# A Burst-Dependent Algorithm for Neuromorphic On-Chip Learning of Spiking Neural Networks

**Michael Stuck**[1] **and Richard Naud**[1,2,3,4]

[1]: Department of Physics, University of Ottawa, 150 Louis Pasteur Pvt, Ottawa, ON, K1N 1A2, Canada
[2]: Department of Cellular and Molecular Medicine, University of Ottawa, 451 Smyth Rd, Ottawa, ON, K1H 8M5, Canada
[3]: Centre for Neural Dynamics and Artificial Intelligence, University of Ottawa, 150 Louis Pasteur Pvt, Ottawa, ON, K1N 1A2, Canada
[4]: University of Ottawa Brain and Mind Research Institute University of Ottawa, 451 Smyth Rd, Ottawa, ON, K1H 8M5, Canada

E-mail: `mstuck@uottawa.ca`

June 2024

**Abstract.** The field of neuromorphic engineering addresses the high energy demands of neural networks through brain-inspired hardware for efficient neural network computing. For on-chip learning with spiking neural networks, neuromorphic hardware requires a local learning algorithm able to solve complex tasks. Approaches based on burst-dependent plasticity have been proposed to address this requirement, but their ability to learn complex tasks has remained unproven. Specifically, previous burst-dependent learning was demonstrated on a spiking version of the XOR problem using a network of thousands of neurons. Here, we extend burst-dependent learning, termed 'Burstprop', to address more complex tasks with hundreds of neurons. We evaluate Burstprop on a rate-encoded spiking version of the MNIST dataset, achieving low test classification errors, comparable to those obtained using backpropagation through time on the same architecture. Going further, we develop another burst-dependent algorithm based on the communication of two types of error-encoding events for the communication of positive and negative errors. We find that this new algorithm performs better on the image classification benchmark. We also tested our algorithms under various types of feedback connectivity, establishing that the capabilities of fixed random feedback connectivity is preserved in spiking neural networks. Lastly, we tested the robustness of the algorithm to weight discretization. Together, these results suggest that spiking Burstprop can scale to more complex learning tasks while maintaining efficiency, potentially providing a viable method for learning with neuromorphic hardware.

## 1. Introduction

The increasing energy demands of neural network-based artificial intelligence (AI) systems have created significant interest in developing more energy-efficient methods [1, 2]. Neuromorphic computing, which takes inspiration from the computational principles of the human brain, has emerged as a promising solution to reduce energy costs associated with AI systems. Two biological aspects, in particular, are thought to yield significant energy savings.

One is the event-based mode of communication [3, 4, 5]. Unlike traditional neural networks that rely on high-precision, continuous activations, SNNs operate using single-bit, binary activations, known as spikes. Running SNNs with low-precision parameters and high spatial and temporal sparsity levels contributes to this energy efficiency [1].

The other advantageous biological property is the co-localization of computing and memory which circumvents the von Neumann bottleneck [6, 7]. Current methods for training SNNs predominantly rely on conventional von Neumann computer architectures, with the learned synaptic weights transferred to spiking neuromorphic hardware to enable energy-efficient inference. While using spiking neuromorphic hardware for inference has demonstrated energy savings in certain applications [8], further energy efficiency can be achieved by training SNNs directly on hardware. Direct hardware training reduces energy consumption and allows for continual learning, essential for systems that need to dynamically adapt to changing environments. Achieving this requires new training algorithms that are compatible with the unique constraints of neuromorphic hardware.

The constraints presented by neuromorphic hardware for learning have many similarities with those present in biological systems. As such, a learning algorithm for spiking neuromorphic hardware must include certain characteristics which are thought to be important for biologically plausible learning. The most essential of these biologically plausible features of learning is local, event-driven synaptic plasticity [9]. This means that synaptic weight changes must be triggered by either presynaptic, postsynaptic or global events and can depend only on the states of the presynaptic neuron, postsynaptic neurons, a global state and the synapse itself. A local alternative to backpropagation is necessary to do on-chip learning of SNNs in neuromorphic hardware.

One promising group of local learning methods for training SNNs are burst-dependent or 'Burstprop' learning methods [10, 11]. These algorithms are grounded in the widely observed burst-dependence of synaptic plasticity [12, 13, 14, 15] and synaptic dynamics which confers synapses with the ability to communicate spike-timing patterns differently to different targets. In hierarchical networks, Burstprop can approximate the backpropagation of error algorithm with single-phase local learning. It accomplishes this by an ensemble multiplexing [16] of error signals, which are transmitted backwards through the network and integrated by dendritic compartments to steer plasticity while the network processes feedforward information through somatic neuron compartments. Thus far, however, Burstprop methods have been limited to solving the exclusive

or (XOR) problem in SNNs [10, 11]. For Burstprop to be considered viable for neuromorphic learning applications, it must be capable of learning more difficult tasks. Similarly, we currently don't know if feedback alignment, a technique used to perform credit assignment without an artificial tying of the connectivity, preserves its function in SNNs using local and online plasticity rules.

In this work, we improve and test current Burstprop methods, allowing it to scale to more difficult learning problems. We consider two improvements: 1) a regularizing feedback connection from the soma to the dendrite of every neuron and 2) communication of two types of bursts, allowing for the multiplexing of signed errors. We evaluate our algorithm using the MNIST digit classification benchmark and show that it can successfully learn this task. We also investigate the impact of the choice of feedback connectivity and weight discretization. The results support that our algorithm is a viable method for supervised learning of SNNs for neuromorphic implementation.

## 2. Background

### *2.1. Learning in SNNs*

Multiple methods for training SNNs have been developed. We first distinguish methods relying on non-local plasticity rules from those using local ones. The local plasticity rules can be further separated into burst-dependent plasticity and spike-timing-dependent plasticity.

*2.1.1. Non-Local Learning Methods* Nonlocal learning methods for SNNs include ANN-SNN conversion [17, 18, 19, 20] and backpropagation-through-time (BPTT) [21, 22]. With ANN-SNN conversion, a rate-based neural network is trained using the backpropagation algorithm which is then converted to a spiking neural network. In real applications, imperfect conversions will strongly alter performance, but this can be remedied with further fine-tuning [8]. With BPTT, backpropagation is performed on an SNN by using surrogate gradients to propagate the gradient through the spiking nonlinearity. To backpropagate gradients through time, the gradient must keep track of the network state at multiple time steps of past activity for each time step. This algorithm is therefore highly nonlocal (but local-only approximations have been derived [23]). These methods do not address the problem faced with training neuromorphic online, we will use these methods to provide a point of reference for the performance metrics.

*2.1.2. STDP Learning Methods* Spike-time-dependent plasticity (STDP) alters connectivity according to the relative timing of spikes in the presynaptic and postsynaptic neurons and is therefore fully local. One local learning algorithm, proposed by Kappel et al. [24] combines a STDP plasticity rule and winner-take-all dynamics and recurrent excitation to approximate hidden Markov model learning in SNNs. Relatedly,

Diehl and Cook [25] combined STDP with winner-take-all dynamics to train a non-recurrent network. Reading out the resulting representation with a supervised plasticity rule, they showed that a single layer could achieve 95% accuracy on MNIST. As only the readout is supervised, this method is sometimes referred to as unsupervised. Another STDP approach combines a trace set by relative coincidence and a global reinforcement by reward-associated signals [26, 27, 28], but such three-factor learning rules tend to scale poorly on problems having a high number of dimensions. Also, how these STDP approaches can scale to multiple layers in a hierarchy remains unknown.

*2.1.3. Burstprop Learning Methods* Another class of local learning algorithms depend on separate somatic and dendritic compartments to represent both feedback error signals and inference signals within the neuron, often through a rate-based representation [29, 30, 31, 32]. For spiking neural networks, burst-dependent algorithms exploit the target-dependence of short-term plasticity [33, 34, 35], dendrite-dependent bursting [36, 37] and burst-dependent plasticity [13, 12, 14, 15] to perform credit assignment in a way that approximates backpropagation [10, 11, 38]. These burst-dependent learning methods, referred to as 'Burstprop' algorithms, can in principle allow for multilayer local learning in SNNs.

The original formulation [10] of the algorithm learns online but with signals coming in two phases. An initial phase in which synaptic plasticity was disabled was required to determine each neuron's baseline burst probability which was then used in the second phase to calculate synaptic plasticity, violating temporal locality. A second formulation of the algorithm (called BurstCCN [11]) adapted the learning rule for single-phase learning by including a second feedback pathway that communicates spike signals such that a baseline burst probability can be preset, avoiding the need for two phases. Addressing complex problems with these algorithms has been almost entirely limited to rate-based networks. Spiking proof-of-principles were done on the XOR problem.

### *2.2. Weight-Transport-Free Learning*

Approaches approximating the backpropagation of error algorithm face a problem known as the weight-transport problem. To accurately approximate the gradient, the feedback pathways must have synaptic weights that are symmetric to feedforward weights (symmetric feedback). Such direct weight-transport involves nonlocal communication weight values between synapses and must be avoided for learning in spiking neuromorphic hardware. Many solutions have been proposed [39, 40, 41, 42, 43, 10, 44, 45], including imposing random but static feedback connectivity (feedback alignment [39, 40]) and learning rules that seek to transport the weights by exploiting correlations in the activity patterns [42, 10, 44].

*2.2.1. Feedback Alignment Methods* It has been shown that feedback weights need not be initialized as symmetrical to feedforward weights, nor to be kept tightly symmetrical

throughout learning to effectively train networks using backpropagation [39]. As long as feedforward weights are initialized within 90° of feedback weights, feedforward weights trained with backpropagation tend to align with feedback weights, leading to accurate credit assignment and effective learning. Since feedforward synapses must align with feedback synapses, this method is referred to as feedback alignment (FA). Building upon this work, direct feedback alignment (DFA) uses random feedback weights directly connecting network outputs to each layer and has also been shown to be effective for learning with backpropagation [40]. While these methods are effective for training shallow, rate-based neural networks, their efficacy tends to decrease with increasing network depth [46, 47, 48].

The question of whether these techniques also scale to spiking neural networks has been partially addressed in Samadi et al. [41], where a spiking feedforward inference phase was combined with an analog learning phase that follows the backpropagation of error algorithm. Zenke and Ganguli [49] have similarly paired a spiking inference phased with analog, non-local, BPTT learning. Relatedly, Zhao et al. [50] have studied SNNs receiving direct but analog gradient information. Xing et al. [**?**] have used a signed spiking communication for the feedforward inference phase, but again paired with analog BPTT communication of gradients. In Payeur et al. (2021) [10] a fully spike-based communication of both feedforward and learning signals, but the feedback alignment approach was only applied to the XOR problem. The present work focuses on MNIST in a fully spiking communication of both feedforward and learning signals.

*2.2.2. Learned Feedback* Several methods implement a learning rule on feedback synapses, allowing them to align with feedforward weights. With the Kolen-Pollack method [51], weight symmetry is achieved by assigning the same plasticity rule to feedforward and feedback synapses and including a weight decay term. While this is effective in aligning weights, the necessity of weight decay may be a problem for some learning methods. Phaseless Alignment Learning (PAL) aligns feedback weights to feedforward weights by learning feedback weights from neuronal noise [44] outside of the presentation of examples. Another algorithm proposed by Burbank [52] uses temporally opposed STDP learning rules on feedforward and feedback synapses to successfully train a spiking autoencoder network. Similarly, Payeur et al. [10] provide a spike-based learning rule for learning the alignment of feedback synapses, which have not been tested on SNNs. These methods form a conceptual rationale for using implementing symmetry as a partially accurate bypass of the learned feedback method.

## 3. Methods

### *3.1. Neuron and Synapse Model*

*3.1.1. Somatic Compartment* Each neuron's soma follows Leaky Integrate and Fire (LIF) dynamics:

$$\tau_s \frac{dV_s(t)}{dt} = -(V_s(t) - E_L) + \frac{I_s(t)}{g_L} \tag{1}$$

where $V_s(t)$ represents the somatic membrane potential, $\tau_s$ the membrane time constant, $E_L$ the resting potential, $g_L$ the leak conductance, and $I_s(t)$ the input current. In our experiments, the somatic membrane time constant was set to $\tau_s = 10\,\text{ms}$. The neuron fires an event when $V_s(t)$ exceeds the threshold potential $V_{\text{threshold}}$, following which $V_s(t)$ resets to $V_{\text{reset}}$:

$$\text{if } V_s(t) \geq V_{\text{threshold}}, \text{ then } V_s(t) \to V_{\text{reset}}. \tag{2}$$

Rescaling to dimensionless variables , we define $\tilde{I}_s = I_s/g_L(V_{\text{threshold}} - E_L)$ and $\tilde{V}_s = V_s - E_L/(V_{\text{threshold}} - E_L)$, equation (1) becomes

$$\tau_s \frac{d\tilde{V}_s(t)}{dt} = -\tilde{V}_s(t) + \tilde{I}_s(t). \tag{3}$$

Every time the somatic voltage crosses the threshold, an event is emitted. This event can be either a single spike or a burst (see next section). Although in biology a burst means several spikes at a high frequency, here they are modelled as a marked point process, tagging every event with a type.

*3.1.2. Eligibility Trace* We use eligibility traces, $\tilde{E}_j(t)$, to indicate the eligibility of individual synapses connecting from a lower-order neuron $j$ to some other higher-order neuron:

$$\tau_{\tilde{E}} \frac{d\tilde{E}_j(t)}{dt} = -\tilde{E}_j(t) + S_j(t). \tag{4}$$

Here the event train $S_i(t) = \sum_k \delta(t - t_k^{(i)})$, where $t_k^{(i)}$ is the $k^{th}$ event time from neuron $i$. The eligibility trace time constant was to $\tau_{\tilde{E}} = 10\,\text{ms}$.

*3.1.3. Dendritic Compartment and Burst Generation* Dendrites come with different properties to the neuron's soma [53]. Here the dendritic compartment refers to an apical-like dendrite with a segregated membrane potential that can nonlinearly integrate feedback information to modulate neuron burst generation. It follows leaky integrator dynamics, using dimensionless inputs and membrane potential:

$$\tau_d \frac{d\tilde{V}_d(t)}{dt} = -\tilde{V}_d(t) + \tilde{I}_d(t). \tag{5}$$

where $\tilde{V}_d(t)$ represents the re-scaled dendritic membrane potential, $\tau_d$ is the dendritic membrane time constant, and $\tilde{I}_d$ is the re-scaled net dendritic input. We set the dendritic membrane time constant to $\tau_d = 10\,\text{ms}$.

There is no threshold in the dendritic compartment, instead, every time the soma emits an event, the state of the dendrite is used to determine if this event is a burst or a single spike. We consider two bursting models.

In the unsigned bursting model, the probability that an event is made into a burst is proportional to the dendritic potential $\tilde{V}_d$, we use a sigmoidal relationship $P_B(\tilde{V}_d(t)|\text{event at } t) = \sigma(\tilde{V}_d(t)) = \frac{1}{1+e^{-\tilde{V}_d(t)}}$.

In the signed bursting model, two types of bursts can be emitted: long bursts and short bursts. Either type of burst can only be generated upon the firing of an event from the soma. Given a somatic event, a burst is generated according to the symmetric function $P_B(\tilde{V}_d(t)|\text{event at } t) = \tanh(\text{abs}(\tilde{V}_d))$. If a burst is said to be generated, the sign of $V_d(t)$ is used to determine its type. If $\tilde{V}_d < 0$, a short burst is generated. If on the other hand $\tilde{V}_d > 0$, a long burst is generated. Because they represent the sign of the dendritic input, we refer to short and long bursts as negative and positive bursts, respectively. In this model, burst generation triggers a reset of the dendritic potential to the baseline, comparable to the somatic membrane potential reset upon a neuron spiking. This adds some refractoriness allowing the algorithm to better approximate backpropagation.

### *3.2. Network Architecture*

The SNNs consist of an input layer of 784 units, corresponding to the number of pixels in each sample image in the MNIST dataset. There are 1, 2, 3 or 4 hidden layers consisting of 100, 200, 400 or 800 hidden neurons as specified in the figures and associated text. The output layer of 10 neurons corresponds to the 10 classes of the MNIST dataset. Except for input units, each unit follows the two-compartment model described above. The network inputs (see Sect. 3.8), as well as any connection between hidden layers or from the hidden layer to the output layer follow the fully connected configuration. Weights are allowed to be positive or negative. The label corresponding to the network's class is used in calculating the network's output error and teaching signal (see Sect. 3.4.1).

### *3.3. Synaptic Communication and Multiplexing*

In our fully connected hierarchical SNNs, the credit-carrying information is communicated by multiplexing feedforward signals using events and feedback error signals using bursts. For the feedforward signals, we assume that synapses operate instantaneous changes in postsynaptic membrane potential upon every event. The input $I_s^{(i)}(t)$ to the soma of neuron $i$ is defined by the equation:

$$I_s^{(i)}(t) = \sum_j w_{ij} S_j(t). \tag{6}$$

In this equation, $w_{ij}$ is the feedforward synaptic weight from lower-order neuron $j$ to higher-order neuron $i$.

*3.3.1. Unsigned burst Communication* For the feedback signals, we again assume that synapses operate instantaneous changes in postsynaptic membrane potential, but upon every burst. We further enforce that these feedback connections target the apical

dendrite and therefore only affect the dendritic input $I_d^{(i)}(t)$ of a given neuron $i$. In the unsigned burst model, we have

$$I_d^{(i)}(t) = \sum_j b_{ij} B_j(t) + I_{\text{rec}}^{(i)}. \tag{7}$$

where $b_{ij}$ are the feedback synaptic weight from higher-order neuron $j$ to lower-order neuron $i$ and $B_j(t)$ is the burst train of neuron $j$. The additional current $I_{\text{rec}}$ appears due to recurrent inputs, see Sect. 3.4.2.

*3.3.2. Signed burst Communication* In the signed burst model, we assume the type of burst can dictate the sign of the post-synaptic effect:

$$I_d^{(i)}(t) = \sum_j b_{ij}(B_j^+(t) - B_j^-(t)) + I_{\text{rec}}^{(i)}. \tag{8}$$

We note that since the same feedback weight $b_{ij}$ operates on two distinct outputs of a neuron, this model requires weight transport.

*3.3.3. Multiplexing* In both signed and unsigned bursting models, when a neuron fires an event, the probability of that neuron sending a positive or negative error signal backward in the network is modulated by the dendritic potential of the neuron, $V_d$, as demonstrated in Figure 2. Similarly, the event rate is proportional to the integrated somatic input (proportional to $V_s$ if there were no reset). Thus the burst probability and the event rate represent two different signals simultaneously. Since each neuron integrates error signals from all higher-order neurons in its dendritic compartment and through the burst generation mechanism, the burst probability signal represents the signal that started with the input onto the apical dendrites at the top of the hierarchy and then backpropagates through it. This allows for the simultaneous online communication of credit-carrying error information from higher to lower-order neurons and feedforward, inference signals from lower to higher-order neurons.

### *3.4. Teaching Signals and Target Rates*

For dendritic inputs to be conceived as providing error signals, we inject a teaching signal at the top of the hierarchy (Sect. 3.4.1) and a layerwise regularisation signal (Sect. 3.4.2).

*3.4.1. Teaching Signals* Teaching signals are used for supervision, applying them recurrently to the dendritic compartments of output neurons such that $I_{\text{rec}}^{(i)}(t) = T_i(t)$ for these neurons. These signals are derived from the exponential moving average of the neuron's firing rate, $\bar{E}(t)$, using the same time constant as for the eligibility trace (see Sect. 3.1.2). The teaching signal, $T(t)$ is calculated as the derivative of the local error, $\varepsilon_T(\bar{E}(t), \hat{E})$, of the neuron's instantaneous firing rate, $\bar{E}(t)$ with respect to the neuron's

target rate, $\hat{E}$. For the neuron corresponding to the correct output class, a high target rate of $\hat{E} = 200\,\text{Hz}$ is set, and for all other neurons, a low rate of $\hat{E} = 20\,\text{Hz}$ is used. With this target, the teaching signal becomes

$$T_i(t) = \gamma_T \frac{\partial \varepsilon_T}{\partial \bar{E}(t)} = -2\gamma_T(\hat{E}_i(t) - \bar{E}_i(t)), \tag{9}$$

where $\epsilon_T$ corresponds to a square error loss. Here $\gamma_T$ controls the strength of the teaching signal. This differential in target rates will backpropagate and control synaptic plasticity through the network, ensuring the correct output neuron becomes more active while other output neurons are suppressed. We found that a teaching signal strength of $\gamma_T = 10^{-3}$ produced a strong enough teaching signal to steer plasticity while avoiding error signal saturation.

*3.4.2. Hidden target rates* Similar to the teaching signals applied to output neurons, hidden neurons receive recurrent regularization signals, such that $I_{\text{rec}}^{(i)}(t) = R_i(t)$ for all neurons except input neurons and output neurons. We use a uniformly low target rates $\hat{E} = 20\,\text{Hz}$:

$$R_i(t) = \gamma_R \frac{\partial \varepsilon_R}{\partial \bar{E}(t)} = -2\gamma_R(\hat{E} - \bar{E}_i(t)). \tag{10}$$

for $\epsilon_R$ a squared error loss. This $R_i(t)$ affects the apical dendrite of the same layer as $\bar{E}_i(t)$. We set $\gamma_R = 10^{-5} \ll \gamma_T$ such that the regularization does not suppress the error information due to the teaching signal. This approach regulates the firing rates of hidden neurons, thereby promoting sparsity and stability in the network's learning process.

### *3.5. Plasticity Rule*

Our algorithm utilizes a local spike-based learning rule to approximate backpropagation-based gradient descent. The synaptic weight changes are calculated as

$$\frac{dw_{ij}}{dt} = -\eta \tilde{E}_j(t)(B_i^+(t) - B_i^-(t)) \tag{11}$$

for the signed bursting model, and

$$\frac{dw_{ij}}{dt} = -\eta \tilde{E}_j(t)(B_i(t) - B_P S_i(t)) \tag{12}$$

for the unsigned bursting model. Here, $B_P$ is the baseline burst probability. The parameter $\eta$ represents the learning rate and was set to $\eta = 10^{-4}$ to achieve stable learning.

For the signed bursting model, synaptic weight changes, occur upon postsynaptic burst generation, with the sign of the weight change determined by the type of burst. For the unsigned bursting model negative weight changes occur upon postsynaptic spikes and positive weight changes occur upon postsynaptic bursts. In both models,

the magnitude of plasticity is proportional to the presynaptic neuron's eligibility trace and the postsynaptic event rate. Plasticity, then, depends on three factors: the activity of the presynaptic neuron (via the eligibility trace), the postsynaptic neuron's activity (triggering plasticity), and the integrated upstream error (via the burst fraction or burst type fraction). These three factors are essential to emulate the backpropagation of error algorithm, where weight adjustments of a given unit depend on three factors: the incoming activity, the derivative of the unit's activity with respect to the sum of its inputs, and the integrated downstream error.

It is worth noting the distinct behaviour of signed and unsigned conditions in the absence of any postsynaptic error ($\tilde{V}_d(t) = 0$). For the unsigned burst model, this condition results in a burst probability of 0.5. Because $\tilde{V}_d = 0$ corresponds to the absence of dendritic input, we call the resulting output burst probability the baseline burst probability. In the associated plasticity rule, the parameter, $B_P = 0.5$, is chosen to match this baseline burst probability, such that there is a balance between positive and negative weight changes. We note that although this ensures that $\Delta w = 0$ on average when $\tilde{V}_d(t) = 0$, the fluctuations around this can cause drifts in the learned state over time. For the signed burst model, plasticity is more stable because the probability of burst generation is 0 when $\tilde{V}_d(t) = 0$ so no plasticity can occur.

### *3.6. Low-Resolution Weights*

To implement low-resolution synaptic weights, $N^2 - 1$ evenly spaced possible weight values were determined ranging from -1 and 1, where $N$ is the weight resolution in bits. The magnitude of the interval between possible weight values is referred to as $\epsilon$. This low resolution was imposed on initial weights and required an adaptation of the plasticity rule.

Initial weights (see Sect. 3.7) were rounded to the nearest possible weight value. During learning, an additional plasticity accumulator variable is added for each synapse. Plasticity is calculated according to the learning rule and added to the accumulator variable. Once a synaptic accumulator reaches $\pm 0.5\epsilon$, the synaptic weight potentiates or depresses by $\epsilon$ and the accumulator variable weight resets. In this way, the network can learn with low-resolution weights while still having an effective learning rate of less than $\epsilon$.

### *3.7. Weight Initialization*

Feedforward synaptic weights were initialized from a normal distribution with standard deviation proportional to the square root of the number of presynaptic neurons to ensure stable learning [54]. It was found that to ensure error signals do not saturate, feedback weights had to be scaled by the number of higher-order neurons divided by the number of lower-order neurons in the layers connected by these weights. For the symmetric feedback implementation, feedback weights are aligned to feedforward weights. For FA and DFA, feedback weights were initialized from a normal distribution with the standard

deviation scaled according to the number of higher-order neurons divided by the number of lower-order neurons while ensuring initial feedback weight matrices are within 90° of feedforward weights, as is done in [39].

### *3.8. Input Encoding*

In our study, we encoded the MNIST Digit Classification Dataset images for a spiking neural network. The dataset consists of 60,000 training and 10,000 test grayscale images, each 28x28 pixels [55]. The images were normalized so that the average pixel intensity of each image was constant. Pixel intensities, ranging from 0 to 1, were linearly mapped to firing rates between 20 Hz and 200 Hz following a Poisson process. Accordingly, each pixel in each image was converted into an input unit spike train of a duration of 100 ms.

### *3.9. Simulation Methods*

We simulated the SNNs using a CPU with Python. Neuron dynamics were simulated using the forward Euler method with a 1 ms time step. For each time step, the network was updated from the input layer to the output layer. Feedforward inputs to a layer were taken from the lower-order outputs at the same time step and feedback inputs to a layer were taken from the higher-order outputs at the previous time step.

*3.9.1. Training* During training, samples from the MNIST dataset are presented to the network sequentially. Teaching signals are applied continuously, and plasticity is engaged throughout the 100 ms. Neuronal state variables ($\tilde{V}_d$ and $\tilde{V}_s$), eligibility traces ($\tilde{E}$) and rate estimates ($\bar{E}$) are reset to zero at the end of each sample presentation.

*3.9.2. Testing* During testing, test samples are presented the same way as in the training phase, but there is no inputs to the dendrites and plasticity is off. The most active output neuron over the sample duration decides the network's predicted class.

### *3.10. Backpropagation-Through-Time*

We compare our results with a backpropagation-through-time method using surrogate gradients [21]. We use the sigmoid function to approximate the spike discontinuity in the backward pass. The neurons follow the same dynamics, with the same parameters as the somatic neuron compartments previously described. We trained the network on the same input encoding, with each sample presented also for 100 ms and used a learning rate of $\eta = 2 \times 10^{-4}$ to balance stability and speed of learning.

## 4. Results

We trained SNNs with Burst-CCN (Burstprop, for short [10, 11]) on a rate-coded version of the MNIST dataset. The algorithm uses two-compartment neuron models with LIF

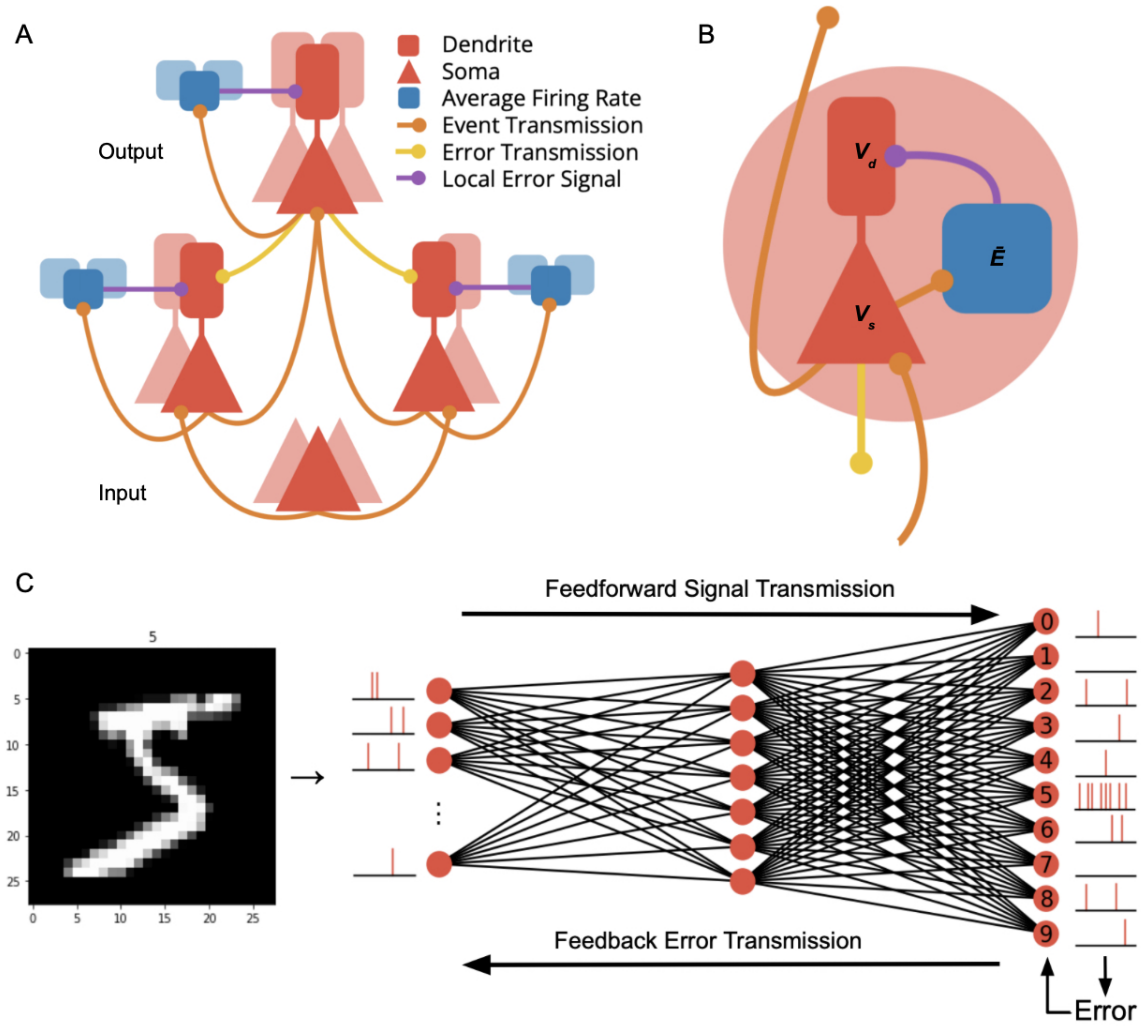


**Figure 1. Overview of Burstprop connectivity and signal transmission:** A. Schematic of the core spiking neural network architecture. Note that feedforward inputs target somas and feedback inputs target feedback. B. Schematic of the core unit. Neuron receives feedforward inputs to its somatic compartment, transmits events to higher-order neurons in the network and bursts/events to lower-order neurons in the network. The neuron's local error is calculated from its average firing rate and target rate and the error signal applied to the neuron's apical dendrite, combined with error signals received from the higher-order neurons. D. Schematic of the network spiking neural network used in the MNIST classification task. A sample image is converted to input Poisson spike trains which are propagated forward through the network. Each red circle corresponds to a neuron with a soma, a dendrite and a self connection as in panel B. The output layer provides the prediction according to the highest firing neuron. An error is continuously calculated and fed back to the output layer. Black lines are meant to indicate the presence of a connection in both directions, according to panel A.

soma and dendrite-dependent bursting implemented as a marked point process. Soma-related events are sent to somata up the hierarchy while dendrite-dependent bursts

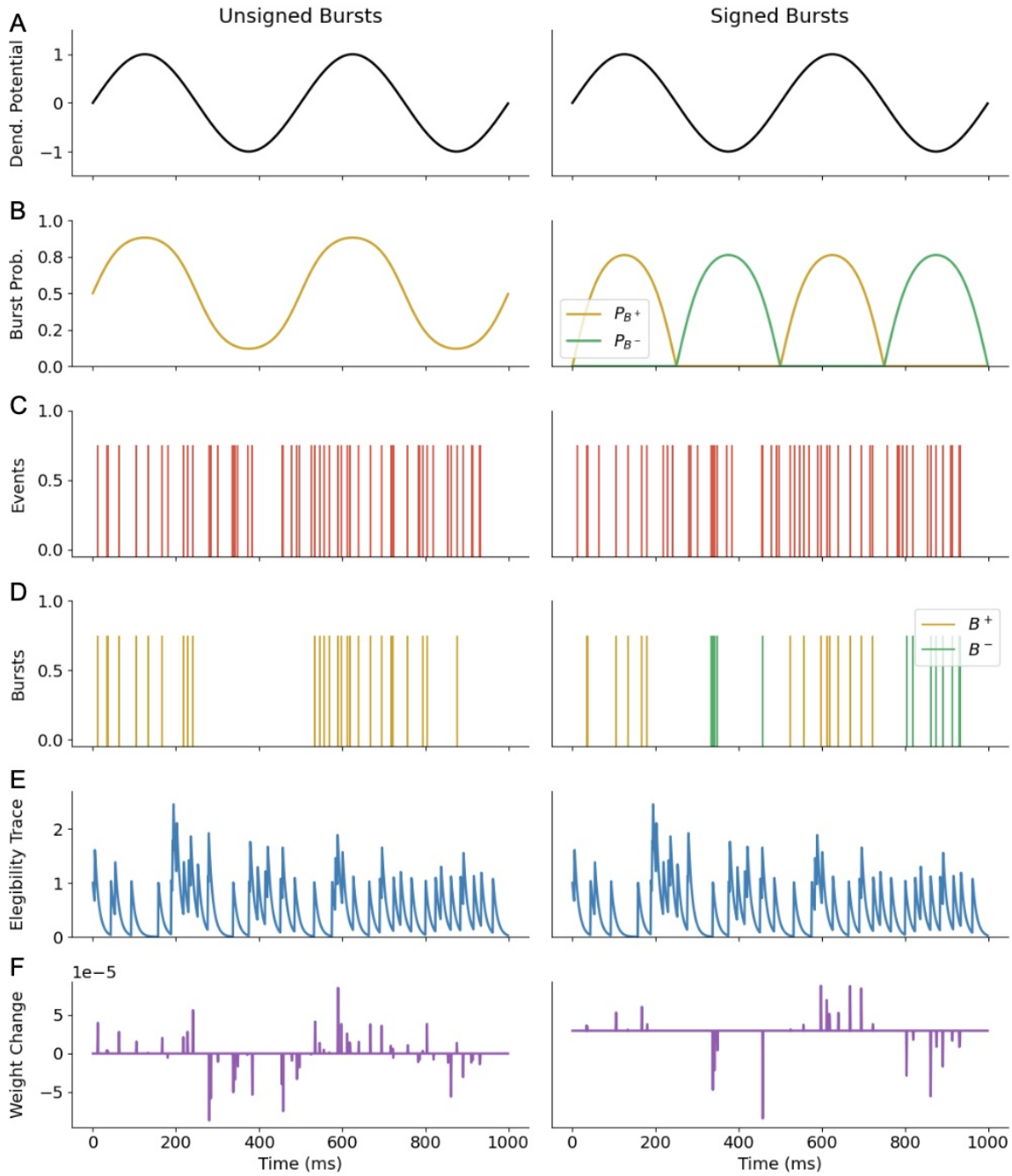


**Figure 2. Dendritic potential steers direction of plasticity for synapses to neurons.** A. Time-varying dendritic potential following a sine curve. B. Burst probabilities. Unsigned Burstprop, $P_B = \sigma(V_d)$. Signed Burstprop, $P_{B^+} = \max(0, \tanh(V_d))$ and $P_{B^-} = \max(0, -\tanh(V_d))$. C. Poisson spike train, with 50 Hz firing rate. D. Burst train generated from spike train and burst probability. E. Eligibility trace of a presynaptic neuron. F. Resultant synaptic plasticity.

are sent to dendrites down the hierarchy, thus multiplexing feedforward signals with feedback signals. Feedback signals are made credit-carrying by letting the dendrites of output units receive the classification error. Plasticity follows a local, burst-dependent

rule chosen to approximate the backpropagation of error algorithm.

While Burstprop uses increases and decreases of burst probability to communicate positive and negative signs of the error signals (Fig. 2), here we also considered an alternative model where the different sign of the error signal triggers different types of bursts. Physiologically, this 'Signed Burst' model could correspond to bursts of different durations engaging different short-term plasticity [35] and long-term plasticity. By construction, the signed burst model is more stable and less subject to drifts in plasticity than the original Burstprop model.

### *4.1. Deep Learning on MNIST Dataset*

To test the performance of our SNNs, we have trained both signed and unsigned versions of networks using symmetric feedback, varying width and depth on the rate encoded MNIST data (Fig. 3). We found by inspecting the resulting test-set errors that both signed and unsigned models were able to effectively train spiking neural networks with one layer with the signed model achieving $97.2 \pm 0.1\%$ and $96.8 \pm 0.1\%$ test classification accuracy respectively on a network with a single hidden layer of 100 neurons. For deeper networks, the signed model outperformed the unsigned one (Fig. 3A1), with the signed model achieving $95.9 \pm 0.1\%$ accuracy compared to $64 \pm 1\%$ accuracy for the unsigned model on a 4 hidden layer network. We did not observe an increase but rather a decrease in performance with the number of hidden layers. While there was no performance gain with depth, increased width was associated with an increase in performance (Fig. 3A2). Our algorithms could thus successfully train SNNs to perform MNIST with a performance that increases with layer width. Furthermore, we conclude that the signed burst model helps the algorithm to remain successful in deep networks.

To contextualize the learning ability, we compared the test-set prediction accuracy of our local learning algorithm to a state-of-the-art, non-local, BPTT-based learning method [21]. We found that in shallow networks, BPTT training applied on the same SNN architecture provided slightly better performance, with $97.7 \pm 0.1\%$ accuracy on a network with a single hidden layer of 100 neurons. This improvement was maintained across the different network widths (Fig. 3A2). Networks trained with BPTT did not show a decrease in performance with network depth ($84 \pm 1\%$ accuracy on a 4 hidden layer network), an observation that echoes previous tests [54]. Because the choice of hyperparameters is likely to influence the comparison of these three algorithms, we conservatively conclude that SNNs with local burst-dependent plasticity rules can achieve good performance on MNIST, even with a small number of neurons and with multiple layers.

That the network performance does not increase with network depth may suggest that the algorithm is only training the output weights. To investigate this, and to determine whether the algorithm is effectively assigning credit to earlier layers, we examine how freezing either the output weights or the input weights affects learning in a single hidden layer network. If learning were limited to the output layer, freezing

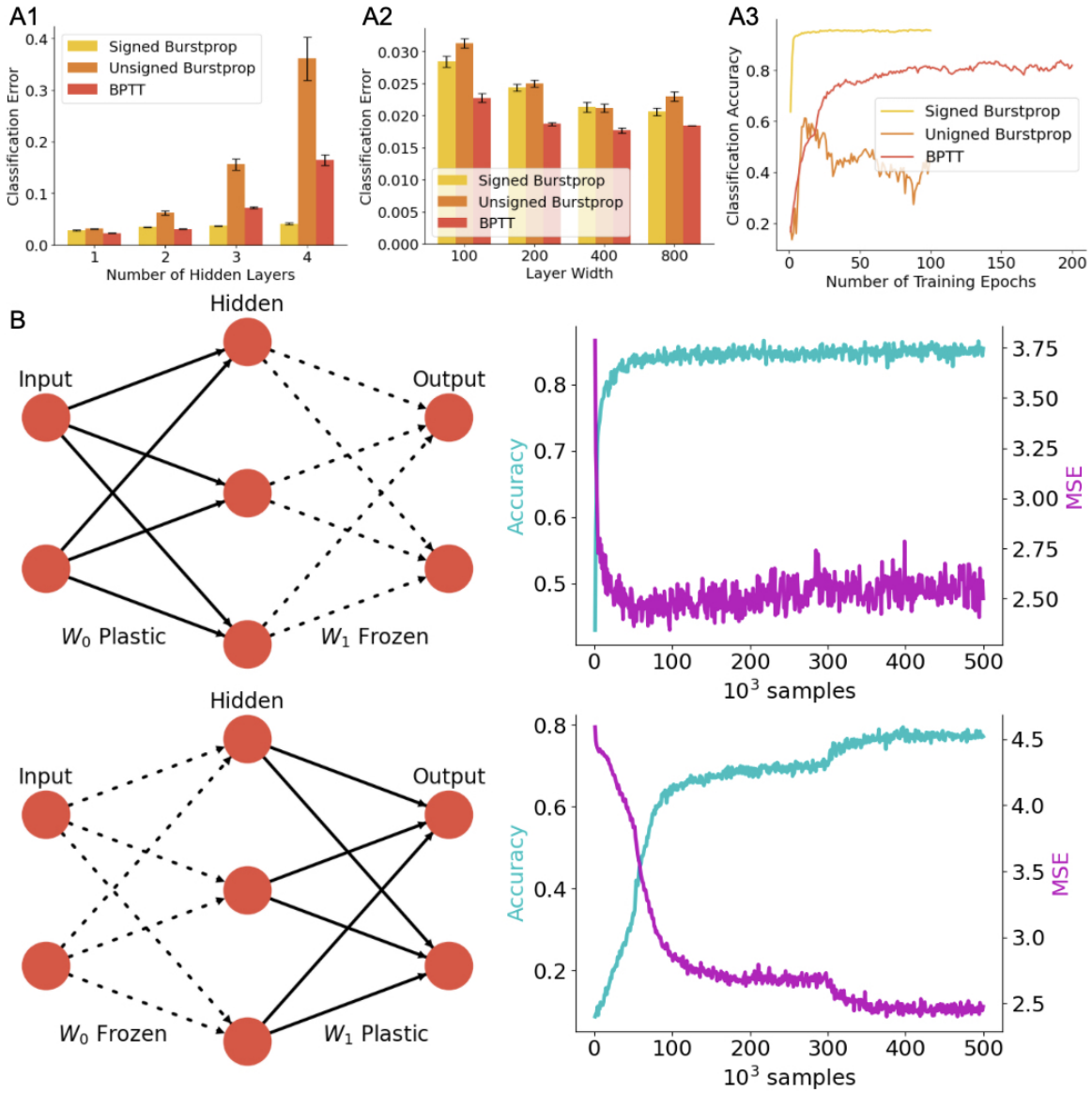


**Figure 3. Burstprop coordinates multilayer learning to solve MNIST.** A. Comparing test classification error on spiking MNIST dataset for Signed Burstprop, Unsigned Burstprop, and BPTT with symmetric feedback for networks with an increasing number of hidden layers, 100 neurons per layer (A1), and networks with increasing layer width, 1 hidden layer (A2). Burstprop algorithms were trained for 100 epochs and BPTT was trained for 200 epochs. A3. Example learning curves for each algorithm on a network of 4 hidden layers, each with 100 neurons. Error bars represent the standard deviation of minimum test error for 10 simulations. B. Signed Burstprop test accuracy and mean square error (MSE) per $10^3$ samples of a single hidden layer neural network of 100 hidden neurons. Row 1: $W_0$ learns and $W_1$ is frozen. Row 2: $W_1$ learns and $W_0$ is frozen.

plasticity in this layer should result in the network being unable to learn. Instead, what we see is that the network remains able to learn effectively when the output layer is frozen (Fig. 3B). To take another approach, forcing the plasticity to only

occur in the output weights should preserve the performance if credit signals were not reaching the lower-order connection. We found that freezing the output weights has a smaller effect on learning capabilities than freezing the input weights. Freezing the input weights made learning much slower (Fig. 3). Together, our burst-dependent SNN can successfully propagate credit-carrying information to neurons and accurately steer plasticity beyond the output layer. This suggests that our algorithm is effectively approximating backpropagation in the way it assigns errors.

### *4.2. Feedback Alignment*

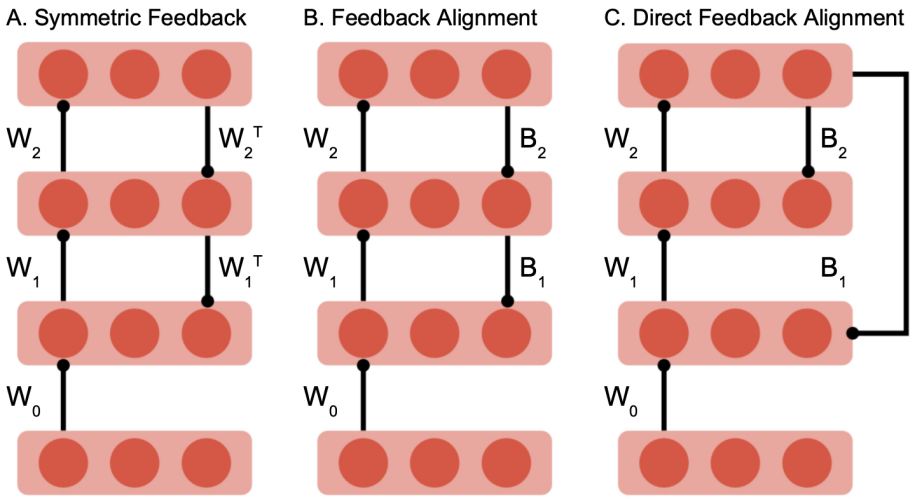


**Figure 4. Feedback Connectivity Schematics:** A. Symmetric weight connectivity: the strength of feedback synapses connecting a higher-order neuron to a lower-order neuron to higher-order neurons are proportional to the synapses connecting lower-order neurons to higher-order neurons. B. Feedback alignment: feedback synapses are static and randomly initialized within 90° of feedforward synapses. C. Direct feedback alignment: Feedback synapses are static and randomly initialized, connecting output neurons to all hidden neurons.

While we have shown that Burstprop can successfully train an SNN on MNIST, we did this by enforcing symmetry between feedforward and feedback weights. To allow for neuromorphic learning, and to address biological plausibility, our algorithm must remain effective with weight-transport-free learning. We thus tested the performance of Burstprop on our spiking MNIST dataset using feedback alignment (FA) and direct feedback alignment (DFA), two algorithms that avoid the weight-transport problem by using random but fixed feedback weights (see Methods and Figure 4). If Burstprop can learn from the data while using random but static feedback weights, this would suggest that weight transport is not necessary for learning in SNNs. We found that both signed (Figure 5B1) and unsigned (Figure 5B2) versions of the algorithm can achieve good performance, achieving $95.6 \pm 0.1\%$ accuracy and $85 \pm 1\%$ accuracy respectively on a 4 hidden layer network with feedback alignment. The decrease in performance with the

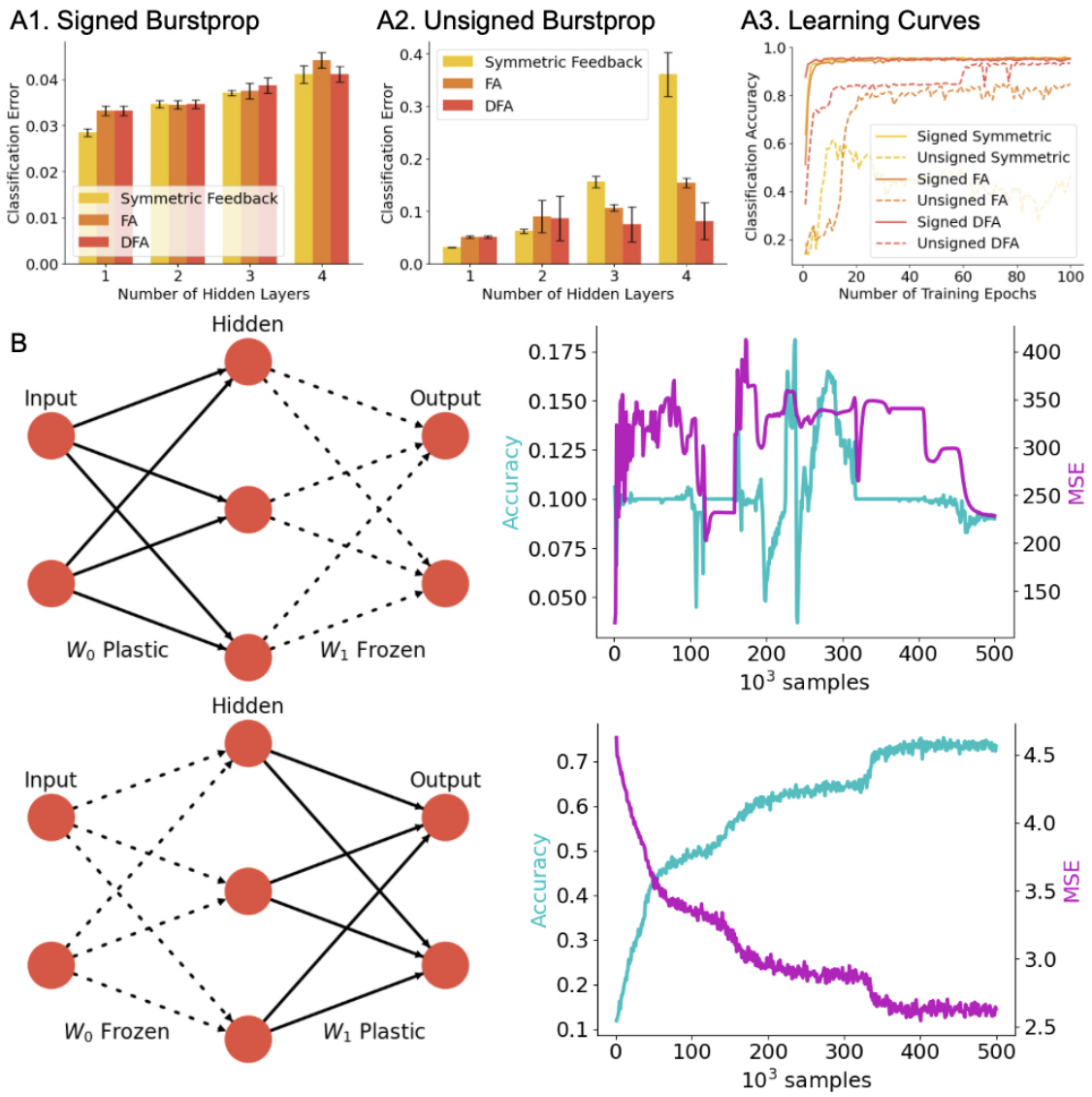


**Figure 5. Burstprop learns with feedback alignment methods:** A. Comparing test classification error on spiking MNIST dataset for different feedback error transportation methods with the Signed Burstprop (B1) and Unsigned Burstprop (B2) algorithms with increasing number of hidden neurons, 100 neurons per layer. Each trained for 100 epochs. A3. Example learning curves for each algorithm on a network of 4 hidden layers, each with 100 neurons. B. Signed Burstprop with FA test accuracy and mean square error (MSE) per $10^3$ samples of a single hidden layer neural network of 100 hidden neurons. Row 1: $W_0$ learns and $W_1$ is frozen. Row 2: $W_1$ learns and $W_0$ is frozen.

number of layers was lessened by switching to FA for the unsigned burst model but unaltered for the signed burst model.

Next, we again investigated how freezing either the first or second layer impacts learning. In principle, learning the set of weights should not be possible if the higher-order layers have not had a chance to learn and align with the feedback [39]. Accordingly,

if we freeze the output weights, learning should be strongly affected. This is indeed what we see (Fig. 5B), whereby freezing the output weights in an FA scenario completely blocked learning. Thus using random but fixed feedback weights can implement weight-transport-free learning in SNNs using local learning rules.

### *4.3. Low-Resolution Synaptic Weights*

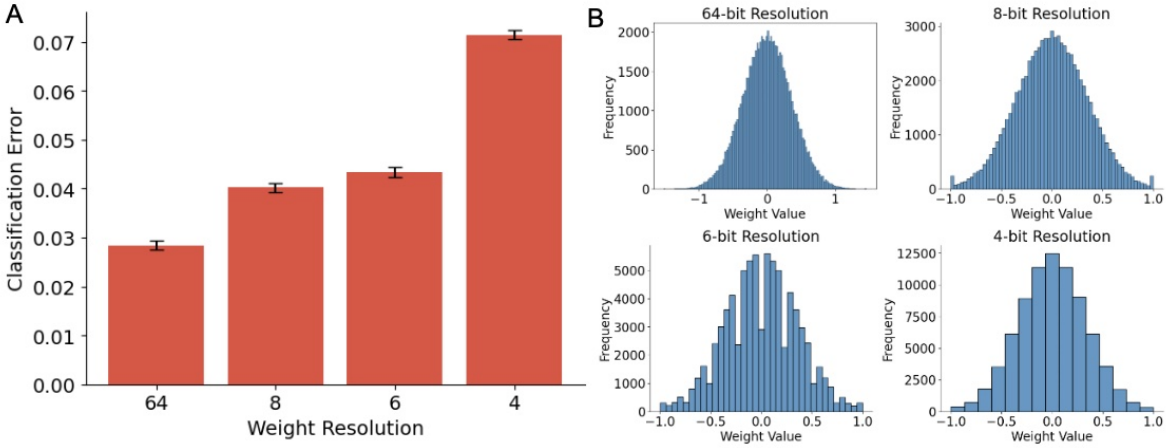


**Figure 6. Burstprop is robust to low-resolution synaptic weights.** A. Comparing test classification error on spiking MNIST dataset for networks of a single hidden layer of 100 neurons, trained with Signed Burstprop, with varying weight resolutions. B. Initial weight distributions of input weights for varying weight resolutions.

An important consideration for implementing SNNs in neuromorphic hardware is the number of distinct values that synaptic weights can take [56]. Spiking neuromorphic hardware may only be able to support small bit resolutions for their weights, in the range of 4 to 8 bits [56]. Here we investigate the impact of decreased synaptic weights on learning performance. Figure 6 shows that learning performance decreases with weight resolution. Still, even with only 4-bit resolution, learning is achieved with over 90% classification accuracy. This result indicates that Burstprop is robust to low bit resolution in the synaptic weights.

## 5. Discussion

We have implemented a spiking version of the Burstprop algorithm that uses local learning rules and marked point processes for differential communication to different post-synaptic targets. Our algorithm improved on the previous efforts solving XOR with over a thousand neurons to solving a more complex task with an order of 100 neurons. We found that our SNNS could achieve a performance similar to that of BPTT. By experimenting with freezing weight layers during learning, we conclude that the algorithm can accurately perform credit assignment beyond the output layer.

The experiments training the algorithm with FA and DFA show that Burstprop can successfully perform weight-transport-free learning. The results also show the algorithm is robust to low-bit resolution synaptic weights. These results indicate that burst-dependent communication and plasticity can form an effective, biologically inspired, local learning method for SNNs that may be useful for neuromorphic learning.

We have shown that feedback alignment methods can be used effectively with SNNs using Burstprop to solve weight transport between feedforward and feedback weights. Our algorithm, however, requires some symmetry between two sets of feedback weights. Biological neurons would need two different synapses to communicate different firing patterns. These symmetric pathways are required to transmit positive and negative error signals. In our experiments, we have been enforcing this symmetry through weight transport, however, many methods could be implemented to align these weights. Further work must be done to find a suitable solution to address this problem. This problem is addressed in previous work [11] by adding a learning rule to feedback synapses, ensuring they converge.

Another weakness of our algorithm is that its fitting capabilities were not improved by increasing network depth. This is common for spiking neural networks [54, 50] and it is unclear whether this is related to the input encoding or could be a fundamental limitation of the algorithm. While neural networks often rely on deep structures to fit complex functions, even a single-layer spiking neural network is a universal function approximator so width could be used to compensate for depth in some applications.

An important limitation of our results worth mentioning is that we limit our experimentation to the MNIST dataset. MNIST is a fairly simple benchmark problem for supervised learning algorithms and many non-neural network algorithms reach high test accuracy on this task. For instance, a linear classifier can achieve 7.6% error [55], a random forest algorithm can achieve 2.4% error, and an SVM can achieve 0.56% error [57]. While our specific spiking version of the dataset is not as simple of a problem, it is still solving the same underlying problem. The goal of this work was not to achieve state-of-the-art performance on this benchmark, but to show that Burstprop can be applied to solve a complex learning task. Further experimentation with more complex problems is required to see how that algorithm scales. Our results open the door for the principles of this algorithm to be used in a hardware implementation of neuromorphic learning.

## References

[1] Catherine D. Schuman, Shruti R. Kulkarni, Maryam Parsa, J. Parker Mitchell, Prasanna Date, and Bill Kay. Opportunities for neuromorphic computing algorithms and applications. *Nature Computational Science*, 2:10–19, January 2022.

[2] Friedemann Zenke, Sander M Bohté, Claudia Clopath, Iulia M Comşa, Julian Göltz, Wolfgang Maass, Timothée Masquelier, Richard Naud, Emre O Neftci, Mihai A Petrovici, et al. Visualizing a joint future of neuroscience and neuromorphic engineering. *Neuron*, 109(4):571–575, 2021.

[3] Man Yao, Ole Richter, Guangshe Zhao, Ning Qiao, Yannan Xing, Dingheng Wang, Tianxiang Hu, Wei Fang, Tugba Demirci, Michele De Marchi, et al. Spike-based dynamic computing with asynchronous sensing-computing neuromorphic chip. *Nature Communications*, 15(1):4464, 2024.

[4] Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, 2019.

[5] Guillermo Gallego, Tobi Delbrück, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew J Davison, Jörg Conradt, Kostas Daniilidis, et al. Event-based vision: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(1):154–180, 2020.

[6] Navnidhi K Upadhyay, Hao Jiang, Zhongrui Wang, Shiva Asapu, Qiangfei Xia, and J Joshua Yang. Emerging memory devices for neuromorphic computing. *Advanced Materials Technologies*, 4(4):1800589, 2019.

[7] Giacomo Indiveri and Shih-Chii Liu. Memory and information processing in neuromorphic systems. *Proceedings of the IEEE*, 103(8):1379–1397, 2015.

[8] Benjamin Cramer, Sebastian Billaudelle, Simeon Kanya, Aron Leibfried, Andreas Grübl, Vitali Karasenko, Christian Pehle, Korbinian Schreiber, Yannik Stradmann, Johannes Weis, et al. Surrogate gradients for analog neuromorphic computing. *Proceedings of the National Academy of Sciences*, 119(4):e2109194119, 2022.

[9] Pierre Baldi and Peter Sadowski. A theory of local learning, the learning channel, and the optimality of backpropagation. *Neural Networks*, 83:51–74, November 2016.

[10] Alexandre Payeur, Jordan Guerguiev, Friedemann Zenke, Blake A Richards, and Richard Naud. Burst-dependent synaptic plasticity can coordinate learning in hierarchical circuits. *Nature Neuroscience*, pages 1–10, 2021.

[11] Will Greedy, Heng Wei Zhu, Joseph Pemberton, Jack Mellor, and Rui Ponte Costa. Single-phase deep learning in cortico-cortical networks. *Advances in Neural Information Processing Systems*, 35, 2022.

[12] Johannes J. Letzkus, Bjorn M. Kampa, and Greg J. Stuart. Learning rules for spike timing-dependent plasticity depend on dendritic synapse location. *J. Neurosci.*, 26(41):10420–10429, October 2006.

[13] Katie C Bittner, Aaron D Milstein, Christine Grienberger, Sandro Romani, and Jeffrey C Magee. Behavioral time scale synaptic plasticity underlies ca1 place fields. *Science*, 357(6355):1033–1036, 2017.

[14] Lea Caya-Bissonnette, Richard Naud, and Jean-Claude Beique. Cellular substrate of eligibility traces during behavioral timescale synaptic plasticity. *BioRxiv*, 2023.

[15] Zachary Friedenberger, Emerson Harkin, Katalin Tóth, and Richard Naud. Silences, spikes and bursts: Three-part knot of the neural code. *The Journal of Physiology*, 601(23):5165–5193, 2023.

[16] Richard Naud and Henning Sprekeler. Sparse bursts optimize information transmission in a multiplexed neural code. *Proceedings of the National Academy of Sciences*, 115(27):E6329–E6338, June 2018.

[17] Ruohong Zhou. A method of converting ann to snn for image classification. In *2023 IEEE 3rd International Conference on Neuromorphic Computing (ICNC)*. IEEE, 2023.

[18] Haoran Gao, Junxian He, Haibing Wang, Tengxiao Wang, Zhengqing Zhong, Jianyi Yu, Ying Wang, Min Tian, and Cong Shi. High-accuracy deep ann-to-snn conversion using quantization-aware training framework and calcium-gated bipolar leaky integrate and fire neuron. *Frontiers in Neuroscience*, 17, March 2023.

[19] Eric Hunsberger and Chris Eliasmith. Training spiking deep networks for neuromorphic hardware. *arXiv preprint arXiv:1611.05141*, 2016.

[20] Sebastian Schmitt, Johann Klähn, Guillaume Bellec, Andreas Grübl, Maurice Guettler, Andreas Hartel, Stephan Hartmann, Dan Husmann, Kai Husmann, Sebastian Jeltsch, et al.

Neuromorphic hardware in the loop: Training a deep spiking network on the brainscales wafer-scale system. In *2017 international joint conference on neural networks (IJCNN)*, pages 2227–2234. IEEE, 2017.

[21] Emre O. Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks. *arXiv preprint arXiv:1901.09948*, 2019.

[22] Sumit Bam Shrestha and Garrick Orchard. Slayer: Spike layer error reassignment in time. *arXiv preprint arXiv:1810.08646*, 2018.

[23] Benjamin Ellenberger, Paul Haider, Jakob Jordan, Kevin Max, Ismael Jaras, Laura Kriener, Federico Benitez, and Mihai A. Petrovici. Backpropagation through space, time, and the brain. *arXiv preprint arXiv:2403.16933*, March 2024.

[24] Bernhard Nessler David Kappel and Wolfgang Maass. Stdp installs in winner-take-all circuits an online approximation to hidden markov model learning. *PLoS Computational Biology*, 10(3):e1003511, March 2014.

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

[26] Robert Legenstein, Dejan Pecevski, and Wolfgang Maass. A learning theory for reward-modulated spike-timing-dependent plasticity with application to biofeedback. *PLOS Comput. Biol.*, 4:e1000180, 2008.

[27] R. V. Florian. Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity. *Neural Computation*, 19:1468–1502, 2007.

[28] Nicolas Frémaux, Henning Sprekeler, and Wulfram Gerstner. Functional requirements for reward-modulated spike-timing-dependent plasticity. *J Neurosci*, 30(40):13326–13337, 2010.

[29] Konrad P Körding and Peter König. Supervised and unsupervised learning with two sites of synaptic integration. *J Comput. Neurosci.*, 11(3):207–215, 2001.

[30] João Sacramento, Rui Ponte Costa, Yoshua Bengio, and Walter Senn. Dendritic cortical microcircuits approximate the backpropagation algorithm. In *Advances in Neural Information Processing Systems*, pages 8721–8732, 2018.

[31] Jordan Guergiuev, Timothy P Lillicrap, and Blake A Richards. Towards deep learning with segregated dendrites. *eLife*, 2017.

[32] Walter Senn, Dominik Dold, Akos F Kungl, Benjamin Ellenberger, Jakob Jordan, Yoshua Bengio, Joao Sacramento, and Mihai A Petrovici. A neuronal least-action principle for real-time learning in cortical circuits. *bioRxiv*, pages 2023–03, 2023.

[33] Alex Reyes, Rafael Lujan, Andrej Rozov, Nail Burnashev, Peter Somogyi, and Bert Sakmann. Target-cell-specific facilitation and depression in neocortical circuits. *Nat. Neurosci.*, 1(4):279–285, 1998.

[34] Massimo Scanziani, Beat H Gähwiler, and Serge Charpak. Target cell-specific modulation of transmitter release at terminals from a single axon. *Proceedings of the National Academy of Sciences*, 95(20):12004–12009, 1998.

[35] John Beninger, Julian Rossbroich, Katalin Toth, and Richard Naud. Functional subtypes of synaptic dynamics in mouse and human. *Cell Reports*, 2024.

[36] ME Larkum, J Zhu, and B Sakmann. A new cellular mechanism for coupling inputs arriving at different cortical layers. *Nature*, 398:338–341, 1999.

[37] Zachary Friedenberger and Richard Naud. Dendritic excitability primarily controls overdispersion. *bioRxiv*, pages 2022–11, 2022.

[38] Mike Stuck and Richard Naud. Burstprop for learning in spiking neuromorphic hardware. In *Proceedings of the 2023 International Conference on Neuromorphic Systems (ICONS)*, pages 1–5, August 2023.

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

[40] Arild Nøkland. Direct feedback alignment provides learning in deep neural networks. *Advances*

*in neural information processing systems*, 29, 2016.

[41] Arash Samadi, Timothy P Lillicrap, and Douglas B Tweed. Deep learning with dynamic spiking neurons and fixed feedback weights. *Neural Comput.*, 29(3):578–602, 2017.

[42] Mohamed Akrout, Collin Wilson, Peter Humphreys, Timothy Lillicrap, and Douglas B Tweed. Deep learning without weight transport. *Advances in neural information processing systems*, 32, 2019.

[43] Li Ji-An and Marcus K Benna. Deep learning without weight symmetry. *arXiv preprint arXiv:2405.20594*, 2024.

[44] Kevin Max, Laura Kriener, Garibaldi Pineda García, Thomas Nowotny, Ismael Jaras, Walter Senn, and Mihai A. Petrovici. Learning efficient backprojections across cortical hierarchies in real time. *arXiv preprint arXiv:2212.10249*, February 2024.

[45] Bill Podlaski and Christian K Machens. Biological credit assignment through dynamic inversion of feedforward networks. *Advances in Neural Information Processing Systems*, 33:10065–10076, 2020.

[46] Sergey Bartunov, Adam Santoro, Blake Richards, Luke Marris, Geoffrey E Hinton, and Timothy Lillicrap. Assessing the scalability of biologically-motivated deep learning algorithms and architectures. *Advances in neural information processing systems*, 31, 2018.

[47] Theodore H Moskovitz, Ashok Litwin-Kumar, and LF Abbott. Feedback alignment in deep convolutional networks. *arXiv preprint arXiv:1812.06488*, 2018.

[48] Albert Jiménez Sanfiz and Mohamed Akrout. Benchmarking the accuracy and robustness of feedback alignment algorithms. *arXiv preprint arXiv:2108.13446*, 2021.

[49] Friedemann Zenke and Surya Ganguli. Superspike: Supervised learning in multilayer spiking neural networks. *Neural Comput.*, 30(6):1514–1541, 2018.

[50] Dongcheng Zhao, Yi Zeng, Tielin Zhang, Mengting Shi, and Feifei Zhao. Glsnn: A multi-layer spiking neural network based on global feedback alignment and local stdp plasticity. *Frontiers in Computational Neuroscience*, 14, November 2020.

[51] J.F. Kolen and J.B. Pollack. Backpropagation without weight transport. In *Proceedings of the 1994 IEEE International Conference on Neural Networks (ICNN)*, pages 1375–1380. IEEE, 1994.

[52] Kendra S. Burbank. Mirrored stdp implements autoencoder learning in a network of spiking neurons. *PLOS Computational Biology*, 11(12):e1004566, December 2015.

[53] Alexandre Payeur, Jean-Claude Béïque, and Richard Naud. Classes of dendritic information processing. *Current Opinion in Neurobiology*, 58:78–85, October 2019.

[54] Julian Rossbroich, Julia Gygax, and Friedemann Zenke. Fluctuation-driven initialization for spiking neural network training. *arXiv preprint arXiv:2206.10226*, 2022.

[55] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[56] Lorenz K. Müller and Giacomo Indiveri. Rounding methods for neural networks with low resolution synaptic weights. *arXiv preprint arXiv:1504.05767*, April 2015.

[57] Yang Liu. The image classification of mnist dataset by using machine learning techniques. *UC Merced Previously Published Works*, December 2021.