ACCELERATING SPIKING NEURAL NETWORK TRAINING USING THE d-BLOCK MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

There is a growing interest in using spiking neural networks (SNNs) to study the brain *in silico* and in emulating them on neuromorphic computers due to their lower energy consumption compared to artificial neural networks (ANNs). Significant progress has been made in directly training SNNs to perform on a par with ANNs in terms of accuracy. However, these methods are slow due to their sequential nature and require careful network regularisation to avoid overfitting. We propose a new SNN model, the d-block model, with stochastic absolute refractory periods and recurrent conductance latencies, which reduces the number of sequential computations using fast vectorised operations. Our model obtains accelerated training speeds and state-of-the-art performance across various neuromorphic datasets without the need for any regularisation and using fewer spikes compared to standard SNNs.

1 INTRODUCTION

Artificial neural Networks (ANNs) are ubiquitous in achieving state-of-the-art performance across various domains, such as image recognition (He et al., 2016), natural language processing (NLP) (Brown et al., 2020) and computer games (Silver et al., 2017; Vinyals et al., 2019). These networks have also proven useful for studying the brain due to their architectural similarities (Richards et al., 2019) and have further advanced our understanding of the computational processes underlying the visual and auditory system (Harper et al., 2016; Singer et al., 2018; Cadena et al., 2019; Francl & McDermott, 2022; Yamins & DiCarlo, 2016). However, ANNs have been criticised for their substantial energy demands resulting from their continued exponential growth in size (Strubell et al., 2019; Schwartz et al., 2020), as exemplified by the GPT language models scaling from 110 million to 1.5 billion to 175 billion parameters to deliver ever-improving advances across various NLP tasks (Radford et al., 2018; 2019; Brown et al., 2020). Furthermore, the applicability of ANNs to neuroscience is confined by their activation function, as the brain employs spikes rather than continuous-valued outputs used by ANN units.

Spiking neural networks (SNNs) are a type of binary neural network (Figure 1a), which overcome these challenges as they consume drastically less energy than ANNs when deployed on neuromorphic hardware (Wunderlich et al., 2019) and their biological realism makes them a favourable model for studying the brain *in silico* (Vogels et al., 2011; Denève & Machens, 2016). However, SNN training remains a challenging problem due to the non-differentiable binary activation function employed by the spiking neurons. This has historically resulted in solutions that impose constraints on the neurons, such as rate codes (O'Connor et al., 2013; Esser et al., 2015; Rueckauer et al., 2016), or only allowing neurons to spike at most once (Bohte et al., 2002; Mostafa, 2017; Comsa et al., 2020).

A recent proposal known as surrogate gradient training can overcome these limitations and has been shown to improve training on challenging datasets using models of increasing biological realism (Eshraghian et al., 2021). Surrogate gradient training replaces the undefined derivate of the neuron's activation function with a surrogate function and uses backpropagation through time (BPPT) for training, since SNNs are a particular form of recurrent neural network (RNN) (Neftci et al., 2019). SNNs thus experience many shortcomings associated with RNNs, such as their notably slow training times resulting from their sequential nature (Kuchaiev & Ginsburg, 2017; Vaswani et al., 2017). Furthermore, SNNs require multiple regularisation terms to avoid

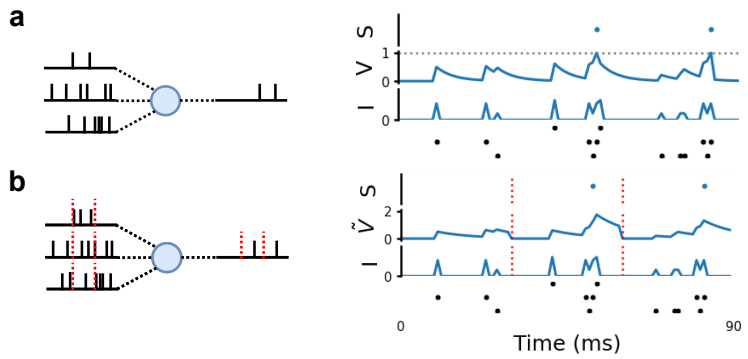

Figure 1: **a.** Left: The standard leaky integrate and fire (LIF) model. Right: Input and output activity of the neuron (bottom panel: Input raster, lower middle panel: Input current I, upper middle panel: Membrane potential V with the dotted line representing the firing threshold, and top panel: Output spikes S). **b.** Left: Our d-block model (here $d = 3$ blocks with edges denoted by vertical dotted red lines). Right: Input and output activity of the neuron (same as in the LIF model but with a change in the upper middle panel, where the d-block computes membrane potentials without reset $\tilde{V}$).

overfitting and to obtain sparse spiking activity (Zenke & Vogels, 2021; Perez-Nieves & Goodman, 2021), where the latter is important in obtaining energy-efficient computations on neuromorphic hardware (Panda et al., 2020).

In this work, we address these shortcomings by proposing a new SNN model with stochastic absolute refractory periods and recurrent conductance latencies. The absolute refractory period is the brief period after a spike during which a neuron cannot spike again; it is a feature of the biology not typically incorporated into standard SNNs. In a recent study, Taylor et al. (2022) proposed a model for accelerated training of SNNs in which individual neurons spike at most once. We use this model as a sub-component (a block) in our new model. We construct our model by concatenating these blocks across time and hence refer to our model as the d-block model, as each neuron can spike at most d times when employing d blocks (Figure 1b). Our contributions are summarised as follows.

1. We propose a new SNN model which uses fewer sequential operations compared to conventional SNNs and is henceforth more parallelisable on GPUs. We experimentally validate our model to obtain accelerated training speeds on synthetic benchmarks (up to 34× speedup) and various neuromorphic datasets (achieving up to 9× speedup).
2. Our model theoretically consumes less energy than conventional SNNs when employed on neuromorphic hardware, as it emits fewer spikes than standard SNNs during inference across multiple neuromorphic datasets. Furthermore, it does not include any training regularisation, thus avoiding the need for time-consuming and energy-demanding hyperparameter searches to find suitable regularisation values.
3. Notably we obtain state-of-the-art results on challenging neuromorphic datasets with an 86.20% and 68.16% accuracy on the Spiking Heidelberg Dataset (SHD) and Spiking Speech Commands (SSC) dataset respectively, raising performance by ~ 3% and ~ 8% over previously published results using standard SNNs.

2 Background and Related Work

2.1 Standard Spiking Model

A spiking neural network (SNN) is a type of recurrent neural network in which neurons output binary signals known as spikes. A neuron i in layer l (consisting of $\mathbb{R}^{N^{(l)}}$ neurons) emits a spike $S_i^{(l)}[t] = f(V_i^{(l)}[t]) \in \{0,1\}$ at time t if its membrane potential $V_i^{(l)}[t]$ reaches firing threshold V_{th}.

$$f(V_i^{(l)}[t]) = \begin{cases} 1, & \text{if } V_i^{(l)}[t] > V_{th} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

The leaky integrate and fire (LIF) model describes the evolution of the membrane potential $V_i^{(l)}(t)$ for resting potential $V_{rest} \in \mathbb{R}$, membrane time constant $\tau \in \mathbb{R}$ and input resistance $R \in \mathbb{R}$ (Gerstner et al., 2014).

$$\tau \frac{dV_i^{(l)}(t)}{dt} = -V_i^{(l)}(t) + V_{rest} + RI_i^{(l)}(t) \tag{2}$$

Without loss of generality the model is normalised ($V_i^{(l)}(t) \in [0,1]$ by $V_{rest} = 0, V_{th} = 1, R = 1$) and discretised (Taylor et al., 2022), such that for simulation time steps $t \in \{1, \dots, T\}$:

$$V_i^{(l)}[t+1] = \beta V_i^{(l)}[t] + (1-\beta) \underbrace{\left(b_i^{(l)} + \sum_{j=1}^{N^{(l-1)}} W_{ij}^{(l)} S_j^{(l-1)}[t+1] + \sum_{j=1}^{N^{(l)}} W_{ij}^{\text{rec}\,(l)} S_j^{(l)}[t] \right)}_{\text{Input current } I_i^{(l)}[t+1]} - \underbrace{S_i^{(l)}[t]}_{\text{Spike reset}} \tag{3}$$

The membrane potential is charged by the constant bias current source $b_i^{(l)}$, and the input $S^{(l-1)}[t] \in \mathbb{R}^{N^{(l-1)}}$ and output spikes $S^{(l)}[t-1] \in \mathbb{R}^{N^{(l)}}$ through feedforward $W^{(l)} \in \mathbb{R}^{N^{(l)} \times N^{(l-1)}}$ and recurrent connectivity $W^{\text{rec}\,(l)} \in \mathbb{R}^{N^{(l)} \times N^{(l)}}$ respectively. At every time step the membrane potential dissipates by a factor $0 \leq \beta = \exp(\frac{-\Delta t}{\tau}) \leq 1$ (for simulation time resolution $\Delta t \in \mathbb{R}$). The membrane potential is at rest $V_{rest} = 0$ in the absence of any input current and a spike $S_i^{(l)}[t] = 1$ is outputted if the potential rises above the firing threshold $V_{th} = 1$ (after which the potential is reduced back close to resting state).

2.2 Training techniques

To date, SNN training remains challenging. The success of backpropagation (Rumelhart et al., 1986) in ANNs does not naturally translate to SNNs due to their non-differentiable nature. Various methods have been proposed to circumvent this issue, however, these either fail to properly utilise the temporal dynamics of neurons or result in slow training times.

Shadow training: ANN to SNN conversion Instead of training a SNN, shadow training converts an already trained ANN to a SNN such that the firing rates of the SNN neurons approximate the activations of the ANN units (O'Connor et al., 2013; Esser et al., 2015; Rueckauer et al., 2016; 2017). Although this method permits deep SNNs to perform well on challenging large-scale datasets like Imagenet (Deng et al., 2009), it endures various shortcomings. Firstly, it requires long simulation durations to obtain reasonable prediction accuracies, which have been argued to largely diminish the energy efficiencies of SNNs emulated on neuromorphic hardware (Davidson & Furber, 2021). Secondly, converted SNNs perform worse than ANNs due to the conversion process. Thirdly, although this performance gap can be reduced by coupling ANN and SNN training (Wu et al., 2021a;b; Kheradpisheh et al., 2021), the modelling applicability to neuroscience is limited due to the imposed rate-code (whereas the brain might instead process stimuli using a temporal code (Guyonneau et al., 2004; Cariani, 1999)).

Direct SNN training Although the standard use of backpropagation is prohibited, alternative approaches have been proposed to estimate the gradients of network weights. Perturbation learning randomly perturbs network weights to approximate gradients, yet this method requires many trials to average out noise and scales poorly with growing network architectures (Williams, 1992; Seung, 2003). Latency learning calculates the gradients at the time of spiking as, unlike spikes, time is continuous (Bohte et al., 2002; Mostafa, 2017; Comsa et al., 2020; Kheradpisheh & Masquelier, 2020). However, this method enforces the modelling constraint that neurons spike

at most once and is affected by the dead neuron problem, where a lack of spike activity is detrimental to learning. Surrogate gradient learning circumvents these issues by replacing the undefined derivative of the spike function with a surrogate function (Esser et al., 2016; Hunsberger & Eliasmith, 2015; Zenke & Ganguli, 2018; Lee et al., 2016). This method allows networks to learn temporal dynamics since it passes gradients through time (Bellec et al., 2018; Shrestha & Orchard, 2018; Neftci et al., 2019), and has thus become the status quo for directly training SNNs. However, learning is very slow as the network is sequentially simulated at every point in time (which we overcome by processing multiple time steps at once).

2.3 Accelerated surrogate gradient training

Some work has addressed the slow training times of SNNs using surrogate gradients. Perez-Nieves & Goodman (2021) managed to obtain faster training speeds by developing a sparse backprop algorithm, which only passes gradients through the membrane potentials that are close to the firing threshold. Although they manage to accelerate the backward pass up to 70×, these speedups depend on the development of custom CUDA kernels, which only support simple feedforward architectures (whereas we accelerate training of SNNs with recurrent connectivity and trainable neural time constants). Other works have managed to speed up training by removing the backward pass and performing all learning online (Bellec et al., 2020; Murray, 2019). These approaches, however, result in inferior performance accuracies on various datasets in comparison to standard surrogate gradient descent. Lastly, a core limitation of all of these speedup approaches is the prerequisite of sequentially simulating the network at every point in time (whereas we lessen this bottleneck).

2.4 Modelling extensions

Various extensions of the standard LIF model have been proposed with increasing biological realism. These models operate on different timescales through the inclusion of trainable neural parameters shown to increase prediction accuracies across different temporal tasks. Modelling extensions include heterogeneous membrane time constants (Perez-Nieves et al., 2021), adaptive firing thresholds (Bellec et al., 2018; Yin et al., 2021), fixed refractory periods (Bellec et al., 2020) and variable conductance latencies (Shrestha & Orchard, 2018; Schrauwen & Van Campenhout, 2004). Following the success thereof, our model includes learnable membrane time constants, stochastic absolute refractory periods and stochastic recurrent conductance latencies.

3 Faster training with the d-block model

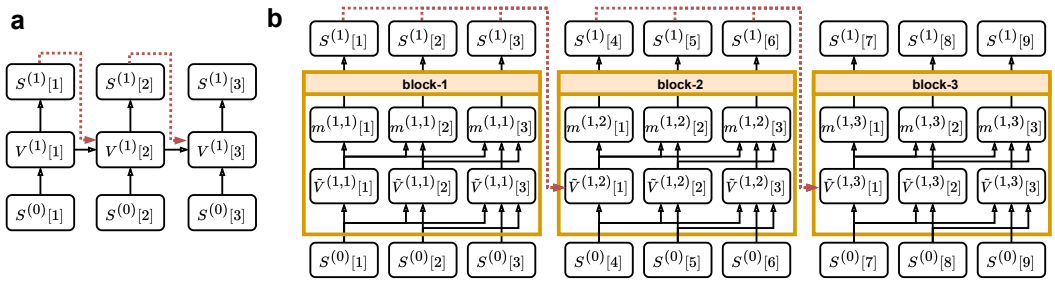

Figure 2: Computational graphs. **a.** Standard model. The membrane potentials $V^{(1)}[t]$ are recurrently dependent on the prior values $V^{(1)}[t-1]$ and charged by input spikes $S^{(0)}[t]$ and output spikes $S^{(1)}[t]$ through feedforward and recurrent weights (denoted by dotted red lines) respectively. Output spikes are generated by passing the membrane potentials $V^{(1)}[t]$ through the spike activation function. **b.** Our d-block model. Input spikes are processed by d (here $d = 3$) equal length blocks, where every block is a single-spike SNN proposed by Taylor et al. (2022). Input spikes $S^{(0)}[t]$ charge the membrane potentials without reset $\tilde{V}^{(1)}[t]$, which are mapped to spike outputs $S^{(1)}[t]$ using a grouping of operations $m^{(1)}[t]$ (see Appendix). Recurrent connections (dotted red lines) transport output spikes from one block to another.

3.1 ACCELERATED SINGLE-SPIKING MODEL

In a recent study Taylor et al. (2022) managed to obtain competitive results across various image and neuromorphic datasets using a new model for training single-spike SNNs in which individual neurons spike at most once. They were able to bypass the dead neuron problem associated with other single-spike training methods (Eshraghian et al., 2021) and overcome the slow sequential nature of surrogate gradient training. In the standard SNN, the input spikes $S^{(0)}[t]$ charge the membrane potentials $V^{(1)}[t]$ and output spikes $S^{(1)}[t]$ are emitted at every point in time t (Figure 2a). This process is sequential as the membrane potentials $V^{(1)}[t]$ are dependent on their prior values $V^{(1)}[t-1]$. Taylor et al. (2022) avoided this sequential computation by instead computing all output spikes $S^{(1)}[t]$ simultaneously across time. Instead of computing the membrane potentials $V^{(1)}[t]$, they calculated membrane potentials without reset $\tilde{V}^{(1)}[t]$ (i.e. no-reset membrane potentials) and through successive stages of processing $m^{(1)}[t]$ (see Appendix) were able to recover the correct output spikes $S^{(1)}[t]$ using fast non-sequential GPU parallelisable operations. From hereon we refer to this model as a "block".

3.2 EXTENDING THE SINGLE-SPIKE BLOCK TO MULTI-SPIKE BLOCKS

Currently, the block emits a single spike over the simulation period T. To extend the model to use multiple spikes, we divide the simulation period T into d chunks of equal length T/d and process each chunk by a block using the same neural parameters (i.e. weight sharing the synaptic connectivity and membrane time constants; Figure 2b). If the input data is not divisible by d, we split the data into d chunks of length $\lfloor T/d \rfloor$ and a final chunk of length $T - d \cdot \lfloor T/d \rfloor$. In this multi-block extension, which we refer to as the d-block model, neurons can spike up to d times over the simulation period T. Every neuron is governed by a stochastic absolute refractory period as a neuron can spike at most once within a block of length T/d and will therefore need to wait between 1 and T/d time steps before it can spike again in the next block.

3.3 INCLUDING RECURRENT CONNECTIONS

Recurrent connections are included within the d-block model by connecting the output spikes of block-n as input to the next block-$(n+1)$. The output spikes $\mathbf{S}^{(l,n)} \in \mathbb{R}^{N^{(l)} \times T}$ in layer l and block n are flattened along time $\max_t S^{(l,n)}[t] \in \mathbb{R}^{N^{(l)}}$ (to denote if the neurons have spiked within the block's duration) before transporting them to the next block $n+1$ via the recurrent connectivity $W^{\text{rec}\,(l)}$.[1] Different from the standard SNN, these connections have a variable conduction latency equivalent in duration to the refractory periods of the afferent neurons in layer l and block-n. The starting (no-reset) membrane potential of neuron i in layer l and block-$(n+1)$ is then computed from the flattened output spikes of block-n and the first input spikes to block-$(n+1)$.

$$\tilde{V}_i^{(l,n+1)}[1] = \underbrace{\left(b_i^{(l)} + \sum_{j=1}^{N^{(l-1)}} W_{ij}^{(l)} S_j^{(l-1,n+1)}[1]\right)}_{\text{Feedforward current}} + \underbrace{\sum_{j=1}^{N^{(l)}} W_{ij}^{\text{rec}\,(l)} \max_t S^{(l,n)}[t]}_{\text{Recurrent current}} \tag{4}$$

4 EXPERIMENTS AND RESULTS

We evaluate the training speedup and performance of our model on real datasets in comparison to prior work. We used PyTorch (Paszke et al., 2017) for all implementations with benchmarks and training conducted on a cluster of NVIDIA Tesla V100 GPUs.

4.1 SPEEDUP BENCHMARKS

We benchmarked the time required to complete a single surrogate gradient training step in our d-block model and the standard SNN for a different number of blocks d, hidden units n, simulation steps t and batch sizes b on a synthetic spike dataset (see Appendix).

[1] For simplicity we have omitted the batch dimension, which is included in our implementation and used in our experiments.

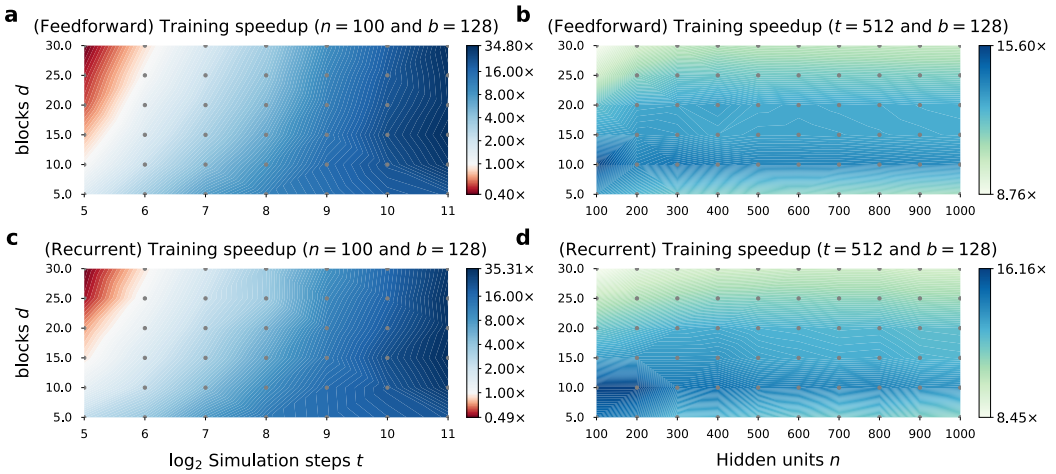

Figure 3: Training speedup of our d-block model over the standard SNN for feedforward and recurrent networks. **a.** Feedforward network training speedup as a function of the number of blocks d and simulation steps t (for fixed hidden neurons $n = 100$ and batch size $b = 128$). **b.** Feedforward network training speedup as a function of the number of blocks d and hidden neurons n (for fixed simulation steps $t = 512$ and batch size $b = 128$). **c.** Same as **a.** but for recurrent networks. **d.** Same as **b.** but for recurrent networks.

Robust speedup for a different number of simulation steps Considering only feedforward SNNs (Figure 3a-b), we find our d-block model to train faster for a growing number of simulation steps, with a maximum training speedup of 34× for $d = 15$ blocks and $t = 2048$ simulation steps in a single hidden layer model of 100 feedforward neurons (Figure 3a). These speedups are robust over different numbers of blocks (e.g. we obtain speedups of 21×, 21× and 18× using $d = 10$, $d = 20$ and $d = 30$ blocks respectively over $t = 1024$ simulation steps), yet start to decline when the number of blocks approaches the number of simulation steps (e.g. we obtain a slower "speedup" of 0.4× when using $d = 30$ blocks and $t = 32$ simulation steps).[2] Speedups are further raised when using smaller batch sizes (with batch sizes $b = 32$ and $b = 64$ obtaining a maximum speedup of 40× and 37× respectively) or fixed membrane time constants (obtaining a maximum speedup of 44×; See Appendix).

Robust speedups using different number of hidden units Also for feedforward SNNs, we find our d-block model to maintain robust training speedups over larger number of hidden units (e.g. obtaining a 13× speedup when using $n = 200$, $n = 400$ or $n = 600$ hidden units for $d = 10$ blocks and $t = 512$ simulation steps; Figure 3b). Again, these speedups are more pronounced when using smaller batch sizes (obtaining a 15× speedup when using $n = 200$, $n = 400$ or $n = 600$ hidden units for $d = 10$ blocks with batch size $b = 32$) or fixed membrane time constants (obtaining a 16× speedup when using $n = 200$, $n = 400$ or $n = 600$ hidden units for $d = 10$ blocks; see Appendix).

Faster training speeds using recurrent connectivity Our recurrently connected model obtains faster training over the standard recurrent SNN. These speedups are slightly greater than the substantial speedups achieved when both models only employ feedforward connectivity. Again, in the recurrent case the training speeds increase with the number of simulation steps (with a maximum speedup of 35× for $d = 15$ blocks and $t = 2048$ simulation steps; Figure 3c) and are robust over larger numbers of hidden units (with an **18**×, 17× and 17× speedup when using $n = 200$, $n = 400$ and $n = 600$ hidden units respectively for $d = 10$ blocks and $t = 512$ simulation steps; Figure 3d). As in the feedforward model, training speedups are further amplified in the recurrent model when using smaller batch sizes (with batch sizes $b = 32$ and $b = 64$ obtaining a

[2] Our d-block model is identical to the standard SNN model when the number of blocks d is equal to the number of simulation steps t, yet our model performs more computational work per time step and hence training slows down as d tends to t in the limit.

maximum speedup of 41× and 39× respectively) or fixed membrane time constants (obtaining a maximum speedup of 44×; See Appendix).

4.2 PERFORMANCE ON REAL DATASETS

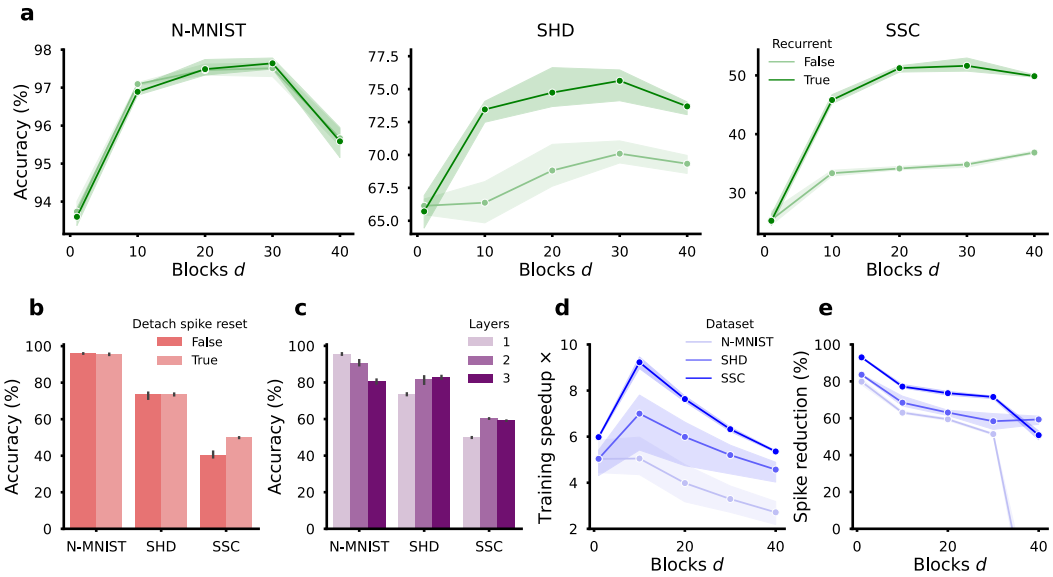

Figure 4: Analysis of our d-block model on challenging neuromorphic datasets. We use a single recurrently connected hidden layer network of 128 neurons and report results for three repeat runs of the model for which the mean and s.d. are plotted. **a.** Accuracy as a function of the number of blocks d using feedforward and recurrent connectivity. **b.** Accuracy with the spike reset being attached or detached from the computational graph during training. **c.** Accuracy as a function of an increasing number of hidden layers. **d.** Training speedup of our model vs the standard SNN as a function of the number of blocks d. **e.** Reduction in spikes during inference of our model vs the standard SNN as a function of blocks d.

Table 1: Performance comparison to existing literature († denotes data augmentation, $^\beta$ denotes trainable time constants and we report our results in bold as an average of three repeat runs).

Dataset	Model	Architecture	Neuron model	Accuracy (%)
N-MNIST	**our model**	400-10 ($d = 30$)	recurrent LIF$^\beta$	**98.04 ± 06**
	Shrestha & Orchard (2018)	500-500-10	feedforward LIF	98.89 ± 0.06
	Lee et al. (2016)	800-10	feedforward LIF	98.66
	Wu et al. (2018)	400-400-10	feedforward LIF	98.78
SHD	**our model**	256-256-20 ($d = 30$)	recurrent LIF$^\beta$	**86.20 ± 0.33**
	Cramer et al. (2020)†	1024-20	recurrent LIF	83.2 ± 1.30
	Perez-Nieves et al. (2021)	128-20	recurrent LIF$^\beta$	82.7 ± 0.80
	Eshraghian & Lu (2022)	3000-20	recurrent LIF	82.27 ± 0.27
	Zenke & Vogels (2021)	256-20	recurrent LIF	82.0 ± 0.02
	Cramer et al. (2022)	186-20	recurrent LIF	76.2 ± 1.3
	Cramer et al. (2020)	128-20	recurrent LIF	71.4 ± 1.90
SSC	**our model**	256-256-256-35 ($d = 30$)	recurrent LIF$^\beta$	**68.16 ± 0.28**
	Perez-Nieves et al. (2021)	128-20	recurrent LIF$^\beta$	60.1 ± 0.7
	Cramer et al. (2020)	128-20	recurrent LIF	50.9 ± 1.1

We tested the applicability of our model on different neuromorphic datasets of increasing difficulty. The simplest dataset is the N-MNIST dataset (Orchard et al., 2015), in which the MNIST image dataset (LeCun, 1998) is converted into spikes using a neuromorphic vision sensor. More challenging are the Spiking Heidelberg Dataset (SHD) and Spiking Speech Commands (SSC) datasets, in which spoken digits and commands are converted into spikes using a model of auditory bushy cells in the cochlear nucleus (Cramer et al., 2020). These challenging temporal datasets provide a good base to evaluate our model as we are able to study the effect of using multiple blocks and recurrent connectivity, whereas other datasets like images are less suited as they are readily solved using single-spike and non-recurrent SNNs (Zhou et al., 2021).

State-of-the-art results on neuromorphic datasets We obtain competitive accuracies across the different neuromorphic datasets, reaching an accuracy of 98.04%, 86.20% and 68.16% on the N-MNIST, SHD and SSC dataset respectively (Table 1). Notably, we improve performance on the challenging SHD and SSC dataset by $\sim 3\%$ and $\sim 8\%$ respectively over prior published results using standard recurrently connected LIF networks.

Improving performance using recurrence, more blocks and additional layers We trained multiple models (comprising a single hidden layer of 128 neurons) across the different datasets to investigate the effect of including recurrence and increasing the number of blocks. We find that including recurrent connections in our model improves prediction accuracies across all datasets, especially on the more challenging SHD and SSC datasets (Figure 4a). Increasing the number of blocks also improves performance (Figure 4a). However, performance starts to drop for a large number of blocks ($d = 40$ across all datasets), likely due to overfitting as we do not include any regularisation. We also investigate the effect of detaching the flow of gradients through the recurrent connections, as doing so has been shown to improve training in standard recurrent SNNs (Zenke & Vogels, 2021). We observe no degradation in performance when detaching the spike reset terms from the computational graphs during training and find that it improves the performance on the SSC dataset (Figure 4b). Lastly, we investigate the effect of using additional hidden layers and find that additional layers improve accuracies on the SHD and SCC datasets, but degrade performance on the N-MNIST dataset (Figure 4c).

Accelerated training We find that our model trains faster than the standard SNN, with our model obtaining a maximum training speedup of $\sim 5\times$, $\sim 7\times$, and $\sim 9\times$ on the N-MNIST, SHD, and SSC datasets, respectively, for $d = 10$ blocks (Figure 4d). The difference in speedups is due to the different temporal lengths and input and output dimensions of the datasets.

Increased spike sparsity during inference Our model uses fewer spikes than standard SNNs during inference, with a $> 50\%$ spike count reduction across all datasets when using $d = 30$ blocks and a $> 80\%$ reduction when using $d = 1$ block(Figure 4e). Our model thus theoretically requires less energy than standard SNNs if emulated on neuromorphic hardware, as the energy consumption scales approximately proportionally with the number of emitted spikes (Panda et al., 2020). However, we find that sparsity starts to decline for a growing number of blocks. Although we obtain favourable sparsity for the SHD (59% spike count reduction) and SSC (50% spike count reduction) datasets, we find our model to obtain worse sparsity on the N-MNIST dataset (41% spike count increase) when using $d = 40$ blocks. This suggests that sparsity advantages are dependent on the number of blocks and the dataset.

5 Discussion

Surrogate gradient descent has been a substantial algorithmic development for directly training SNNs (Eshraghian et al., 2021). However, given the relation of SNNs to RNNs (Neftci et al., 2019), SNNs have been slow to train due to their sequential nature, precluding efficient parallelisation on GPUs (Kuchaiev & Ginsburg, 2017; Vaswani et al., 2017). In a recent study Taylor et al. (2022) proposed a model for accelerating the training of single-spike SNNs, which eliminates all sequential computation and exclusively relies on fast vectorised operations. In this work, we extend their model from the single-spike to the multi-spike setting by concatenating the single-spike model along time, thus processing blocks of time as opposed to each time step individually

as done in standard SNNs. We experimentally validate our new multi-spike model, demonstrating that it achieves faster training speeds on both synthetic benchmarks (up to 34× speedup) and on various neuromorphic datasets (up to ~ 9× speedup). Furthermore, our model achieves state-of-the-art results on the challenging SHD and SSC neuromorphic datasets, with an accuracy of 86.20% and 68.16%, respectively, raising performance by ~ 3% and ~ 8% over prior published results using standard LIF networks. We are also able to do so without any training regularisation, whereas other works employ various regularisation constraints to avoid overfitting and to obtain an efficient spike code (Zenke & Vogels, 2021; Perez-Nieves & Goodman, 2021). Even so, we find our model to emit fewer spikes during inference in comparison to standard SNNs, thus theoretically lowering energy requirements relative to the standard SNN when emulating our model on neuromorphic hardware (Panda et al., 2020). Although these results demonstrate the effectiveness of our model, there are various avenues for improving our work in future investigations.

Faster training speeds Directly employing standard backprop for SNN training does not take advantage of their sparse spiking nature and renders training slow by performing many unnecessary dense tensor operations. Perez-Nieves & Goodman (2021) address this issue by proposing a sparse gradient descent algorithm for accelerating the backward pass. As pointed out by Taylor et al. (2022), the development of a sparse gradient descent algorithm would also accelerate training in the block model and consequently also accelerate training in our d-block model.

Improving performance There are multiple extensions for increasing the accuracy of our model on real datasets: Firstly, different architectures like convolutional networks could be explored, which have previously shown to be effective in the single-spike block model (Taylor et al., 2022). Secondly, different types of surrogate functions could be benchmarked as these have shown to have an effect on training (Zenke & Vogels, 2021). Thirdly, our model could be extended to include additional mechanisms, such as temporal attention (Yao et al., 2021) or adaptive firing thresholds (Yin et al., 2021), as these have shown to perform better than the standard LIF model. Lastly, we have limited ourselves to using blocks of equal length, whereas better performance might be obtained using blocks of varying lengths.

Theoretical insights Our model is able to obtain greater accuracies than the standard SNN on the challenging SHD and SSC neuromorphic datasets. We hypothesise two reasons as to why this might be the case: (1) our model avoids overfitting the training data due to the implicit regularisation imposed by the stochastic absolute refractory periods and recurrent conductance latencies. (2) our model obtains a more salient flow of gradient through time due to the recurrent conductance latencies acting as a form skip connection (Srivastava et al., 2015; He et al., 2016), but through time. It would be useful to more rigorously investigate these hypotheses to obtain a deeper theoretical understanding of the inner workings of our model.

Extended avenues SNNs form a valuable basis for studying the brain *in silico* and in obtaining energy efficient compute when emulated on neuromorphic hardware. We hope the d-block model further contributes toward these goals, by allowing computational neuroscientists to train SNNs over longer timespans with finer temporal granularity and for neuromorphic engineers to further reduce the energy requirements of machine learning systems by adopting our sparse spiking model.

6 Reproducibility Statement

We outline the theoretical construction of our model in section 3. An implementation of our model and instructions to replicate the experiments and results of this paper can be found at https://github.com/webstorms/FastSNN, where we publish our code under the BSD 3-Clause Licence. Lastly, training details are provided in the Appendix.

References

Guillaume Bellec, Darjan Salaj, Anand Subramoney, Robert Legenstein, and Wolfgang Maass. Long short-term memory and learning-to-learn in networks of spiking neurons. *Advances in neural information processing systems*, 31, 2018.

Guillaume Bellec, Franz Scherr, Anand Subramoney, Elias Hajek, Darjan Salaj, Robert Legenstein, and Wolfgang Maass. A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature communications*, 11(1):1–15, 2020.

Sander M Bohte, Joost N Kok, and Han La Poutre. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, 48(1-4):17–37, 2002.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Santiago A Cadena, George H Denfield, Edgar Y Walker, Leon A Gatys, Andreas S Tolias, Matthias Bethge, and Alexander S Ecker. Deep convolutional models improve predictions of macaque v1 responses to natural images. *PLoS computational biology*, 15(4):e1006897, 2019.

Peter Cariani. Temporal coding of periodicity pitch in the auditory system: an overview. *Neural plasticity*, 6(4):147–172, 1999.

Iulia M Comsa, Krzysztof Potempa, Luca Versari, Thomas Fischbacher, Andrea Gesmundo, and Jyrki Alakuijala. Temporal coding in spiking neural networks with alpha synaptic function. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8529–8533. IEEE, 2020.

Benjamin Cramer, Yannik Stradmann, Johannes Schemmel, and Friedemann Zenke. The heidelberg spiking data sets for the systematic evaluation of spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

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.

Simon Davidson and Steve B Furber. Comparison of artificial and spiking neural networks on digital hardware. *Frontiers in Neuroscience*, 15:345, 2021.

Sophie Denève and Christian K Machens. Efficient codes and balanced networks. *Nature neuroscience*, 19(3):375–382, 2016.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Jason K Eshraghian and Wei D Lu. The fine line between dead neurons and sparsity in binarized spiking neural networks. *arXiv preprint arXiv:2201.11915*, 2022.

Jason K Eshraghian, Max Ward, Emre Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D Lu. Training spiking neural networks using lessons from deep learning. *arXiv preprint arXiv:2109.12894*, 2021.

Steve K Esser, Rathinakumar Appuswamy, Paul Merolla, John V Arthur, and Dharmendra S Modha. Backpropagation for energy-efficient neuromorphic computing. *Advances in neural information processing systems*, 28, 2015.

Steven K. Esser, Paul A. Merolla, John V. Arthur, Andrew S. Cassidy, Rathinakumar Appuswamy, Alexander Andreopoulos, David J. Berg, Jeffrey L. McKinstry, Timothy Melano, Davis R. Barch, Carmelo di Nolfo, Pallab Datta, Arnon Amir, Brian Taba, Myron D. Flickner, and Dharmendra S. Modha. Convolutional networks for fast, energy-efficient neuromorphic computing. *Proceedings of the National Academy of Sciences*, 113(41):11441–11446, 2016. doi: 10.1073/pnas.1604850113.

Andrew Francl and Josh H McDermott. Deep neural network models of sound localization reveal how perception is adapted to real-world environments. *Nature Human Behaviour*, 6(1):111–133, 2022.

Wulfram Gerstner, Werner M Kistler, Richard Naud, and Liam Paninski. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

Rudy Guyonneau, Rufin VanRullen, and Simon J Thorpe. Temporal codes and sparse representations: a key to understanding rapid processing in the visual system. *Journal of Physiology-Paris*, 98(4-6):487–497, 2004.

Nicol S Harper, Oliver Schoppe, Ben DB Willmore, Zhanfeng Cui, Jan WH Schnupp, and Andrew J King. Network receptive field modeling reveals extensive integration and multi-feature selectivity in auditory cortical neurons. *PLoS computational biology*, 12(11):e1005113, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Eric Hunsberger and Chris Eliasmith. Spiking deep networks with lif neurons. *arXiv preprint arXiv:1510.08829*, 2015.

Saeed Reza Kheradpisheh and Timothée Masquelier. Temporal backpropagation for spiking neural networks with one spike per neuron. *International Journal of Neural Systems*, 30(06): 2050027, 2020.

Saeed Reza Kheradpisheh, Maryam Mirsadeghi, and Timothée Masquelier. Spiking neural networks trained via proxy. *arXiv preprint arXiv:2109.13208*, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Oleksii Kuchaiev and Boris Ginsburg. Factorization tricks for lstm networks. *arXiv preprint arXiv:1703.10722*, 2017.

Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. Training deep spiking neural networks using backpropagation. *Frontiers in neuroscience*, 10:508, 2016.

Hesham Mostafa. Supervised learning based on temporal coding in spiking neural networks. *IEEE transactions on neural networks and learning systems*, 29(7):3227–3235, 2017.

James M Murray. Local online learning in recurrent networks with random feedback. *Elife*, 8: e43299, 2019.

Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

Peter O'Connor, Daniel Neil, Shih-Chii Liu, Tobi Delbruck, and Michael Pfeiffer. Real-time classification and sensor fusion with a spiking deep belief network. *Frontiers in neuroscience*, 7:178, 2013.

Garrick Orchard, Ajinkya Jayawant, Gregory K Cohen, and Nitish Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in neuroscience*, 9:437, 2015.

Priyadarshini Panda, Sai Aparna Aketi, and Kaushik Roy. Toward scalable, efficient, and accurate deep spiking neural networks with backward residual connections, stochastic softmax, and hybridization. *Frontiers in Neuroscience*, 14:653, 2020.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Nicolas Perez-Nieves and Dan Goodman. Sparse spiking gradient descent. *Advances in Neural Information Processing Systems*, 34:11795–11808, 2021.

Nicolas Perez-Nieves, Vincent CH Leung, Pier Luigi Dragotti, and Dan FM Goodman. Neural heterogeneity promotes robust learning. *Nature communications*, 12(1):1–9, 2021.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Blake A Richards, Timothy P Lillicrap, Philippe Beaudoin, Yoshua Bengio, Rafal Bogacz, Amelia Christensen, Claudia Clopath, Rui Ponte Costa, Archy de Berker, Surya Ganguli, et al. A deep learning framework for neuroscience. *Nature neuroscience*, 22(11):1761–1770, 2019.

Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, and Michael Pfeiffer. Theory and tools for the conversion of analog to spiking convolutional neural networks. *arXiv preprint arXiv:1612.04052*, 2016.

Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience*, 11:682, 2017.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Benjamin Schrauwen and Jan Van Campenhout. Extending spikeprop. In *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541)*, volume 1, pp. 471–475. IEEE, 2004.

Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. Green ai. *Communications of the ACM*, 63(12):54–63, 2020.

H Sebastian Seung. Learning in spiking neural networks by reinforcement of stochastic synaptic transmission. *Neuron*, 40(6):1063–1073, 2003.

Sumit B Shrestha and Garrick Orchard. Slayer: Spike layer error reassignment in time. *Advances in neural information processing systems*, 31, 2018.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Yosef Singer, Yayoi Teramoto, Ben DB Willmore, Jan WH Schnupp, Andrew J King, and Nicol S Harper. Sensory cortex is optimized for prediction of future input. *elife*, 7:e31557, 2018.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*, 2019.

Luke Taylor, Andrew King, and Nicol Harper. Accelerating spiking neural network training. *arXiv preprint arXiv:2205.15286*, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Tim P Vogels, Henning Sprekeler, Friedemann Zenke, Claudia Clopath, and Wulfram Gerstner. Inhibitory plasticity balances excitation and inhibition in sensory pathways and memory networks. *Science*, 334(6062):1569–1573, 2011.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Jibin Wu, Yansong Chua, Malu Zhang, Guoqi Li, Haizhou Li, and Kay Chen Tan. A tandem learning rule for effective training and rapid inference of deep spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2021a.

Jibin Wu, Chenglin Xu, Xiao Han, Daquan Zhou, Malu Zhang, Haizhou Li, and Kay Chen Tan. Progressive tandem learning for pattern recognition with deep spiking neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021b.

Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12:331, 2018.

Timo Wunderlich, Akos F Kungl, Eric Müller, Andreas Hartel, Yannik Stradmann, Syed Ahmed Aamir, Andreas Grübl, Arthur Heimbrecht, Korbinian Schreiber, David Stöckel, et al. Demonstrating advantages of neuromorphic computation: a pilot study. *Frontiers in neuroscience*, 13: 260, 2019.

Daniel LK Yamins and James J DiCarlo. Using goal-driven deep learning models to understand sensory cortex. *Nature neuroscience*, 19(3):356–365, 2016.

Man Yao, Huanhuan Gao, Guangshe Zhao, Dingheng Wang, Yihan Lin, Zhaoxu Yang, and Guoqi Li. Temporal-wise attention spiking neural networks for event streams classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10221–10230, 2021.

Bojian Yin, Federico Corradi, and Sander M Bohté. Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks. *Nature Machine Intelligence*, 3(10): 905–913, 2021.

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

Friedemann Zenke and Tim P Vogels. The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks. *Neural Computation*, 33(4):899–925, 2021.

Shibo Zhou, Xiaohua Li, Ying Chen, Sanjeev T Chandrasekaran, and Arindam Sanyal. Temporal-coded deep spiking neural network with easy training and robust performance. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11143–11151, 2021.

A Appendix

A.1 The block model

The block model (Taylor et al., 2022) is described using three steps:

1. Time series of presynaptic spikes $\mathbf{S}_j^{(l-1)}$ [3] are mapped to a time series of input currents $\mathbf{I}_i^{(l)}$ using a tensor multiplication.

$$I_i^{(l)}[t] = \sum_{j=1}^{N^{(l-1)}} W_{ij}^{(l)} S_j^{(l-1)}[t] \tag{5}$$

2. Membrane potentials $\tilde{\mathbf{V}}_i^{(l)}$ (without reset) are calculated using a convolution (where $\boldsymbol{\beta} = [\beta^0, \beta^1, \cdots, \beta^{T-1}]$).

$$\begin{aligned} \tilde{V}_i^{(l)}[t] &= \beta^t V_i^{(l)}[0] + (1-\beta) \sum_{k=1}^{t} \beta^{t-k} I_i^{(l)}[k] \\ &= \beta^t V_i^{(l)}[0] + (1-\beta)\big(\mathbf{I}_i^{(l)} \circledast \boldsymbol{\beta}\big)[t] \end{aligned} \tag{6}$$

[3] Bold face variables denotes arrays as opposed to scalar values.

3. Correct output spikes $\mathbf{S}_i^{(l)}$ are obtained using the following operations (which we grouped as m in the main paper body). (1) Erroneous output spikes $\tilde{\mathbf{S}}_i^{(l)}$ are obtained by passing the no-reset membrane potentials $\tilde{\mathbf{V}}_i^{(l)}$ through the spike function f. (2) These erroneous output spikes are transformed into a latent encoding $\mathbf{z}_i^{(l)} = \sum_{k=1}^{t} \tilde{S}_i^{(l)}[k](t-k+1)$, where every element therein encodes an ordering of the spikes. (3) Correct output spikes are obtained through transformation $\mathbf{S}_i^{(l)} = g(\mathbf{z}_i^{(l)})$, where

$$g(\mathbf{z}_i^{(l)})[t] = \begin{cases} 1, & \text{if } z_i^{(l)}[t] = 1 \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

A.2 Synthetic spike dataset for the speed benchmarks

As outlined in (Taylor et al., 2022), we constructed binary input spikes of shape $B \times N \times T$ (B being the batch size, N the number of input neurons and T the number of simulation steps), such that every batch dimension b had a firing rate $r_b \sim \mathbf{U}(u_{\min}, u_{\max})$ uniformly sampled (with $u_{\min} = 0$Hz and $u_{\max} = 200$Hz). For every batch dimension we generated a random binary spike matrix of shape $N \times T$, such that every input neuron in the matrix had an expected firing rate of r_bHz.

A.3 Training details and hyperparameters

A.3.1 Readout neurons

Every network had an output layer of readout neurons (containing the same number of neurons as the number of classes within the dataset trained on), where we removed the spike and reset mechanism (as done in Zenke & Vogels (2021)). The output of readout neuron c in response to input sample c was taken to be the summated membrane potential over time $o_{b,c} = \sum_t V_{b,c}^L[t]$.

A.3.2 Beta clipping

To enforce correct neuron dynamics we clipped the values of $\beta_i^{(l)}$ into the range $[0, 1]$.

$$\beta_i^{(l)} = \begin{cases} 1, & \text{if } \beta_i^{(l)} > 1 \\ 0, & \text{if } \beta_i^{(l)} < 0 \end{cases} \tag{8}$$

A.3.3 Weight initialisation

All network connectivity weights were sampled from a uniform distribution $\mathbf{U}(-\sqrt{N^{-1}}, \sqrt{N^{-1}})$ where N is the number of afferent connections. All biases were initialised as 0. All neurons in the hidden layers were initialised with a membrane time constant $\tau = 10$ms and $\tau = 20$ms for the readout neurons.

A.3.4 Supervised training loss

We trained all networks to minimise a cross-entropy loss (with B and C being the number of batch samples and dataset classes respectively)

$$\mathscr{L} = -\frac{1}{B} \sum_{b=1}^{B} \sum_{c=1}^{C} y_{b,c} \log(p_{b,c}) \tag{9}$$

Variable $y_{b,c} \in \{0, 1\}^C$ is the one hot target vector and $p_{b,c}$ are the network prediction probabilities, which were obtained by passing the readout neuron outputs $o_{b,c}$ through the softmax function.

$$p_{b,c} = \frac{\exp o_{b,c}}{\sum_{k=1}^{C} \exp o_{b,k}} \tag{10}$$

A.3.5 SURROGATE GRADIENT

We used the fast sigmoid function as our surrogate gradient (Zenke & Ganguli, 2018), which has been shown to work well in practice (Zenke & Vogels, 2021). Here hyperparameter β_{sur} (which we set to 10 in all experiments) defines the slope of the gradient.

$$\frac{df_{sur}(V)}{dV} = (\beta_{\text{sur}}|V|+1)^{-2} \tag{11}$$

A.3.6 TRAINING PROCEDURE

We used the Adam optimiser (with default parameters) (Kingma & Ba, 2014) for all training starting with an initial learning rate of 0.001, which was decayed by a factor of 10 every time the number of epochs reached a new milestone. A model was saved if it managed to lower the training error at the end of each epoch.

A.4 EXTENDED SPEEDUP RESULTS

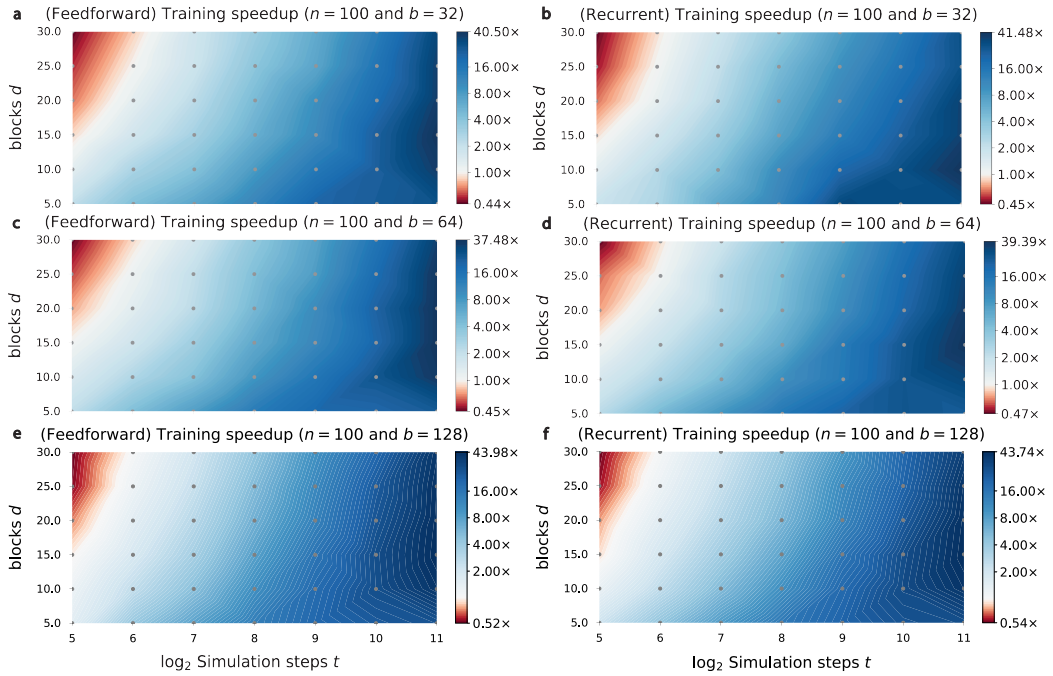

Figure 5: Training speedup of our d-block model over the standard SNN for feedforward and recurrent networks as a function of the number of blocks d and simulation steps t (for fixed hidden neurons $n = 100$). **a.-b.** Batch size 32. **c.-d.** Batch size 64. **e.-f.** Batch size 128 but using fixed membrane time constants. Left column: Feedforward networks and right column: Recurrent networks.

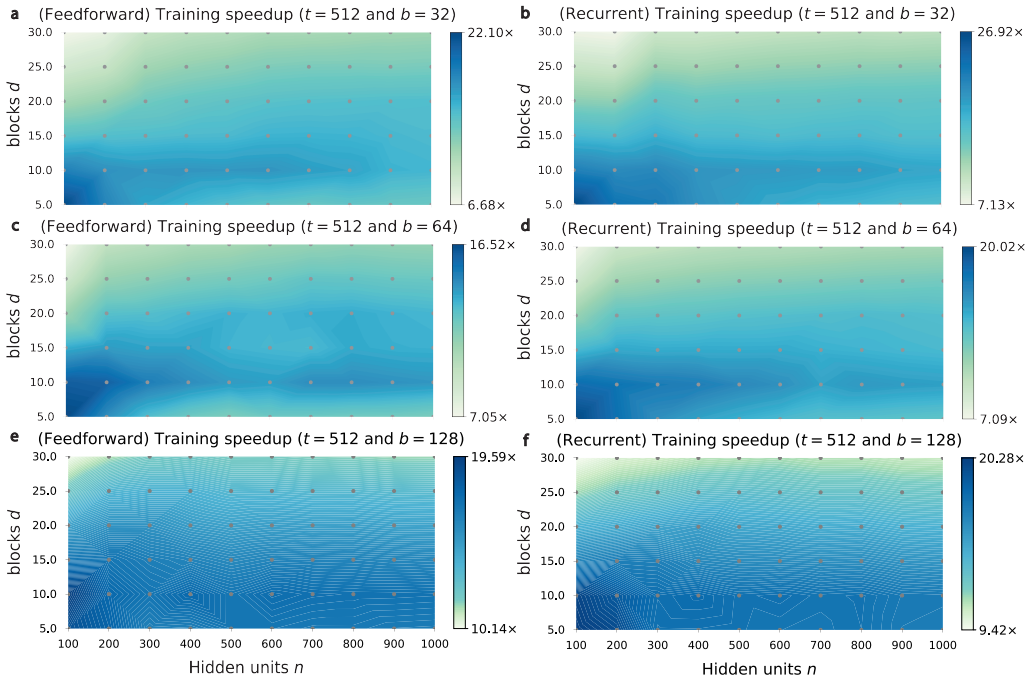

Figure 6: Training speedup of our d-block model over the standard SNN for feedforward and recurrent networks as a function of the number of blocks d and hidden neurons n(for fixed simulation steps $t = 512$). **a.-b.** Batch size 32. **c.-d.** Batch size 64. **e.-f.** Batch size 128 but using fixed membrane time constants. Left column: Feedforward networks and right column: Recurrent networks.