Approximating Back-propagation for a Biologically Plausible Local Learning Rule in Spiking Neural Networks

Amar Shrestha
Syracuse University
Syracuse, NY, USA
amshrest@syr.edu

Haowen Fang
Syracuse University
Syracuse, NY, USA
hfang02@syr.edu

Qing Wu
US Air Force Research Laboratory
NY, USA
qing.wu.2@us.af.mil

Qinru Qiu
Syracuse University
Syracuse, NY, USA
qiqiu@syr.edu

ABSTRACT

Asynchronous event-driven computation and communication using spikes facilitate the realization of spiking neural networks (SNN) to be massively parallel, extremely energy efficient and highly robust on specialized neuromorphic hardware. However, the lack of a unified robust learning algorithm limits the SNN to shallow networks with low accuracies. Artificial neural networks (ANN), however, have the backpropagation algorithm which can utilize gradient descent to train networks which are locally robust universal function approximators. But backpropagation algorithm is neither biologically plausible nor neuromorphic implementation friendly because it requires: 1) separate backward and forward passes, 2) differentiable neurons, 3) high-precision propagated errors, 4) coherent copy of weight matrices at feedforward weights and the backward pass, and 5) non-local weight update. Thus, we propose an approximation of the backpropagation algorithm completely with spiking neurons and extend it to a local weight update rule which resembles a biologically plausible learning rule spike-timing-dependent plasticity (STDP). This will enable error propagation through spiking neurons for a more biologically plausible and neuromorphic implementation friendly backpropagation algorithm for SNNs. We test the proposed algorithm on various traditional and non-traditional benchmarks with competitive results.

KEYWORDS

Spiking Neural Networks, backpropagation, Neuromorphic, Local Learning, Spike-Timing Dependent Plasticity

1 INTRODUCTION

The brain uses discrete action potentials (spikes) in time through plastic synapses for communication. This has inspired the third generation of neural networks called spiking neural networks (SNN). The spikes have uniform magnitude and are sparse in time. Although the SNNs utilize a highly simplified spike generation mechanisms as compared to its biological counterpart [6], each spike still contains high information content and its conveyed through spike timing and/or spike rates. This sparse communication mechanism facilitates energy reduction in the brain and also translates to hardware implementations [14][20].

Besides requiring minimal power, the brain can make sense of huge amounts of concurrent sensory information from a noisy environment through deep and complex neural structures such as the visual cortex, primary auditory cortex etc. Although spiking networks have theoretically been shown to have Turing-equivalent computing power [13], it remains a challenge to train deep SNNs; threshold functions that generate spikes are discontinuous, so they do not have derivatives and cannot directly utilize gradient-based optimization algorithms for training. Biologically plausible learning mechanism spike-timing-dependent plasticity (STDP) [18] and its variants are local in synapse and time, thus the plasticity of the synaptic weights are dependent on the relative timings of the pre and post-synaptic spikes. It is difficult to train multi-layer SNNs by utilizing only the local information. Thus, SNNs utilizing STDP are limited to very shallow networks [4][22].

In contrast to SNNs, artificial neural networks (ANNs) are comprised of neurons with continuous non-linear activation functions and communicate through high-precision values. These neurons are differentiable enabling gradient-based optimization methods and they can be stacked in multiple (deep) layers producing a locally robust universal function approximator. Recently, with the availability of computing power and large labeled datasets, ANNs have become very deep producing breakthrough performances in various fields of pattern recognition. Despite their effectiveness, they are computationally expensive and not suitable for implementations on portable hardware.

Even though ANNs were originally inspired by the brain and the representations learned by ANNs in tasks like image classification is shown to be similar to those observed in receptive fields of the visual cortex [26], learning the synaptic weights through backpropagation appears biologically unrealistic [3] in various ways: (a) Neuron models in ANNs send a continuous-valued output which is biologically unrealistic. Whereas, it is not trivial to compute their derivative. (b) Neuron activities are not differentiable, while gradient descent requires local derivative of the neuron to backpropagate the error. (c) The connection between neurons in SNN is unidirectional. A backward path must be added specifically. However, it then will create the "weight transport" problem. Two copies of the weight must be stored in both the forward path and backward path. Since the weight is constantly changing during learning, maintaining the coherence between the two copies will create significant overhead. (d) Errors in ANN are propagated as real value. This representation is not compatible with SNN.

In this work, we systematically solve the above biological implausibility issues with backpropagation with an intent to approximate the entire backpropagation algorithm utilizing only the spiking neurons and adapt it towards a local learning rule which resembles STDP. We validate the approximation and the local learning rule using three benchmarks. The following summarizes our contributions:

(1) We formulated the backpropagation algorithm in terms of spiking neurons to approximate the non-linearity of the forward pass and the linearity of the backward pass.

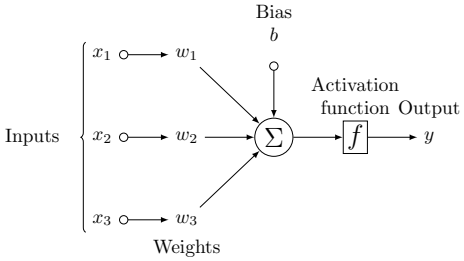


Figure 1: Spiking Neuron Model

(2) We show that, through a proper connection between the forward and backward paths, gradient descent can be achieved using local weight update that resembles STDP. This connection is derived by adapting the back-propagated errors as the target difference in the difference target propagation algorithm.
(3) We propose to use a neural circuit to calculate the derivative of the *L2* loss in terms of spikes.

2 RELATED WORKS

There have been various approaches to adopt the backpropagation algorithm to train deep SNNs directly. One category of approaches keeps track of the membrane potential at spike times and backpropagate errors based on that. SpikeProp [2] is the first attempt to train an SNN using such an approach. But SpikeProp is very limited to single-spike learning. A similar category of approach [27] [11] treats the discontinuities during spike times as noise and smoothen the membrane potential to essentially make is continuous. These approaches utilize spike-rate to compute the loss and membrane potential to compute the error derivative, and hence create a discrepancy. Neftci et. al. (2017) [138] proposed an event-driven random backpropagation (eRBP) algorithm simplifying the backpropagation chain path. But this work requires multicompartmental neurons to enable error to locally modulate plasticity. In [25], a supervised learning method was proposed (BP-STDP) where the backpropagation update rules were converted to temporally local STDP rules for multilayer SNNs.

In summary, existing works have some major limitations: a) they either require high-precision back-propagating error derivatives or do not adhere to the linearity of the original backpropagation algorithm. b) Each neuron must know the membrane potential of its presynaptic neighbors in order to determine the synaptic weight change.c) Each neuron must also know the error derivatives of its postsynaptic neighbors in order to calculate its own error derivative. All these limitations violate the constraints of local communication rule in spike domain. Meanwhile, there are approaches [21][19] to convert ANNs to SNNs which is not condusive to neuromorphic implementation and are thus out-of-scope for this work.

3 METHODS

In this section, we analyze the root cause of biological implausibilities in the backpropagation algorithm and later we will show that they can be eliminated for SNNs through a suitable sequence of approximation techniques. This includes the approximation of the derivatives of spiking neurons (Section 3.1), representing linearly propagated errors using spiking neurons (Section 3.2), local learning through error spikes (section 3.3) and completely spike-based computation of *L2* loss's derivative (Section 3.4). The whole method is put together in the algorithm in Section 3.6. Table 1 provides a list of symbols used relating to neuron's state and output for clarity.

Symbol Forward pass	Symbol Backward pass	Meaning
U	E	Membrane potential
u	ϵ	Accumulated sub-threshold membrane potential
h	e	Spike count
s	es	Spike train

Table 1: Symbols

3.1 Derivatives of Spiking Neuron Function

Fig. 1 shows a general model of a simple neuron in a neural network in which there are a set of inputs (x_1, x_2, x_3), synapses from those inputs to the neuron (w_1, w_2, w_3), those inputs are accumulated and then goes through an activation function to produce an output y. In SNNs, the inputs and outputs are spike trains, and the activation function is a discontinuous threshold function. The Leaky Integrate-and-Fire (LIF) is the most popular neuron model [24] because of its simplistic representation of a spiking neuron. Here, we utilize a discrete-time variant of the non-leaky IF neuron that accumulates the input spikes over time and produces a binary spike when the membrane potential crosses a clear threshold. The membrane potential of neurons in layer i at time t is:

$$U_i(t) = \sum w_{i-1,i}.s_{i-1}(t) + b_i + U_i(t-1) \tag{1}$$

$$\text{if} \quad U_i(t) \geq \theta \quad \text{then} \quad s_i(t) = 1, \quad \text{else} \quad s_i(t) = 0$$

where $w_{i-1,i}$ is the synaptic weight between pre-synaptic neurons in layer $i-1$ and post-synaptic neurons in layer i, b_i is the bias. $s_{i-1}(t)$ and $s_i(t)$ are pre and postsynaptic spike trains respectively. The neuron spikes when the membrane potential reaches the threshold θ and it's membrane potential is reset to the resting potential of 0.

The spike in the spike train is represented as a delta Dirac function at the spike time t_s

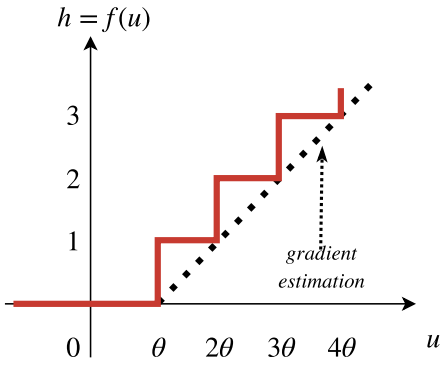
$$s(t_s) = \delta(t - t_s)$$

such that the sub-threshold membrane potential of the postsynaptic neuron between consecutive spikes $s_i(t_{s1})$ and $s_i(t_{s2})$ is given by

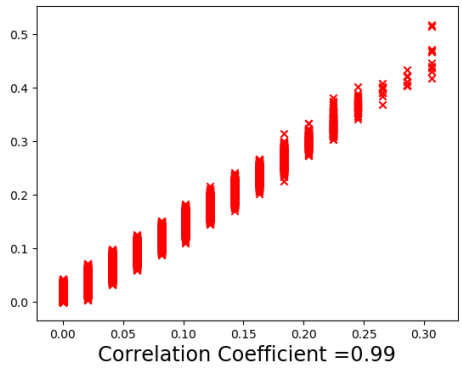
$$U_i(t) = \sum w_{i-1,i}(\sum_{t=t_{s1}}^{t_{s2}} \delta(t - t_s)) + b_i(t_{s2} - t_{s1}) \tag{2}$$

The activity of a neuron can be represented either in terms of their rate or spike count. Here, as we intend to compute the loss based on the spike counts, we utilize the spike counts as neuron's activity which is given by the sum of delta Dirac functions over the spiking time interval T as shown in Eq. 3.

$$h(T) = \sum_{t=0}^{T} s(t_s) = \sum_{t=0}^{T} \delta(t - t_s) \tag{3}$$



(a) Spiking Activation Function



(b) Correlation between ANN (y-axis) and SNN (x-axis)

Figure 2: Approximation of the forward pass

where t_s are the post-synaptic spike times of neuron j.

The spike counts can also be represented in terms of the neuron 's membrane potential.

$$h(T) = f(u(T)) = \left\lfloor \frac{u(T)}{\theta} \right\rfloor \tag{4}$$

where θ is the spiking threshold and $u(T)$ is the accumulated sub-threshold membrane potential computed in Eq. 2 over the time period T

$$u(T) = \sum_{s=0}^{N} U(t_s)$$

where t_s is the post-synaptic spike time, N is the number of spikes in the post-synaptic spike train such that $T > t_N$

The fig. 2(a) shows the function $f(u)$ for the spike count over time T is a staircase function of the total accumulated sub-threshold membrane potential u. This represents the activation function of the spiking neurons over that time window. Fig. 2(b) shows a strong correlation of the hidden layers in the forward pass between the ANN and the SNN with the same weight initialization and input sample in the MNIST experiments. The ladder-like effect seen in Fig. 2(b) due to the discretization caused by the limited time window.

The function is clearly discontinuous and has no derivatives when $u = n\theta$ where $n = 1, 2, .., T$. The derivative is zero at all other u. To estimate the function's gradient, we approximate $f(u)$ by a piece-wise linear function as shown byt the dotted line in fig. 2(a). The function is zero for $u < \theta$, then increases linearly with a slope $\frac{1}{\theta}$. This approximation resembles a ReLU function shifted by θ. This shift by θ prevents computation of the gradient based on the membrane potential before the first spike. This is important as we compute the loss in terms of spikes and require the gradient on the basis of the spikes. Thus, the derivative of $f(u)$ is approximated as

$$h' = f'(u) = \begin{cases} \frac{1}{\theta} & \text{if } u \geq \theta \\ 0 & \text{if } u < \theta \end{cases} \tag{5}$$

3.2 Spike Representation of Error Derivatives

Let us assume a network with l layers. For a hidden layer i which is post-synaptic for layer $i-1$ and pre-synaptic for layer $i+1$ in a time period T,

$$u_i = w_{i-1,i} \cdot h_{i-1} + b_i \tag{6}$$

and the spike count from Eq. 4 is

$$h_i = \left\lfloor \frac{u_i}{\theta_i} \right\rfloor$$

Now, let us assume a loss function L such that the derivative of the loss function at the $i+1$ layer is

$$\epsilon_{i+1} = \frac{\partial L}{\partial h_{i+1}}$$

Now, using the chain rule, the error derivative at layer i is:

$$\epsilon_i = \frac{\partial L}{\partial h_i} = \frac{\partial L}{\partial h_{i+1}} \cdot \frac{\partial h_{i+1}}{\partial u_{i+1}} \cdot \frac{\partial u_{i+1}}{\partial h_i}$$

$$\epsilon_i = \epsilon_{i+1} \cdot \frac{1}{\theta_{i+1}} \cdot w^T_{i,i+1} \quad \text{when } u_{i+1} > \theta_{i+1} \tag{7}$$

We intend to represent the error derivative using spiking neurons such that ϵ now represents the accumulated sub-threshold membrane potential for spiking neurons without bias. Such that we utilize the same activation function as for the feedforward path given in Eq. 4 to get a formulation as shown in Eq. 6

$$\epsilon_i = e_{i+1} \cdot w^T_{i,i+1} \tag{8}$$

where e_3 is the error spike count such that we can model the error network with IF neurons as well. In other words, ϵ_i is the accumulated sub-threshold membrane potential of the error neuron whose input is the spikes e_{i+1} from the upper-level error neuron.

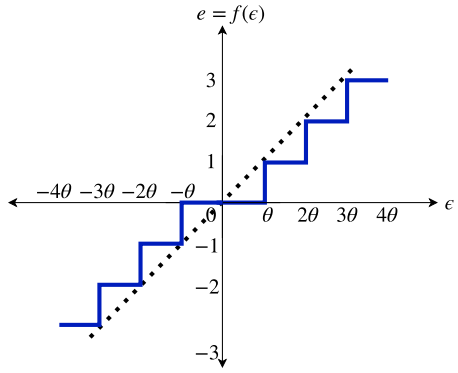
So, the membrane potential of error neurons in in layer i at time t is:

$$E_i(t) = \sum w^T_{i,i+1}.es_{i+1}(t) + E_i(t-1)$$

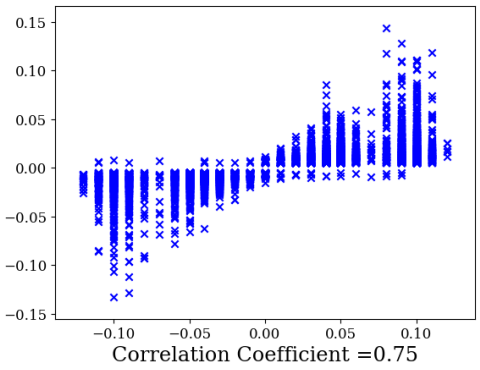
$$\text{if} \quad E_i(t) \geq \theta \quad \text{then} \quad es_i(t) = 1, \quad \text{else} \quad es_i(t) = 0$$

where $w_{i,i+1}$ is the synaptic weight matrix from layer i to layer $i+1$ and $es_{i+1}(t)$ is the spike train in the feedback path from layer $i+1$. The neuron spikes when the membrane potential reaches the threshold θ and it's membrane potential is reset to the resting potential of 0.

The activation function in Eq. 4 approximates a ReLU function which is non-linear. To approximate the linearity of the backward pass of backpropagation, we adopt a two-channel spikes approach; one for positive and another for a negative spike as mentioned in [23] and the spike count for the error derivatives is given as



(a) Spiking Activation Function



(b) Correlation between ANN (y-axis) and SNN (x-axis)

Figure 3: Approximation of the backward pass

$$e = \begin{cases} e^+ = \lfloor \frac{\epsilon}{+\theta} \rfloor & \text{if } \epsilon > 0 \\ e^- = \lfloor \frac{\epsilon}{-\theta} \rfloor & \text{if } \epsilon < 0 \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Eq. 9 essentially performs a linear combination of positive and negative versions of the non-linear function Eq. 4 to approximate a linear function as shown in fig. 3(a). Fig. 3(b) shows a fairly strong correlation of the hidden layers in the backward pass between the ANN and the SNN with the same weight initialization and input sample in the MNIST experiments.

3.3 Local Learning

Given Eq. 5 and 7 and the chain rule, we can derive the weight update for incoming weights in layer i as

$$\Delta w_{i-1,i} = \epsilon_i \cdot h_i^{'} \cdot h_i \tag{10}$$

In terms of the synapse, this weight update is based on local variables presynaptic activity and derivative of the postsynaptic activity. But the error derivative is not local to the synapse. So, we take inspiration from the concept of an alternate credit assignment method called difference target propagation [10] and modify it to derive a local weight update formulation.

The main idea of target propagation is to provide each feedforward unit's activation a target value which is close to the activation value. Once the target is computed, the gradient of the loss between the feedforward value and the target value is propagated only locally. In the limit where the target is very close to the feedforward value, target propagation should behave like backpropagation.

Let us assume a network with l layers. Let $\hat{h_i}$ be the target spike count for the hidden layer i, such that the local $L2$ loss is given as

$$L(\hat{h_i}, h_i) = \left\| \hat{h_i} - h_i \right\|_2^2$$

such that the local weight update using the chain rule is given as

$$\Delta w_{i-1,i} = \frac{\partial L(\hat{h_i}, h_i)}{\partial w_{i-1,i}} = \frac{\partial L(\hat{h_i}, h_i)}{\partial h_i} \cdot \frac{\partial h_i}{\partial u_i} \cdot \frac{\partial u_i}{\partial w_{i-1,i}}$$

$$\Delta w_{i-1,i} \propto (\hat{h_i} - h_i) \cdot \frac{1}{\theta_i} \cdot h_{i-1} \quad \text{when } u_i > \theta_i \tag{11}$$

where the threshold θ is a constant that is incorporated in the learning rate η such that local weight update is given as

$$\Delta w_{i-1,i} = \eta \cdot (\hat{h_i} - h_i) \cdot h_{i-1} \tag{12}$$

In [10], the targets are determined utilizing an autoencoder with inverse synaptic weights which are learnt through reconstruction. Here, we use the error derivatives as the difference to determine the target activation value.

$$\hat{h_i} = f(\hat{u_i}) = f(u_i + e_i \cdot w_{ei})$$

From Eq. 4,

$$\hat{h_i} = \left\lfloor \frac{u_i + e_i \cdot w_{ei}}{\theta_i} \right\rfloor$$

Now, if $w_{ei} = \gamma \theta_i$ then

$$\hat{h_i} = h_i + \gamma e_i \tag{13}$$

Eq. 13 essentially provides the connectivity from the error neurons to the feedforward neurons as shown in fig. 4(b) which is a one-to-one connection with non-plastic synaptic weight equal to a constant γ factor of the spiking threshold, to produce the target spike count.

From Eq. 13 we can also conclude

$$\hat{h_i} - h_i \propto e_i \tag{14}$$

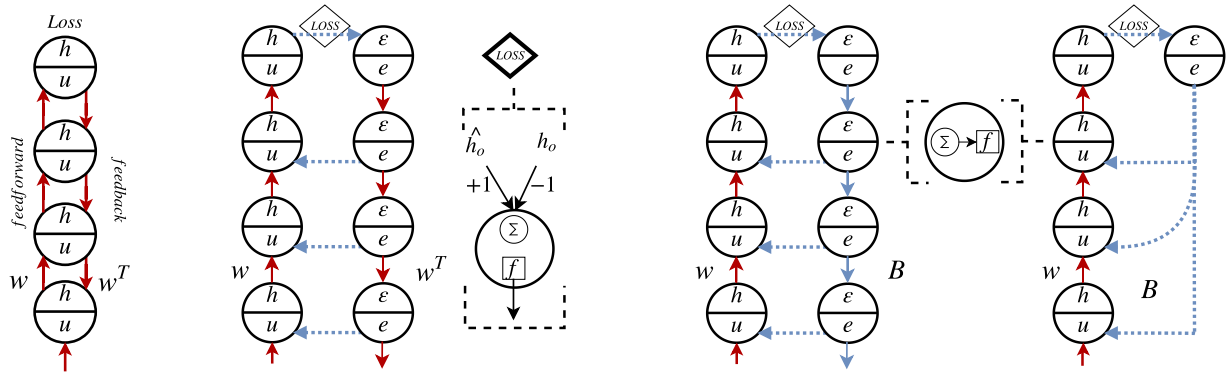
Interestingly, given Eq. 14, the local weight update in Eq. 12 relates closely to the formulation of the rate-based STDP rule in [1] in which the synaptic weight update is in proportion to the product of the presynaptic activity and the rate of change of the postsynaptic activity. This conclusion is supported by fig. 5 which shows the strong correlation between our local weight update and a basic two-factor event-based STDP rule that fit the biological findings in [18]. Thus, we call the local learning rule given by Eq. 12 Error-modulated STDP (EMSTDP).

3.4 Derivative of $L2$ Loss

We utilize the $L2$ loss for the output layer in a network with l layers given as

$$L(\hat{h_l}, h_l) = \left\| \hat{h_l} - h_l \right\|_2^2$$

where $\hat{h_o}$ and h_o are the target and output spike counts respectively.



(a) Original Backprop. **(b) Error-modulated STDP** **(c) Error-modulated STDP with FA and DFA**

Figure 4: Network structures

L2 loss's derivative results in a simple linear combination of the target and the output value

$$\frac{\partial L(\hat{h_l}, h_l)}{\partial h_l} = (\hat{h_l} - h_l)$$

The choice of *L2* loss is preferable as its derivative can be implemented simply by a specific connectivity as shown in fig. 4(b) of IF neurons whose membrane potential is given as

$$E_l(t) = w_L \hat{s_l}(t) + (-w_L) s_l(t) + E_l(t-1)$$

where $\hat{s_l}(t)$ and $s_l(t)$ are the target and output spike trains respectively and w_L is the non-plastic synaptic weight to the neuron that asynchronously computes the loss gradient in terms of spikes. The neural circuit is shown in fig. 4(b) with $w_L = 1$.

Thus, the accumulated subthreshold membrane potential of the output loss derivative is

$$\epsilon_l = w_L \hat{h_l} + (-w_L) h_l$$

And its propagated backwards as the error derivatives for the hidden layer as per Eq. 9 with a threshold θ_l. As the loss is determined per spike basis, the target output firing rate dictates the choice of θ_l. For example, if $w_L = 1$ then for a target output firing rate of 0.2 we fix the θ_l to 5.

3.5 Feedback Alignment

Probably, one of the most important issues is the weight transport problem in backpropagation in terms of biological plausibility. Two coherent copies of the weight must be stored in both the forward path and backward path but there is no known biological mechanism for synapses to know the synaptic strength of other synapses. Also, since the weight is constantly changing during learning, maintaining the coherence between the two copies will create significant overhead. [12] proposes the Feedback Alignment (FA) method which involves using fixed and random weights for the feedback path to convey error signals with no assumptions on its structure. In fully connected networks with FA, it was observed that the angle between the back-propagated gradient with symmetric weights and the FA propagated gradient converges from approximately orthogonal to roughly 45°, meaning that the FA weight updates are correlated but not identical to those with symmetric weights. [16] introduced direct feedback alignment (DFA) in which the output error signal is propagated directly to all hidden layers instead of through adjacent layers using fixed and random weights. In this work, we adopt both FA and DFA schemes to avoid the weight transport problem. The resulting networks are shown in fig. 4(c).

3.6 Error-modulated STDP pipeline

The formulation provided in the previous sections shows how to represent error and backpropagate it in spike domain. It also provides a guideline to design the feedback path including its connectivity and neuron model as shown in fig. 4. In fig. 4, red solid directed lines denote plastic synapses whereas blue lines denote fixed synapses. Fig. 4(a) denotes a vanilla multilayer perceptron (MLP) with a forward pass of continuous-valued activations and backward pass of continuous-valued error derivatives. Fig. 4(b) denotes the resultant network structure for EMSTDP derived in Section 3.3 and the spiking circuitry for the derivative of *L2* loss. Fig. 4(c) denotes the variations of EMSTDP in which weight symmetry is avoided by using feedback alignment and direct feedback alignment methods.

All the operations in the network with an arbitrary number of layer is asynchronous except for the weight update which is computed only when the error derivatives have driven the hidden layers to its targets. Algorithm 1 shows the error-modulated STDP algorithm for the SNN with an arbitrary number of layers.

Each input sample is a sequence of spikes (Poisson sampling of input intensity) which is presented for a period of T time steps. Given the local weight update rule Eq. 12, requires the postsynaptic spike counts before it's driven to its target and after it settles to the target, it necessitates two phases. The spikes are propagated through the forward pass for the whole period. After a warm-up period, ideally, $\frac{T}{2}$ when the firing rates for the hidden layer settle to a fixed point, the spiking circuit computing the derivative of the loss starts propagating spikes through the spiking error circuit.

The spiking error circuit drives the hidden layers in the forward pass toward its target. In the remaining time period, the neurons in the hidden layers settle towards its target firing rate. During the two phases, the local activity i.e. pre and postsynaptic spike counts are recorded, and at the end of T, the synaptic weight update is calculated by Eq. 12.

Algorithm 1: Error-Modulated STDP algorithm

Initialization: Time window T

Spiking neuron i in forward pass

Incoming spike train s_{i-1}

$h_i, \hat{h_i}, h_{i-1} \leftarrow 0$

begin

  while $t < T$ **do**

    $s_i(t) = 0$

    $U_i(t) += b_i$

    if $s_{i-1}(t) == 1$ **then**

      $U_i(t) += w_{i-1,i}$

      $h_{i-1}++$

    if $es_i(t) == 1$ **then**

      $U_i(t) += \gamma\theta_i$

    if $U_i(t) \geq \theta_i$ **then**

      $s_i(t) = 1$

      $U_i(t) += 0$

      if $t < T/2$ **then**

        h_i++

      else

        $\hat{h_i}++$

  $\Delta w_{i-1,i} = \eta(\hat{h_i} - h_i) \cdot h_{i-1}$

  $w_{i-1,i} += \Delta w_{i-1,i}$

Spiking neuron i in backward pass

Incoming spike train es_{i+1}

begin

  while $t < T$ **do**

    $es_i(t) = 0$

    if $t > T/2$ **then**

      if $es_{i+1}(t) == 1$ **then**

        $E_i(t) += w_{i+1,i}$

      if $E_i(t) \geq \theta_i$ **then**

        $es_i(t) = 1$

        $E_i(t) = 0$

4 EXPERIMENTS AND RESULTS

In this section, we will present different experiments and their results to evaluate the performance of the error-modulated STDP algorithm. In the first experiment, we can observe the conclusions presented in [1] such that our algorithm produces a profile of weight updates similar to [18]. Then, we'll test the algorithm on three datasets; MNIST digit classification, Fashion MNIST and Sign language classification.

Training an SNN can be difficult as the performance can be sensitive to some present hyperparameters. Thus, it's important to adopt and standardize hyperparameter selection during training, this means weight initialization and threshold selection for the algorithm.

In deep learning, it is essential to have a proper initialization method that should avoid reducing or magnifying the magnitudes of input signals exponentially [5]. Similarly, in SNNs it is essential to prevent vanishing and explosion of spikes through the layers. Variance matching initialization schemes attempt to keep $Var(y_L) \approx Var(y_1)$, where y_1 is the output of the first layer and y_L is the output of the last. In order to do so, the following condition must be met, for any given layer l: $\frac{1}{2} n_l Var(w_l) = 1$ where n_l is the number of neurons in the previous layer. Thus, based on [5], we draw the weights from a Gaussian distribution with $\mu = 0$ and $var = \frac{scale}{n_l}$.

Similarly, the choice of thresholds is also important to prevent extreme sparsity of spikes through the layers. Here, we adopt a simple standardization of the thresholds given by $\theta_l = n_l.\sigma(w_l).\beta$ where β is the constant that determines the percentage of input neurons to spike for output neuron to spike.

And throughout this section, we will use the following notations for the SNN network structure: Input dimensions are separated by $\times$, layers are separated by $-$ and the last dimension of the network structure is the output layer. For examples, $28 \times 28 - 500 - 10$ represents a network input of 28×28 dimension, a hidden layer with 500 neurons and an output layer with 10 neurons.

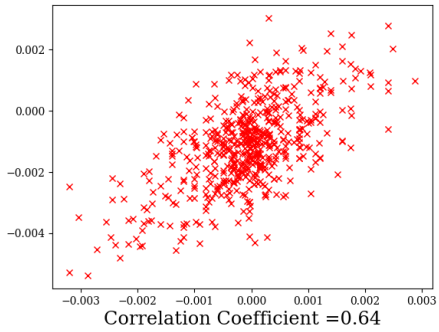


Figure 5: Correlation between EMSTDP and a basic STDP weight change

4.1 STDP Observation

[1] has shown that updating the weights in proportion to the rate of change of postsynaptic activity times the presynaptic activity yielded a behavior similar to the STDP observations. They also linked this STDP rule to stochastic gradient descent and suggested that STDP would do gradient descent on the prediction errors if the rate of change of postsynaptic activity is proportional to the gradient of the error. In our formulation of EMSTDP, we have shown that the EMSTDP algorithm satisfies the condition i.e. rate of change proportional to the error gradient. Thus, utilizing EMSTDP, we should achieve a strong correlation between our local weight update and a basic two-factor event-based STDP rule that fit the biological findings in [18].

In a simple network with two neurons (pre- and post-synaptic) connected through a synapse, we first simulated random sets of the postsynaptic spike trains (s_j) induced by an externally driven voltage or some intrinsic bias and without any pre-synaptic spikes. This is recorded as post-synaptic spike counts (h_j). Then we simulate random sets of pre-synaptic spike trains (s_i) through the synapse with a random set of strengths. For each of these combinations, we record the new post-synaptic spike counts ($\hat{h_j}$) to calculate the respective change in post-synaptic activity. For each configuration, we now compute the weight change based on Eq. 12. For the same combinations, we also apply a basic two-factor event-based STDP rule with an STDP window of $10ms$ and record the corresponding total weight change. The resulting fig. 5 shows the relationship between weight change for EMSTDP and the basic STDP rule.

Dataset	Method	Learning	Network Structure	Accuracy%
MNIST	Lee (2016) [11]	BP	$28 \times 28 - 800 - 10$	98.64
	Lee (2016) [11]	BP	$28 \times 28 - 500 - 500 - 10$	98.7
	Neftci (2017) [15]	BP	$28 \times 28 - 500 - 10$	97.71
	Neftci (2017) [15]	BP	$28 \times 28 - 500 - 500 - 10$	97.98
	O'Connor (2016) [17]	BP	$28 \times 28 - 300 - 300 - 10$	96.4
	Jin (2019) [7]	BP	$28 \times 28 - 800 - 10$	98.84
	Diehl (2015) [4]	STDP	$28 \times 28 - 1600 - 10$	95
	Tavanaei (2017) [25]	STDP	$28 \times 28 - 1000 - 10$	96.6
	Tavanaei (2017) [25]	STDP	$28 \times 28 - 500 - 150 - 10$	97.2
	EMSTDP-SW	STDP	$28 \times 28 - 500 - 10$	97
	EMSTDP-DFA	STDP	$28 \times 28 - 500 - 10$	96.8
	EMSTDP-SW	STDP	$28 \times 28 - 500 - 500 - 10$	97.3
	EMSTDP-DFA	STDP	$28 \times 28 - 500 - 500 - 10$	96.8
Fashion MNIST	Vanilla BP	BP	$28 \times 28 - 100 - 100 - 10$	87.7
	EMSTDP-SW	STDP	$28 \times 28 - 500 - 500 - 10$	86.1
	EMSTDP-DFA	STDP	$28 \times 28 - 500 - 500 - 10$	85.3
Australian sign language	Vanilla BP	BP	$45 \times 22 - 150 - 150 - 50$	98.5
	EMSTDP-SW	STDP	$45 \times 22 - 150 - 150 - 50$	97.5
	EMSTDP-DFA	STDP	$45 \times 22 - 150 - 150 - 50$	97.1

Table 2: Performance Comparisons

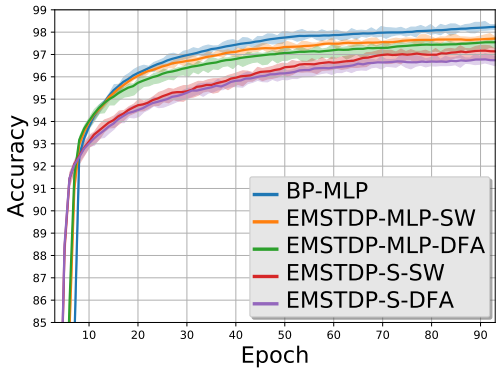


Figure 6: Performance comparison of vanilla BP, EMSTDP in MLP and EMSTDP in SNN with symmetric weights (SW) and direct feedback alignment (DFA)

4.2 MNIST Digit Classification

MNIST [9] is a standard benchmark to test the performance of representation learning algorithms. The task is to classify handwritten digits from 0 to 9. The MNIST handwritten digit dataset consists of 60k samples for training and 10k for testing, each of which is a 28×28 grayscale image. We convert each pixel value of an MNIST image into a spike train using Poisson sampling based on which the probability of spike generation is proportional to the pixel intensity. Using Poisson sampling, we flatten and encode each 28×28 image of the MNIST dataset into a 784 spike trains with duration T. The simulation time step is set to be 1ms. No pre-processing or data augmentation is done in our experiments.

The experiments are carried out for two network structures $28 \times 28 - 500 - 10$ and $28 \times 28 - 500 - 500 - 10$ for symmetric weights (SW) and direct feedback weights (DFA) each. Each configuration is run for a time window of $T = 200$ for each spike sequence. Each network is trained for 200 epochs, each epoch containing randomly sampled 10k images from the training set. For comparison, we also train an MLP with vanilla back-prop (BP-MLP) and an MLP with a network structure and weight update as given by EMSTDP (EMSTDP-MLP). All the results presented in this section are an average of 5 trials after 200 epochs for the best set of parameters obtained from an extensive grid search.

Fig. 6 shows the convergence comparison for BP-MLP, EMSTDP-MLP, and EMSTDP with spiking neurons (EMSTDP-S) with symmetric weights (SW) and direct feedback weights (DFA) for a time window T and two-layered architecture. As expected, MLP with backpropagation converge faster and to a higher accuracy (98.2%) than EMSTDP-MLP (97.8%) which is a modification of difference target propagation in which change in activity doesn't purely represent the error derivative. The spiking EMSTDP performed slightly worse (97.3%) that the MLP counterpart which is again expected due to the loss in precision due to spike encoding and approximations in the forward and backward pass. The DFA versions of EMSTDP-MLP (97.5%) and EMSTDP-S (96.8%) also perform worst than their symmetric weight counterparts as DFA is a less exact optimization method compared to BP with symmetric weights [16].

In the experiments, we also observed that with larger time window T there is faster convergence and better accuracies. This is expected as the input spikes trains are generated with Poisson sampling and a larger time window allows the spike train to better represent the input pixel values. Similarly, through the hidden layer, the precision of the approximated activation function is better with higher time window.

Table 2 shows our results in comparison to related works. The results show that EMSTDP produces competitive to better results as compared to other works such as [25] where the learning rule is only temporally local or perform regular forms of STDP [4]. Our work produces comparable but slightly worse results than related

works which only approximate backpropagation in an SNN [15] [11] [7] where the error is non-spiking. The loss in performance can be assigned to the approximation and propagation of error derivatives using spiking neurons and also to the attempt to adapt backpropagation to a local weight update rule.

4.3 Fashion MNIST

Fashion-MNIST [28] is a dataset comprising of 28×28 grayscale images of 70, 000 fashion products from 10 categories. The training set has 60, 000 images and the test set has 10, 000 images and it is intended to serve as a direct drop-in replacement for the original MNIST dataset, as it shares the same image size, data format and the structure of training and testing splits. Table 2 shows our results in comparison to an ANN with vanilla BP. The results show that EMSTDP produces competitive results for this dataset as well.

4.4 Sign Language

We trained an MLP with back-prop, SNN with EMSTDP-SW and EMSTDP-DFA with network structure $45 \times 22 - 150 - 150 - 50$ to recognize Australian sign language symbols. The Australian sign language dataset [8] involve movements of hand, wrist, and fingers. The data set is collected from two Fifth Dimension Technologies (5DT) data gloves, one right and one left. Each sample is a sequence of data frames containing sensor data. Each sign has approximately 45 frames on average. Since the number of samples is limited, we augmented the data by adding 10% noise. The objective in this domain is: given labeled recordings of different signs, learn to classify an unlabelled instance. Table 2 shows the corresponding results.

5 CONCLUSION

In this work, we devised a set of approximations and formulations to move backpropagation towards a more biologically plausible local learning algorithm EMSTDP. We first separate the forward and backward pass of the back-prop algorithm into two separate spiking networks. We formulate the discontinuous activation function of the forward and backward in terms of spike counts such that it approximates a ReLU function and a linear function respectively. We then apply the idea of difference target propagation to derive a local credit assignment rule which resembles as well as produce STDP characteristics. And finally, we solve the problem of weight symmetry in back-prop by utilizing direct feedback alignment. We tested this algorithm on MNIST, Fashion MNIST and Australian sign language dataset with results comparable to better than related works.

REFERENCES

[1] Yoshua Bengio, Thomas Mesnard, Asja Fischer, Saizheng Zhang, and Yuhuai Wu. 2015. STDP as presynaptic activity times rate of change of postsynaptic activity. *arXiv preprint arXiv:1509.05936* (2015).

[2] Sander M. Bohte, Joost N. Kok, and Johannes A. La PoutrÃĪ. 2000. SpikeProp: backpropagation for networks of spiking neurons.. In *ESANN*. 419–424.

[3] Francis Crick. 1989. The recent excitement about neural networks. *Nature* 337, 6203 (1989), 129–132.

[4] Peter U. Diehl and Matthew Cook. 2015. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in Computational Neuroscience* 9 (2015), 99–99.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*. 1026–1034.

[6] Alan L Hodgkin and Andrew F Huxley. 1952. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology* 117, 4 (1952), 500–544.

[7] Yingyezhe Jin, Wenrui Zhang, and Peng Li. 2018. Hybrid macro/micro level backpropagation for training deep spiking neural networks. In *Advances in Neural Information Processing Systems*. 7005–7015.

[8] Mohammed Waleed Kadous et al. 2002. *Temporal classification: Extending the classification paradigm to multivariate time series*. University of New South Wales.

[9] Yann LeCun. 1998. The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/* (1998).

[10] Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, and Yoshua Bengio. 2015. Difference target propagation. *european conference on machine learning* (2015), 498–515.

[11] Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. 2016. Training Deep Spiking Neural Networks Using Backpropagation. *Frontiers in Neuroscience* 10 (2016), 508.

[12] Timothy P. Lillicrap, Daniel Cownden, Douglas B. Tweed, and Colin J. Akerman. 2016. Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications* 7, 1 (2016), 13276.

[13] Wolfgang Maass. 1996. Lower bounds for the computational power of networks of spiking neurons. *Neural Computation* 8, 1 (1996), 1–40.

[14] Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, et al. 2014. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* 345, 6197 (2014), 668–673.

[15] Emre O Neftci, Charles Augustine, Somnath Paul, and Georgios Detorakis. 2017. Event-driven random back-propagation: Enabling neuromorphic deep learning machines. *Frontiers in neuroscience* 11 (2017), 324.

[16] Arild NÃżkland. 2016. Direct Feedback Alignment Provides Learning in Deep Neural Networks. *neural information processing systems* (2016), 1037–1045.

[17] Peter O'Connor and Max Welling. 2016. Deep spiking networks. *arXiv preprint arXiv:1602.08323* (2016).

[18] Guo qiang Bi and Mu ming Poo. 1998. SYNAPTIC MODIFICATIONS IN CULTURED HIPPOCAMPAL NEURONS : DEPENDENCE ON SPIKE TIMING, SYNAPTIC STRENGTH, AND POSTSYNAPTIC CELL TYPE. *The Journal of Neuroscience* 18, 24 (1998), 10464–10472.

[19] Daniel Rasmussen. 2018. NengoDL: Combining deep learning and neuromorphic modelling methods. *arXiv* 1805.11144 (2018), 1–22. http://arxiv.org/abs/1805.11144

[20] Jae-sun Seo, Bernard Brezzo, Yong Liu, Benjamin D Parker, Steven K Esser, Robert K Montoye, Bipin Rajendran, José A Tierno, Leland Chang, Dharmendra S Modha, et al. 2011. A 45nm CMOS neuromorphic chip with a scalable architecture for learning in networks of spiking neurons. In *2011 IEEE Custom Integrated Circuits Conference (CICC)*. IEEE, 1–4.

[21] William Severa, Craig M Vineyard, Ryan Dellana, Stephen J Verzi, and James B Aimone. 2018. Whetstone: A Method for Training Deep Artificial Neural Networks for Binary Communication. *arXiv preprint arXiv:1810.11521* (2018).

[22] Amar Shrestha, Khadeer Ahmed, Yanzhi Wang, and Qinru Qiu. 2017. Stable spike-timing dependent plasticity rule for multilayer unsupervised and supervised learning. In *2017 International Joint Conference on Neural Networks (IJCNN)*. 1999–2006.

[23] Amar Shrestha, Khadeer Ahmed, Yanzhi Wang, David P. Widemann, Adam T. Moody, Brian C. Van Essen, and Qinru Qiu. 2017. A spike-based long short-term memory on a neurosynaptic processor. In *Proceedings of the 36th International Conference on Computer-Aided Design*. 631–637.

[24] Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothee Masquelier, and Anthony S. Maida. 2019. Deep learning in spiking neural networks. *Neural Networks* 111 (2019), 47–63.

[25] Amirhossein Tavanaei and Anthony S. Maida. 2019. BP-STDP: Approximating backpropagation using spike timing dependent plasticity. *Neurocomputing* 330 (2019), 39–47.

[26] James C. R. Whittington, Timothy H. Muller, Shirley Mark, Caswell Barry, and Timothy E. J. Behrens. 2018. Generalisation of structural knowledge in the Hippocampal-Entorhinal system. *neural information processing systems* (2018), 8493–8504.

[27] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. 2018. Spatio-Temporal Backpropagation for Training High-performance Spiking Neural Networks. *Frontiers in Neuroscience* 12 (2018).

[28] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. arXiv:cs.LG/cs.LG/1708.07747