69.86

when  $\Delta t < 0$  there is a high chance that the synapse weight increases again, leading to higher weights for synapses of neurons at the edges (Fig. 9(a)). Sin STDP performs better than Cos but not as well as Conventional (Fig. 9(b) and Fig. 12(a)). It suggests that a larger weight for neurons with  $\Delta t > 0$  leads to better accuracy. It should be noted that if  $\tau_0$  is smaller, then more instances occur where synapse weights decrease for  $\Delta t > 0$ , so the accuracy drops.

### 3.2 Changing Synapse Model and Trade-offs

The best STDP rule comes out to be **Conventional STDP**, so we proceed with it and evaluate on the synapse models. Table 3 gives the architecture and average accuracy obtained by classifying on the MNIST dataset. The number of

Table 3: MNIST classification using Conventional STDP for various synapses

Number of Classes	Ideal Synapse		Linear Synapse		Non-linear Synapse	
	Output Neurons	Accuracy (%)	Output Neurons	Accuracy (%)	Output Neurons	Accuracy (%)
5	80	92.73	80	91.67	60	91.07
6	160	88.93	140	87.88	140	84.73
7	160	87.32	140	85.58	140	83.98
8	160	87.03	140	86.58	140	82.40
10	160	76.27	160	74.46	120	71.20

input neurons is 784, and the number of test images for computing accuracy is 1500. The learning rate varies between 0.03 to 0.13, as we go from Ideal Synapse to a Non-linear Synapse (with 25 states). It can be clearly seen in all cases, the accuracy drops from moving from Ideal to Linear to Non-linear Synapse. The weight evolution for five-class classification can be seen in Fig. 10, the final synapse weights are grouped together into six bins, based on their final weights, and for two epochs, the average weight of each bin is recorded. We see that for the Ideal Synapse (Fig. 10(a)), the weight decrease for each bin is uniform, whereas for the Non-linear Synapse (with 25 states), the synapse reach their final weight in less time (Fig. 10(b)). Both in the case of Linear and Non-linear Synapse, varying the number of states from 25 to 5 decreases the accuracy, for a particular classification task, as seen in Fig. 11(a). However, with an increase



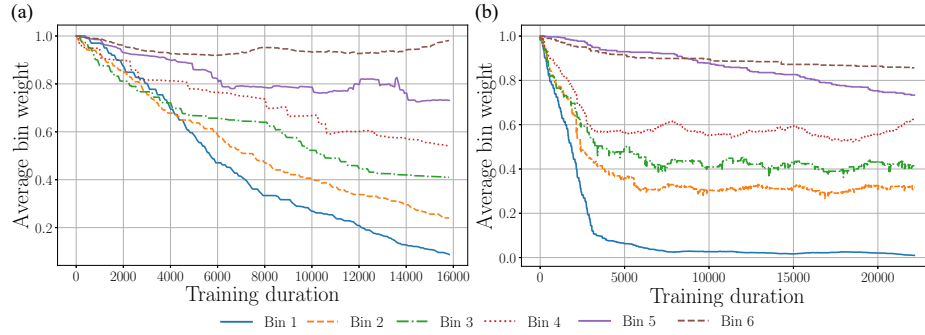


Fig. 10: Weight evolution for (a) an Ideal Synapse and (b) Non-linear Synapse based SNN for classifying digits 0 to 4.

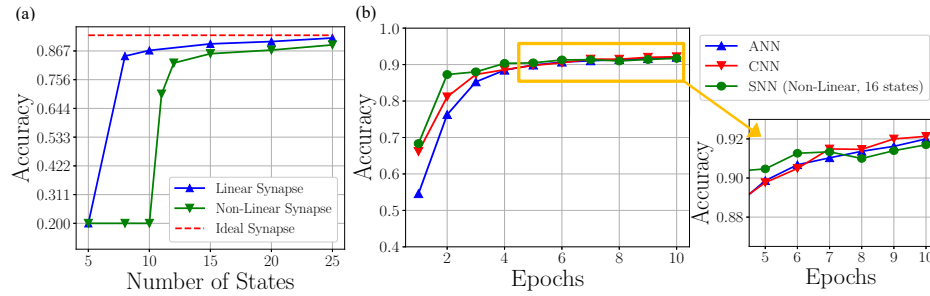


Fig. 11: (a) Accuracy vs. Number of States, for five-class MNIST, (b) Comparison of SNN with ANN and CNN.

in the number of states, the Non-linear Synapse reaches the accuracy levels of the Linear Synapse.

The measured time of training the model on AMD Ryzen 7840HS on five-class MNIST with Conventional STDP and Non-linear Synapse for one epoch, is  $3 \text{ min} \pm 5\text{s}$  (averaged over 10 epochs) and the inference of a single sample, comes out to be  $6.553 \pm 0.0058 \text{ ms}$  (averaged over 7500 inferences).

### 3.3 Comparison with ANN and CNN

We train an ANN and a CNN to classify digits from 0 to 4, and report the accuracy as epochs increase. We compare these two with an SNN with Non-linear Synapse with 16 states respectively (Fig. 11(b)). Note that we keep the training and testing images the same for all models. So, both CNN and ANN are expected to perform poorly as there are only 100 training images and 1500 testing images. However, with more training images, CNN can achieve an accuracy of 99% [15], but the network architecture will not be simple. All accuracy numbers reported throughout the paper were averaged over 5 independent runs (with standard deviation  $< 1\%$ ).

## 4 Discussion

SNN reaches a good accuracy in a small number of epochs, with a small amount of training data, and with a simple architecture (Fig. 12(c) for overview of the methodology). From Fig. 12(b) it is seen that for five-class MNIST, our model usually confuses two with three, which produce very similar spike-train pattern under our rate encoding. Fig. 12(a) shows that the learned weights for classes 2 and 3 significantly overlap, which explains the high misclassification between these two classes. Unlike conventional ANNs/CNNs that process dense,

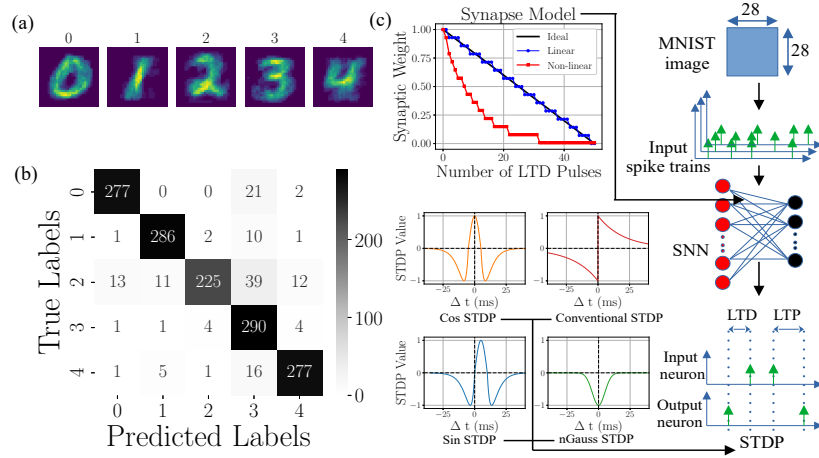


Fig. 12: (a) Weights learned by our network, (b) Confusion matrix for our SNN on five-class MNIST, (c) Overall workflow of our methodology.

synchronous activations, our STDP based SNN leverages sparse, event-driven spikes [4]. This allows it to converge in fewer epochs on small datasets and, when mapped to memristive crossbars, consume less energy per image than a GPU-trained CNN [16]. In practice, SNNs excel on always-on, low-power platforms, for example, edge Internet of Things devices with Dynamic Vision Sensor (DVS) cameras, real-time sensory processing in robotics, or ultra-efficient neuromorphic chips where conventional nets (with their high multiply-accumulate counts) are impractical or wasteful [16,10]. Although we focused on the canonical MNIST benchmark, future work will extend our STDP-synapse framework to harder datasets (Fashion-MNIST, CIFAR-10) and to temporal DVS streams and the deployment will be studied on hardware memristor crossbar array to get an actual measurement of power/memory used.

## 5 Conclusion

Our study shows that neuromorphic computing with SNNs is effective for pattern recognition. Among the four STDP variants examined, Conventional STDP

performed best on five-, seven-, and ten-class MNIST classification, achieving over 90% accuracy in five-class MNIST classification. This shows the importance of causal learning, where synaptic weights increase when pre-synaptic neurons trigger post-synaptic firing. We observed trade-offs between classification performance and hardware realism of synapses (Linear  $\rightarrow$  Non-linear). The Non-linear and Linear models achieved 91.07% and 91.67% accuracy respectively, while the Ideal synapse model reached 92.73%. Accuracy dropped significantly when synapse states were reduced from 25 to 5, indicating that hardware with few memory states performs poorly. Despite these limitations, our SNN architecture offers several advantages over traditional neural networks. It converges in fewer training epochs and achieves decent accuracy with only 100 training images for five-class classification. This efficiency makes SNNs attractive for applications like autonomous systems and edge devices, where conserving power is essential.

**Acknowledgments.** This work is supported in part by Anusandhan National Research Foundation (ANRF) Inclusivity Research Grant (IRG) ANRF/IRG/2024/000139/ENS and startup research grant by IIT Gandhinagar. This work is supported from the tools received under the C2S project from MEITY, Government of India.

**Disclosure of Interests.** We have no competing interests to declare that are relevant to the content of this article.

## References

1. Abbott, L.F., Nelson, S.B.: Synaptic plasticity: taming the beast. *Nature Neuroscience* **3**(S11), 1178–1183 (nov 2000). <https://doi.org/10.1038/81453>
2. Bi, G.q., Poo, M.m.: Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of Neuroscience* **18**(24), 10464–10472 (1998). <https://doi.org/10.1523/JNEUROSCI.18-24-10464.1998>
3. Biswas, A., Prasad, S., Lashkare, S., Ganguly, U.: A simple and efficient snn and its performance & robustness evaluation method to enable hardware implementation (2016), <https://arxiv.org/abs/1612.02233>
4. Diehl, P.U., Cook, M.: Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in Computational Neuroscience* **Volume 9 - 2015** (2015). <https://doi.org/10.3389/fncom.2015.00099>
5. Fontaine, B., Peña, J.L., Brette, R.: Spike-threshold adaptation predicted by membrane potential dynamics in vivo. *PLoS Computational Biology* **10**(4), e1003560 (apr 2014). <https://doi.org/10.1371/journal.pcbi.1003560>
6. Furber, S.: Digital neuromorphic technology: current and future prospects. *National Science Review* **11**(5), nwad283 (11 2023). <https://doi.org/10.1093/nsr/nwad283>
7. Goupy, G., Tirilly, P., Bilasco, I.M.: Paired competing neurons improving stdp supervised local learning in spiking neural networks. *Frontiers in Neuroscience* **Volume 18 - 2024** (2024). <https://doi.org/10.3389/fnins.2024.1401690>
8. Guan, X., Mo, L.: Unsupervised conditional reflex learning based on convolutional spiking neural network and reward modulation. *IEEE Access* **8**, 17673–17690 (2020). <https://doi.org/10.1109/ACCESS.2020.2968240>

9. Gupta, A., Long, L.N.: Character recognition using spiking neural networks. In: 2007 International Joint Conference on Neural Networks. pp. 53–58. IEEE (aug 2007). <https://doi.org/10.1109/ijcnn.2007.4370930>
10. Jiang, X., Zhang, Q., Sun, J., Cao, J., Ma, J., Xu, R.: Fully spiking neural network for legged robots. In: ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 1–5 (2025). <https://doi.org/10.1109/ICASSP49660.2025.10890793>
11. Kasiński, A., Ponulak, F.: Comparison of supervised learning methods for spike time coding in spiking neural networks. *International Journal of Applied Mathematics and Computer Science* **16**(1), 101–113 (2006), <http://eudml.org/doc/207768>
12. Kim, T., Hu, S., Kim, J., Kwak, J.Y., Park, J., Lee, S., Kim, I., Park, J.K., Jeong, Y.: Spiking neural network (snn) with memristor synapses having non-linear weight update. *Frontiers in Computational Neuroscience* **15** (mar 2021). <https://doi.org/10.3389/fncom.2021.646125>
13. Koch, C., Segev, I. (eds.): *Methods in Neuronal Modeling: From Ions to Networks*. MIT Press, Cambridge, MA, 2nd edn. (2003)
14. Lashkare, S., Chouhan, S., Bhat, A., Ganguly, U.: General izhikevich dynamics in pr0.7 ca0.3 mno3 rram neuron. In: 2020 International Symposium on VLSI Technology, Systems and Applications (VLSI-TSA). p. 48–49. IEEE (aug 2020). <https://doi.org/10.1109/vlsi-tsa48913.2020.9203710>
15. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. *Proceedings of the IEEE* **86**(11), 2278–2324 (1998). <https://doi.org/10.1109/5.726791>
16. Moitra, A., Bhattacharjee, A., Li, Y., Kim, Y., Panda, P.: When in-memory computing meets spiking neural networks—a perspective on device-circuit-system-and-algorithm co-design. *Applied Physics Reviews* **11**(3), 031325 (09 2024). <https://doi.org/10.1063/5.0211040>
17. Platkiewicz, J., Brette, R.: A dynamical system analysis of the adaptive spike threshold. *BMC Neuroscience* **8**(S2) (Jul 2007). <https://doi.org/10.1186/1471-2202-8-s2-p119>
18. Prezioso, M., Merrih-Bayat, F., Hoskins, B.D., Adam, G.C., Likharev, K.K., Strukov, D.B.: Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature* **521**(7550), 61–64 (May 2015). <https://doi.org/10.1038/nature14441>
19. Shen, G., Zhao, D., Zeng, Y.: Backpropagation with biologically plausible spatiotemporal adjustment for training deep spiking neural networks. *Patterns* **3**(6), 100522 (2022). <https://doi.org/10.1016/j.patter.2022.100522>
20. Sporea, I., Grüning, A.: Supervised learning in multilayer spiking neural networks. *Neural Computation* **25**(2), 473–509 (2013). [https://doi.org/10.1162/NECO\\_a\\_00396](https://doi.org/10.1162/NECO_a_00396)
21. Wittenberg, G.M., Wang, S.S.H.: Malleability of spike-timing-dependent plasticity at the ca3-ca1 synapse. *Journal of Neuroscience* **26**(24), 6610–6617 (jun 2006). <https://doi.org/10.1523/jneurosci.5388-05.2006>
22. Xu, Y., Zeng, X., Han, L., Yang, J.: A supervised multi-spike learning algorithm based on gradient descent for spiking neural networks. *Neural Networks* **43**, 99–113 (2013). <https://doi.org/10.1016/j.neunet.2013.02.003>
23. Zhao, D., Zeng, Y., Li, Y.: Backeisnn: A deep spiking neural network with adaptive self-feedback and balanced excitatory–inhibitory neurons. *Neural Networks* **154**, 68–77 (2022). <https://doi.org/10.1016/j.neunet.2022.06.036>