Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks

Bojian Yin[1,*], Federico Corradi[2], and Sander M. Bohté[1,3,4]

[1]CWI, Machine Learning group, Amsterdam, The Netherlands
[2]Stichting IMEC Netherlands, Holst Centre, Eindhoven, The Netherlands
[3]Univ of Amsterdam, Faculty of Science, Amsterdam, The Netherlands
[4]Rijksuniversiteit Groningen, Faculty of Science and Engineering, Groningen, The Netherlands
[*]byin@cwi.nl (corresponding author)

ABSTRACT

Inspired by more detailed modeling of biological neurons, Spiking neural networks (SNNs) have been investigated both as more biologically plausible and potentially more powerful models of neural computation, and also with the aim of extracting biological neurons' energy efficiency; the performance of such networks however has remained lacking compared to classical artificial neural networks (ANNs). Here, we demonstrate how a novel surrogate gradient combined with recurrent networks of tunable and adaptive spiking neurons yields state-of-the-art for SNNs on challenging benchmarks in the time-domain, like speech and gesture recognition. This also exceeds the performance of standard classical recurrent neural networks (RNNs) and approaches that of the best modern ANNs. As these SNNs exhibit sparse spiking, we show that they theoretically are one to three orders of magnitude more computationally efficient compared to RNNs with comparable performance. Together, this positions SNNs as an attractive solution for AI hardware implementations.

Introduction

The success of brain-inspired deep learning in AI is naturally focusing attention back onto those inspirations and abstractions from neuroscience[1]. One such example is the abstraction of the sparse, pulsed and event-based nature of communication between biological neurons into neural units that communicate real values at every iteration or timestep of evaluation, taking the rate of firing of biological spiking neurons as an analog value (Figure 1**a**). Spiking neurons, as more detailed neural abstractions, are theoretically more powerful compared to analog neural units[2] as they allow the relative timing of individual spikes to carry significant information. A real-world example in nature is the efficient sound localization in animals like Barn Owl's using precise spike-timing[3]. The sparse and binary nature of communication similarly has the potential to drastically reduce energy consumption in specialized hardware, in the form of neuromorphic computing[4].

Since their introduction, numerous approaches to learning in spiking neural networks have been developed[5–9]. All such approaches define how input signals are transduced into sequences of spikes, and how output spike-trains are interpreted with respect to goals, learning rules, or loss functions. For supervised learning, approaches that calculate the gradient of weights with respect to the loss have to deal with the discontinuous nature of the spiking mechanism inside neurons. Local linearized approximations like SpikeProp[5] can be generalized to approximate "surrogate" gradients[10], or even calculated exactly in special cases[11]. The use of surrogate gradients in particular has recently resulted in rapidly improving performance on select benchmarks, closing the performance gap with conventional deep learning approaches for smaller image recognition tasks like CIFAR10 and (Fashion) MNIST, and demonstrating improved performance on temporal tasks like TIMIT speech recognition[12]. Still, spiking neural networks (SNNs) have struggled to demonstrate a clear advantage compared to classical artificial neural networks (ANNs)[13,14].

Here, we introduce Spiking Recurrent Neural Networks (SRNNs), networks that include recurrently connected layers of spiking neurons (Figure 1**b**). We demonstrate how these networks can be trained to high performance on hard benchmarks, exceeding existing state-of-the-art in SNNs, and approaching or exceeding state-of-the-art in classical recurrent artificial neural networks. The high-performance in SRNNs is achieved by applying back-propagation-through-time (BPTT)[15] to spiking neurons using a novel Multi-Gaussian surrogate gradient and using adaptive spiking neurons where the internal time-constant parameters are co-trained with network weights. The Multi-Gaussian surrogate gradient is constructed to include negative slopes, inspired by the ELU activation function[16]: we find that the Multi-Gaussian surrogate gradient consistently outperforms other existing surrogate gradients. Similarly, co-training the internal time-constants of adaptive spiking neurons proved consistently beneficial. We demonstrate that these ingredients jointly improve performance to a competitive level while maintaining sparse

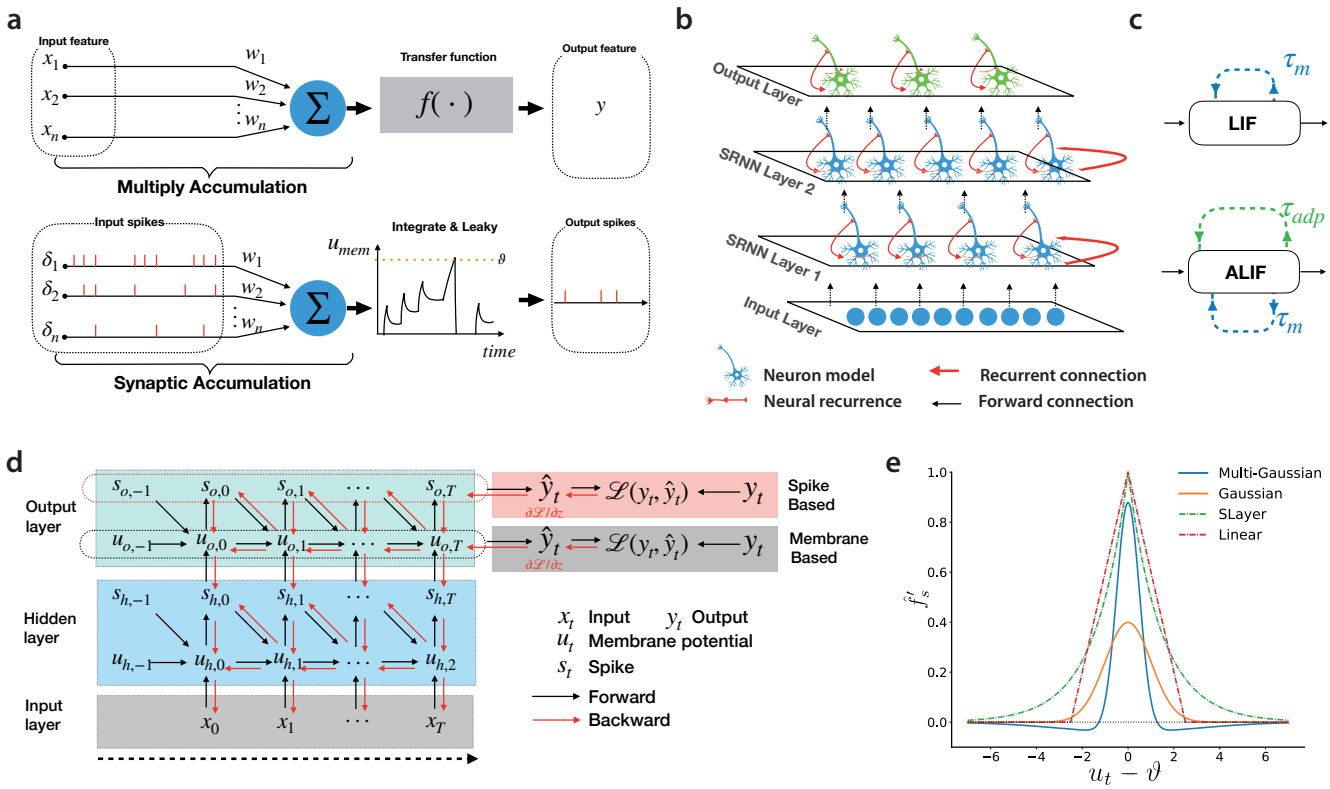


Figure 1. **a**, Top: a classical artificial neural unit computes a weighted sum over input activations and then computes an output activation from this sum using a non-linear transfer function $f()$. Time is modeled as iterated recomputation of the network graph. Bottom: Spiking neurons receive spikes that are weighted and added to the internal state (membrane potential) that further develops through time following differential equations. When the membrane potential crosses a threshold, a spike is emitted and the potential is reset. **b**, Example architecture of a Spiking Recurrent Neural Network: an input layer projects to a layer of recurrently connected spiking neurons. Recurrent layers then project to a read-out layer. Multiple recurrent layers can be connected in a feedforward fashion, here shown for two recurrent layers. **c**, The decaying threshold and membrane potential of the LIF and ALIF neurons can be modeled as an internal state induced by self-recurrency. **d**, Roll-out of the computational graph of a spiking neuron as used for backpropagation-through-time. **e**, Illustration of different surrogate gradient functions $\hat{f}_s'$ as a function of the neuron's membrane potential and threshold.

average network activity.

We demonstrate the superior performance of SRNNs for well-known benchmarks that have an inherent temporal dimension, like ECG wave-pattern classification, speech (Google Speech Commands, TIMIT), radar gesture recognition (SoLi), and classical hard benchmarks like sequential MNIST and its permuted variant. We find that the SRNNs need very little communication, with the average spiking neuron emitting a spike once every 3 to 30 timesteps depending on the task. Calculating the theoretical energy cost of computation, we then show that in SRNNs cheap Accumulate (AC) operations dominate over more expensive Multiply-Accumulate (MAC) operations. Based on relative MAC vs. AC energy cost[13,14], we argue that these sparsely spiking SRNNs have an energy advantage ranging from one to three orders of magnitude over RNNs and ANNs with comparable accuracy, depending on network and task complexity.

Spiking Recurrent Neural Networks

We focus here on multi-layer networks of recurrently connected spiking neurons, as illustrated in Figure 1**b**; variations include networks that receive bi-directional input (bi-SRNNs; Figure S2**a**).

Spiking neurons are derived from models that capture the behavior of real biological neurons[17]. While biophysical models like the Hodgkin-Huxley model are accurate, they are also costly to compute[18]. Phenomenological models like the Leaky-integrate-and-fire (**LIF**) neuron model trade-of levels of biological realism for interpretability and reduced computational cost: the LIF neuron model integrates input current in a leaky fashion and emits a spike when its membrane potential crosses its threshold from below, after which the membrane potential is reset to the reset membrane potential; the current leak is

determined by a decay time-constant τ_m.

As an exceedingly simple spiking neuron model, the LIF neuron lacks much of the complex behavior of real neurons, including responses that exhibit longer history dependency like spike-rate adaptation[18]. Bellec et al.[19] demonstrated how using a spiking neuron model that uses a generic form of adaptation improved performance in their SNNs. In this adaptive LIF (**ALIF**) neuron, the LIF neuron model is augmented with an adaptive threshold that is increased after each emitted spike, and which then decays exponentially with time-constant τ_{adp}. Both LIF and ALIF neurons can be thought of as neural units with self-recurrency, as illustrated in Figure 1**c**.

BPPT, Surrogate-Gradient and Multi-Gaussian Given a loss-function $\mathscr{L}(t|\theta)$ defined over neural activity at a particular time t, the error-backpropagation-through-time (BPTT) algorithm[15] updates network parameters θ in the direction that minimizes the loss by computing the partial gradient $\partial\mathscr{L}(t)/\partial\theta$ using the chain-rule. Here, the parameters θ include both the synaptic weights and the respective neural time-constants. In recurrently connected networks, past neural activations influence the current loss and by unrolling the network the contribution of these past activations to a current loss is accounted for. The roll-out of network activity through which the gradient is computed is illustrated in Figure 1**d**.

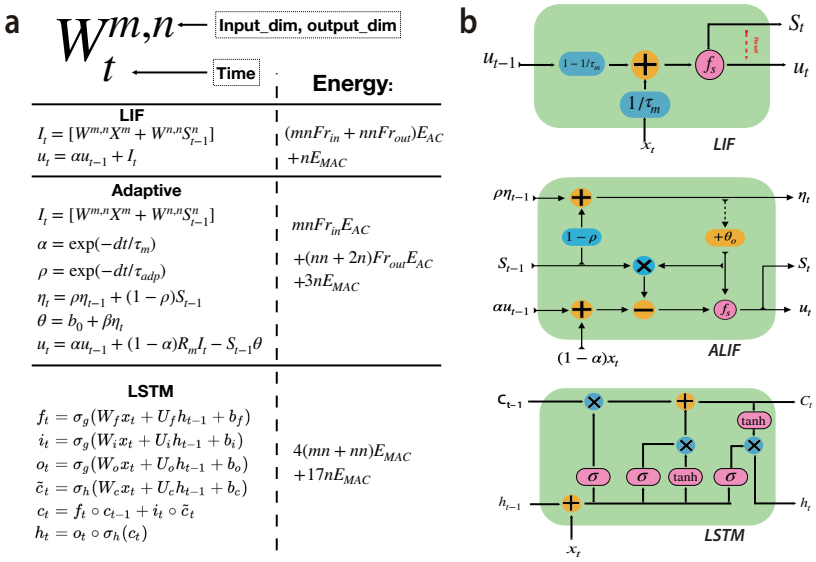


Figure 2. **a**, Theoretical energy computation of different layers. The computational complexity calculation follows[20]. **b**, LIF, ALIF and LSTM internal operation schematic.

The discontinuous nature of the spiking mechanism in spiking neurons makes it difficult to apply the chain-rule connecting the backpropagating gradient between neural output and neural input[5]; in practice, replacing the discontinuous gradient with a smooth gradient function, a "**surrogate gradient**" has proven effective[10,12,21] and has the added benefit of allowing the mapping of spiking neural networks to recurrent neural networks in optimized Deep Learning frameworks like PyTorch and Tensorflow[10]. Multiple surrogate gradient functions have been proposed and evaluated, including Gaussian, linear[19] and SLayer[6] functions; for these functions however, no significant differences in performance are reported[10].

Inspired by the ELU[16] and LeakyRelu activation function, we here define the Multi-Guassian (MG) a novel surrogate gradient $\hat{f}_s{}'(\cdot)$ comprised of a weighted sum of multiple Guassians $\mathscr{N}$ where the hyperparameter h and s are chosen such that the Multi-Guassian contains negative parts:

$$\hat{f}_s{}'(u_t|\vartheta) = (1+h)\mathscr{N}(u_t|\vartheta,\sigma^2) - h\mathscr{N}(u_t|\sigma,(s\sigma)^2) - h\mathscr{N}(u_t|-\sigma,(s\sigma)^2), \tag{1}$$

where u_t is the spiking neuron's membrane potential and ϑ its internal threshold. The negative parts of the ELU and Leaky-RELU functions are thought to alleviate the "dying ReLU" problem[22] where the sum of a neuron's input is negative for all inputs, and the neuron effectively does not participate in the network computation. The shape of the Multi-Gaussian (MG) and various other surrogate gradient functions is illustrated in Figure 1**e**.

Computational Cost To estimate the efficiency of SNNs and compare them to ANNs, we calculate the number of computations required in terms of accumulation (AC) and multiply-and-accumulate (MAC) operations[23]. We do this for an SRNN network with LIF or ALIF neurons and compare to a complex recurrent ANN structure like an LSTM[24] in Figure 2 – for other

Table 1. Comparison of SRNN performance to respective RNN and SNN state-of-the art accuracy (Acc.).

Task	Network	Method	Acc.	Task	Network	Method	Acc.
ECG	*RNN-SoTa*	Bi-LSTM	80.8%	SSC	*RNN-SoTa*	LSTM[27]	73.1%
	SRNN	**Ours**	**85.9%**		*CNN-SoTa*	CNN[27]	77.7%
SMNIST	*RNN-SoTa*	IndRNN[28]	99.5%		SNN-base	LIF[27]	50.1%
	RNN	LSTM[29]	98.2%		SRNN-SoTA	SNN[30]	60.1%
	SRNN-SoTa	LSNN[19]	96.4%		**SRNN**	**Ours**	**74.2%**
	SRNN	**Ours**	**98.7%**	SoLi	*CNN-SoTa*	CNN[31]	77.7%
PSMNIST	*RNN-SoTa*	IndRNN[28]	97.2%		RNN-SoTa	CNN+LSTM[31]	87.2%
	RNN	LSTM[29]	88%		**SRNN**	**Ours**	**91.9%**
	SRNN	**Ours**	**94.3%**	GSC	*RNN-SoTa*	Att RNN[32]	95.6%
SHD	*RNN-SoTa*	Bi-LSTM	87.2%		*CNN-SoTa*	TinySpeech[23]	92.4%
	CNN-SoTa	CNN[27]	92.4%		SNN-SoTa	LSNN[12]	91.2 %
	SNN-base	LIF[27]	71.4%		**SRNN**	**Ours**	**92.1%**
	SNN	SNN[33]	82.2%	TIMIT	*RNN-SoTa*	Bi-LSTM[34]	68.9%
	SNN-SoTa	SNN[30]	82.7%		SNN-SoTa	LSNN[19]	65.4%
	SRNN	**Ours**	**90.4%**		**Bi-SRNN**	**Ours**	**66.1%**

ANNs, see Figure S2b. In ANNs, the contribution from one neuron to another requires a MAC for every timestep, multiplying each input activation with the respective weight before adding to the internal sum. In contrast, for a spiking neuron a transmitted spike requires only an AC at the target neuron, adding the weight to the potential, and where spike inputs may be quite sparse. In addition, the spiking neuron's internal state requires updating every timestep at the cost of several MACs depending on the spiking neuron model complexity[14]. As calculating MACs is much more energetically expensive compared to ACs (e.g., 31x on 45nm CMOS[25]), the relative efficiency of SNNs is determined by the number of connections times activity sparsity and the spiking neuron model complexity. Additionally, we remark that in hardware, multiplication circuits require substantially more die area compared to addition circuits[26].

Results

Tasks Recurrent neural networks (RNNs) provide state-of-art performance in various sequential tasks that require memory[35] typically in small and compact networks, and can operate in an online fashion. We distinguish two kinds of sequential tasks: streaming tasks, where many inputs map to many specified outputs (many-to-many), and classification tasks where an input sequence maps to a single output value (many-to-one). Sequential classification tasks can additionally be computed in an online fashion, where a classification is determined for each timestep.

We selected benchmark tasks that have an inherent temporal dimension can also be computed with neural networks of modest size to fit the dynamics and constraints of spiking neural networks. For these tasks, we trained several different SRNN network architectures with various gradients, hyperparameters, and spiking neuron models and compared them to classical and state-of-the-art RNN architectures.

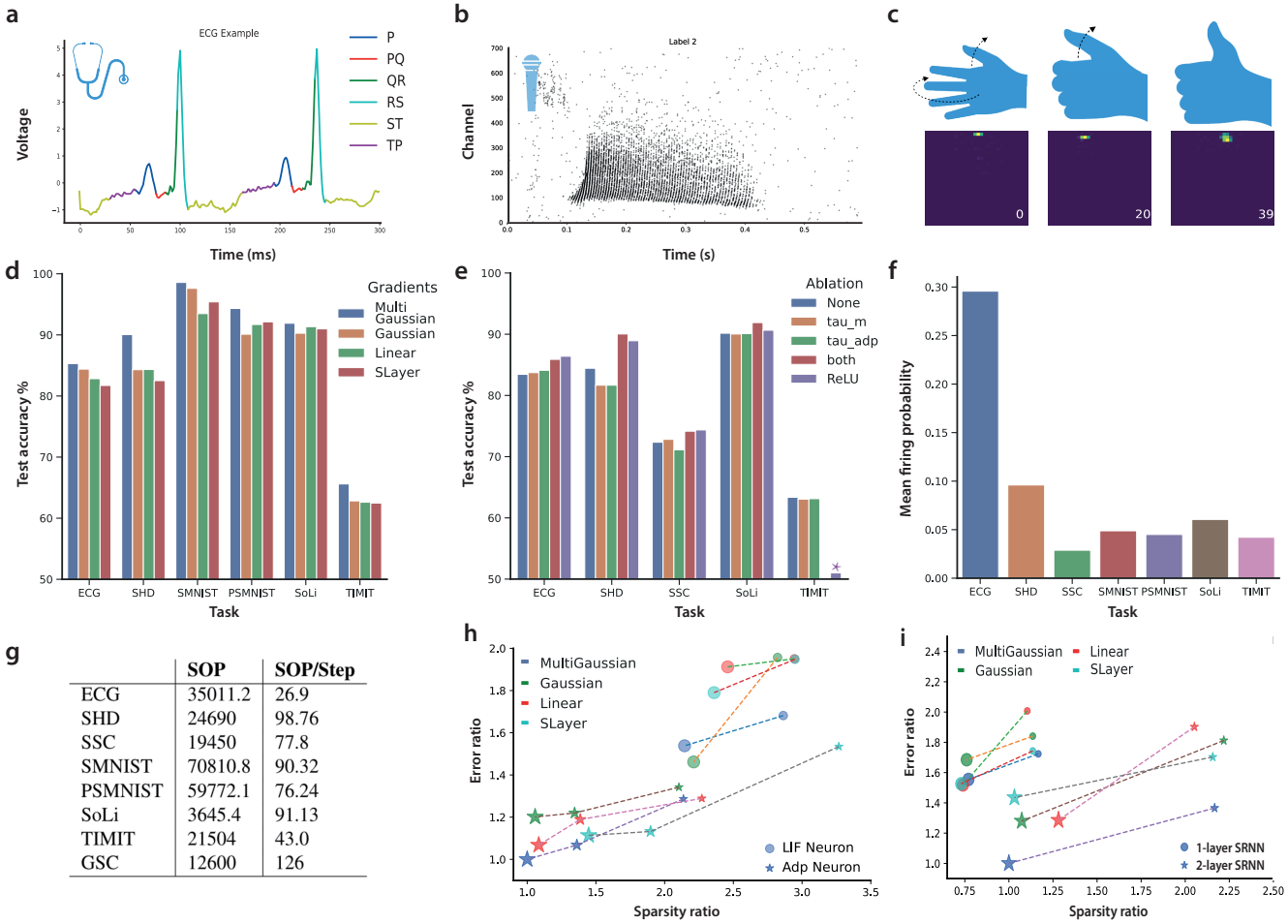
The electrocardiogram (**ECG**)[36] signal is composed of six different characteristic waveforms – P, PQ, QR, RS, ST, and TP – whose shape and duration inform clinicians on the functioning of the cardiovascular system. The task requires the continuous recognition of all six waveforms, where we use signals from the QTDB dataset[36]. The ECG-wave labeling is an online and streaming task using only past information.

The sequential- and permuted-sequential **S/PS-MNIST** datasets are standard sequence classification tasks of length 784 derived from the classical MNIST digit recognition task by presenting pixels one-at-time. The permuted version also first permutes each digit-class removing spatial information.

The Spiking Heidelberg Dataset (**SHD**) and Spiking Speech Command (**SSC**) Dataset[27] are SNN specific sequence classification benchmarks comprised of audio converted into spike trains based on a detailed ear model.

The **SoLi** dataset[31] gesture recognition task is comprised of a set of gestures in form of a sequence of radar returns collected the SoLi SoLid-state millimeter-wave radar sensor. We treat the SoLi task as both an online streaming and classification task by processing frames sequentially - we thus obtain two measures for the SoLi task, per-frame accuracy and whole sequence accuracy for streaming and classification respectively.

Both the Google Speech Commands (**GSC**) dataset[37] and the **TIMIT** dataset[38] are classical speech recognition benchmarks where for TIMIT, we compute the Frame Error Rate (FER) and where, similar to[12], we apply a bidirectional architecture such



	SOP	SOP/Step
ECG	35011.2	26.9
SHD	24690	98.76
SSC	19450	77.8
SMNIST	70810.8	90.32
PSMNIST	59772.1	76.24
SoLi	3645.4	91.13
TIMIT	21504	43.0
GSC	12600	126

Figure 3. Examples of **a**, a single ECG signal channel labeled for each timestep; **b**, the input spike-trains for the spoken number "seven" in the SHD dataset and **c**, example of gesture data – the temporal evolution of the gesture (upper row) and the corresponding RDI (bottom row); **d**, Effects of various surrogate gradients on performance. **e**, Effects of training the time constant hyperparameters τ_m and τ_{adp}; the legend denotes which hyperparameters are trained; ReLU denotes the non-spiking analog SRNN. **f**, The per timestep spike probability of the SRNNs on various tasks. **g**, Total average Spike Operations (SOP) per sample and SOPs per sample per step (timestep/frame); **h**,Effect of neuron types in terms of test accuracy and sparsity with various gradients (shown for SoLi dataset); the size of the nodes indicates the network size and the color of nodes represents the gradient type; **i**, Effect of the number of hidden recurrent layers on test accuracy and sparsity with various gradients (shown for SHD dataset).

that also future information is used to classify each frame (illustrated in Figure S2**a**). Samples from the ECG, SHD and SoLi datasets are shown in Figures 3**a-c**.

As shown in Table 1, we find that these SRNNs achieve novel state-of-the-art for Spiking Neural Networks on all tasks, exceed conventional RNNs like LSTM models, and approach or exceeds the state-of-the-art of modern RNNs. We see that SRNNs substantially close the accuracy gap (SHD, SSC, GSC) compared to non-recurrent architectures like convolutional neural networks (CNNs) and Attention-based networks – the latter networks however are typically comprised of many more neurons or parameters and cannot be computed in an online or streaming fashion.

We plot the accuracy for the various tasks using different surrogate gradients in Figure 3**d**: while we see that there is little difference between previously developed gradients like Gaussian, Linear, and SLayer, we find that the Multi-Gaussian function consistently outperforms these gradients. We also find that independently of the surrogate gradient used, training the time-constants in the Adaptive LIF neurons consistently improves performance, as shown in the ablation study in Figure 3e: not training either τ_m or τ_{adp}, or neither, reduces performance. Much of the power of the SRNNs seem to derive from their multi-layer recurrent and self-recurrent architecture. When we make the spiking neurons non-spiking by eliminating the

spiking mechanism and communicating the RELU value of the membrane potential, we find that for almost all tasks we achieve performance that slightly exceeds that of the spiking SRNNs.

The trained SRNNs communicate sparingly: most networks exhibit sparseness less than 0.1, and only the ECG task requires more spikes as it was tuned to use the smallest SRNN network (46 neurons). Sparseness of neural activity, expressed as average firing probability per timestep per neuron, is plotted in Figure 3**f**. In general, we find that increasing network sizes improves accuracy while decreasing average sparsity (Figure 3**h-i**) – though the total number of spikes used in the network increases. The total average number of spikes required per sample (SOPs) and per sample per step (SOP/step) for the highest performing SRNNs are given in Figure 3**g**.

Plotting the performance of networks using either ALIF or LIF neurons, we find that using ALIF neurons consistently improves both performance and activity sparseness in the networks (Figure 3**h**). Similarly, splitting a single large recurrent layer into two layers of recurrently connected layers in the SRNN architecture improves both performance and sparsity in the SHD task (Figure 3**i**); we observed similar improvements in the other tasks.

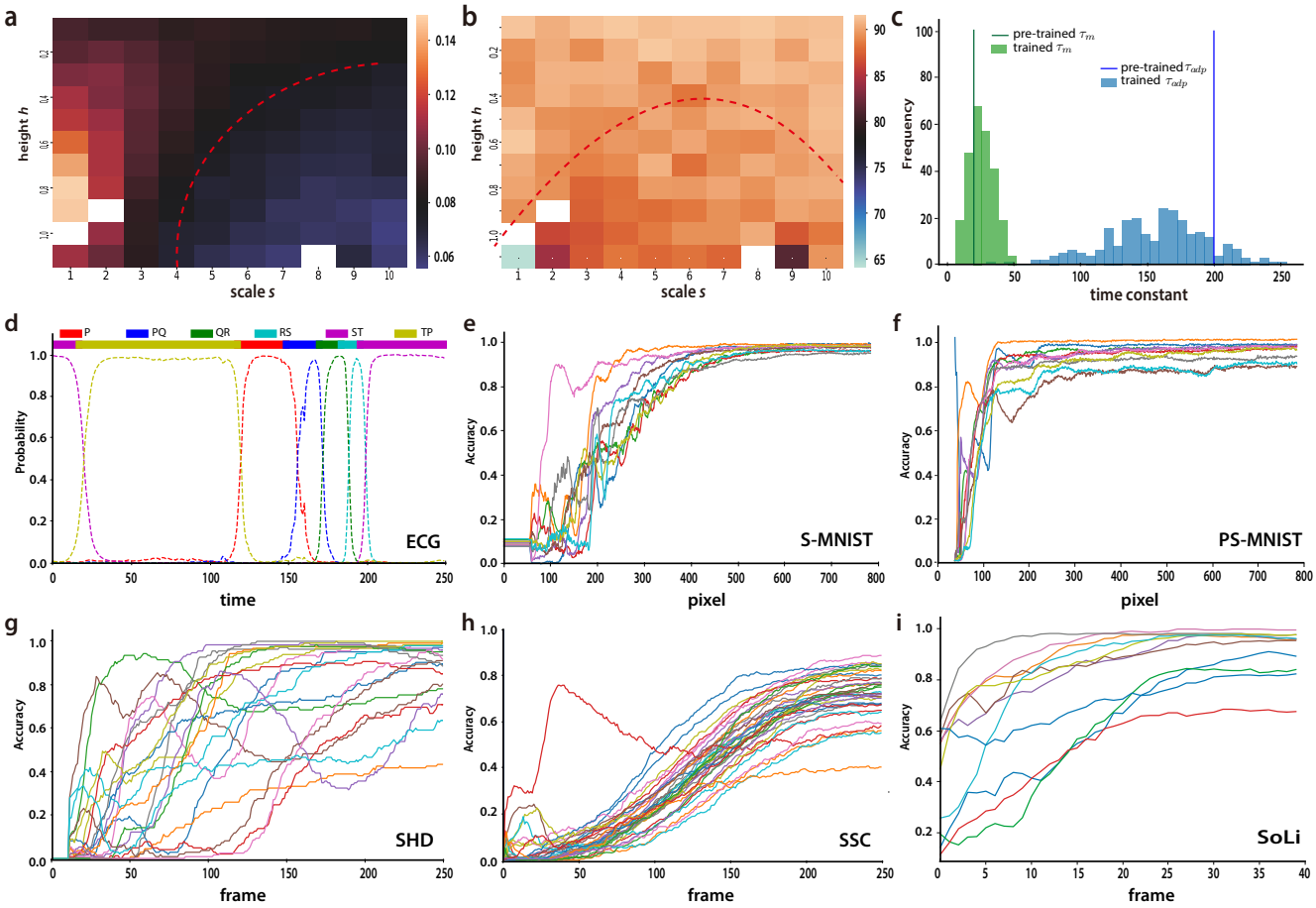


Figure 4. **a,b**, Grid search for h and s parameters for the Multi-Gaussian surrogate gradient on SoLi dataset. The red dotted line highlights solutions with high accuracy **a**, and high sparsity **b**. **c**, evolution of spiking neuron time constants evolving before and after training **d**, example of ECG streaming classification: the prediction probability of each output label is calculated from the normalized output neurons' membrane potential (dashed lines, bottom); top: the color-coded true labels. **e-i** Temporal evolution of classification accuracy for the S-MNIST recognition task **e**, the PS-MNIST task **f**, the SHD recognition task **g**,, the SSC dataset **h**,, and for the SoLi dataset **i**.

We carried out a grid search on the SoLi and SHD datasets for the h and s hyperparameters to determine the optimal parameter values for the Multi-Gaussian surrogate gradient. We find that there is a range of values where we can obtain both competitive accuracy and high sparsity (orange dotted line in Figure 4**a-b**) – we used a similar parameter search for the other tasks. The training procedure also substantially 'learns' the time-constants for the respective tasks: as shown in Figure 4**c** for the SHD task, starting from a tight distribution of time-constants, the spiking neurons in the trained network converge to using a wide variety of time-constants - the same effect is observed in the other tasks (not shown).

Table 2. Comparison of SRNN energy consumption to respective RNN and SNN state-of-the-art accuracy. Relative energy cost is calculated using the number of MACs and ACs required during inference with $Energy_{nn} = \sum_{t \in T} 3.1MAC + .1AC$[25]; fr denotes the average spiking probability in the SRNNs per timestep. The Bidirectional-LSTM network 290* contains 290 LSTM units. The accuracy in SoLi dataset is per frame accuracy. For GSC, the ReLu SRNN did not converge. ** For TIMIT, the complexity of comparably accurate networks was not available.

Task	Method	Network	Acc.	Energy/Step		Ratio	Task	Method	Network	Acc	Energy/Step		Ratio
				MAC	AC*fr						MAC	AC*fr	
ECG	Bi-LSTM	290*	80.8	181.8K		1.7Kx	SHD	Bi-LSTM	128+128+100	87.2	1.1M		1.7Kx
	ReLU	4+36+6	86.4	1.9k		18x		ReLu	128+128	88.9	142.6k		125x
	Ours (LIF)	4+36+6	49.7	42	.5k	0.5x		Ours (ALIF)	128+128	87.8	788	10.7k	**1x**
	Ours	4+36+6	85.9	90	.5k	**1x**	SoLi	LSTM	512+512	77.7	2.7M		604x
S MNIST	ReLu	64+256+256	99.0	157.3k		59x		ReLu	512+512	79.6	1.1M		246x
	Ours	64+256+256	98.7	2k	20k	**1x**		Ours	512+512	79.8	3.1k	42.4k	1x
PS MNIST	ReLu	64+256+256	93.5	157.3k		63x	GSC	ReLu	300+300		222.6k		167x
	Ours	64+256+256	94.3	2k	15.3k	**1x**		Ours	300+300	92.2	1k	10.1k	**1x**
SSC	ReLu	400+400	74.4	766.6k		236x	TIMIT**	Ours	256+61	66.1	1.6k	56.7k	**1x**
	Ours	400+400	74.2	2.4k	26.1k	**1x**							

The streaming and online nature of several of the tasks allows the network to make any-time decisions. Figure 4**d** shows the classification for the various ECG waveforms for every timestep. When a new wave is presented, there is a brief delay before this class is correctly identified. In Figures 4**e-i**, the average online classification performance is shown for the S-MNIST, PS-MNIST, SHD, SSC, and SoLi datasets. We see that the S-MNIST and PS-MNIST digits can be recognized reliably quickly, while the SSC sounds require distinctly more time. The SHD sound recognition is much more erratic, and inspection of the data shows that this is caused by the various classes being placed at different times in the sound clip. Figure 4**i** plots the accuracy as a function of the number of frames shown for the SoLi task. Most gestures can be recognized reliably already after having presented only 25 out of the 42 frames - comparing favorably with[31]: the SRNN allows decisions to be made earlier and with better accuracy.

Given the relative AC and MAC energy cost from[14,25,39] and the computational complexity calculations from Figure 2**a**, we plot in Table 2 the relative energy efficiency of the various networks. We see that for the more complex tasks, SRNNs are theoretically at least 59x more energy efficient compared to RNNs at equivalent performance levels, where for most tasks the non-spiking (ReLU) SRNN compares most favourably. More classical RNN structures like LSTMs require many more parameters and operations, often being 1000x less efficient – we also calculate similar estimates for other RNN structures in Table S2.

Discussion

We showed how multi-layered recurrent network structures are able to achieve new state-of-the-art performance for SNNs on sequential and temporal tasks. This was accomplished by using adaptive spiking neurons with learned temporal dynamics trained with backpropagation-through-time using a novel surrogate gradient, the Multi-Gaussian, where the Multi-Gaussian gradient proved to consistently outperform the other surrogate gradients. These results approach or equal the accuracy of conventional RNNs. When expressed in terms of computational operations, they demonstrate a decisive theoretical energy advantage of one to three orders of magnitude over conventional RNNs. This advantage furthermore increases for more complex tasks that required larger networks to solve accurately.

Neither the SRNNs nor the presented RNNs were optimized beyond accuracy and (for the SRNNs) sparsity: no optimizations like pruning and quantization were applied. When we compare the SRNN for the GSC task against the Attention-based CNN-network TinySpeech[23], representing the recent state-of-the-art in efficiency-optimized speech recognition, we find that at an equivalent performance level, the SRNN still requires 19.6x fewer MACs, and where, unlike TinySpeech, the SRNN operates in an online and streaming fashion (data in Table S2).

We focused on temporal or sequential problems with relatively limited input dimensionality. With RNNs, such problems can be solved with relatively small neural networks and hold direct promise for implementation in ultra-low power EdgeAI solutions. This also was the reason for emphasizing streaming or online solutions where no or fixed preprocessing and buffering is required: problems where a temporal stream first has to be segmented and where these segments are then classified greatly increase the complexity of such solutions. As we demonstrated, most classification decisions could be made early with near-optimal accuracy.

Using surrogate-gradients, the BPTT-gradient in the SRNNs can be computed using standard deep learning frameworks, where we used PyTorch[40]. With this approach, complicated architectures and spiking neuron models can be trained with state-of-the-art optimizers, regularizers, and visualization tools. At the same time, this approach is costly in terms of memory

use and training time, as the computational graph is fully unrolled over all timesteps, and the abundant spatial and temporal sparsity is not exploited in the frameworks. This also limits the size of the networks to which this approach can be applied: for significantly larger networks, either dedicated hardware and/or sparsity optimized frameworks are needed[41]. Approximations to BPTT like eProp[12] or alternative recurrent learning methods like RTRL[42] may also help alleviate this limitation.

We remark that the energy advantage of SRNNs we computed is theoretical: while the computational cost in terms of MACs is well-accepted[23,39], this measure ignores real-world realities like the presence or absence of sufficient local memory, the cost of accessing memory, and the potential cost of routing spikes from one neuron to another. In many EdgeAI applications, the energy-cost of conventional sensors may also dominate the energy equation. At the same time, the numbers we present are unoptimized in the sense that other than optimizing the surrogate gradient for both sparsity and accuracy, we did not prune the networks or applied other standard optimization and quantization techniques. Substantial improvements here should be fairly straightforward. Training parameters of spiking neuron models in the SRNNs can be extended further to approaches that include parameterized short-term plasticity[43] and more complicated spiking neuron models.

The effectiveness of adjusting time-constant parameters to the task may also have implications for neuroscience: though effective time-constants of real spiking neurons are variable and dynamic[17], the benefit of training these parameters in SRNNs suggests these neural properties may be subject to learning processes in biology.

Methods

In the SRNNs, the LIF spiking neuron is modeled as:

$$u_{t-1} = u_{t-1}(1 - S_{t-1}) + u_r S_{t-1} \tag{2}$$
$$u_t = u_{t-1}(1 - 1/\tau_m) + R_m I_t/\tau_m \tag{3}$$
$$S_t = f_s(u_t, \vartheta) \tag{4}$$

where $I_t = \sum_{t_i} w_i \delta(t_i) + I_{inj,t}$ is the input signal comprised of spikes at times t_i weighted by weight w_i and/or an injected current $I_{inj,t}$; u is the neuron's membrane potential which decays exponentially with time-constant τ_m, ϑ is the threshold, R_m is the membrane resistance (which we absorb in the synaptic weights). The function $f_s(u_t, \vartheta)$ models the spike-generation mechanism as function of the threshold ϑ, which is set to 1 when the neuron spikes and otherwise is 0 (where the approximating surrogate gradient is then $\hat{f}'_s(u_t, \vartheta)$). The value for the reset potential u_r was set to zero. The ALIF neuron is similarly modeled as :

$$u_t = \alpha u_{t-1} + (1 - \alpha) R_m I_t - \vartheta S_{t-1} \tag{5}$$
$$\eta_t = \rho \eta_{t-1} + (1 - \rho) S_{t-1} \tag{6}$$
$$\vartheta = b_0 + \beta \eta_t \tag{7}$$
$$S_t = \hat{f}_s(u_t, \vartheta), \tag{8}$$

where α, γ are parameters related to the temporal dynamics, $\alpha = \exp(-dt/\tau_m)$ and $\rho = \exp(-dt/\tau_{adp})$, ϑ is a dynamical threshold comprised of a fixed minimal threshold b_0 and an adaptive contribution $\beta \eta_t$; ρ expresses the single-timestep decay of the threshold with time-constant τ_{adp}. The parameter β is a constant that controls the size of adaptation of the threshold; we set β to 1.8 for adaptive neurons as default. Similarly, α expresses the single-timestep decay of the membrane potential with time-constant τ_m.

The SRNNs were trained using BPTT, various spiking neuron models with plastic time-constants and with various surrogate gradients. Apart from the SSC and SHD datasets, analog input values are encoded into spikes either using spikes generated by a level-crossing scheme (ECG) or by directly injecting a proportional current into the first spiking layer (S-MNIST, PS-MNIST, SoLi, TIMIT, GSC). To decode the output of the network, we used one of two methods: either spike-counting over the whole time-window, for the (P)S-MNIST task, non-spiking LIF neurons (TIMIT, SHD, SoLi, and GSC), or spiking ALIF neurons (ECG). With spike-counting, classification is decoded from the sum of the output spikes as $\hat{y} = \mathrm{softmax}(\sum_t S^t_{i,out})$ where $S^t_{i,out}$ is the spike of the output neuron i at time t. For either non-spiking LIF neurons and spiking ALIF neurons as outputs, a softmax classification is computed from the output neurons' membrane potential $u_{out,t}$ at each timestep as $\hat{y}_t = \mathrm{softmax}(u_{out,t})$. For ECG, we used spiking ALIF neurons for outputs as they performed best, which we believe is related to the fact that this is the only task where classification switches within the sample - the spiking then functions effectively as resets.

We use a standard BPTT approach[12] to minimize the cross-entropy (CE) or negative-log-likelihood (NLL) loss for each task using the Adam[44] optimizer, where we unroll all input timesteps from end to the start. The error-gradient is calculated and accumulated through all timesteps after which the weights are updated. BPTT for the spiking neurons is calculated retrogradely along with the self-recurrence circuits. As shown in Figure 1**d**, given an input sequence $X = x_0, x_1, x_2, \ldots, x_T$, and a neuron with initial states $\{u_{h,0}, u_{o,0}, S_{h,0}, S_{o,0}\}$, we obtain for each timestep $t \in \{0, T\}$ the spiking neuron states $\{u_{h,t}, S_{h,t}, u_{o,t}, S_{o,t}\}$, where

$S_{h,t}$ refers to a neuron firing-or-not in a hidden layer and $S_{o,t}$ to an output neuron (if spiking), and $u_{h,t}$ and $u_{o,t}$ denote hidden and output neurons' membrane potentials. We then obtain a classification $\hat{y}(t)$ either for each timestep or for the whole sequence $\hat{y}$ and an associated loss. In classification tasks with C classes, the prediction probability of class c – $\hat{y_c}$ is computed after having read the whole sequence, and then the loss of the network is calculated as $\mathscr{L} = \sum_{c=1}^{C} y_c \log \hat{y_c}$, where y_c is the target probability of class c. In streaming tasks (ECG, SoLi), the total loss is computed as the sum of the loss at each timestep – $\mathscr{L} = \sum_{t=1}^{T} \mathscr{L}_t$. For the BPTT-derived gradient, we compute $\frac{\partial \mathscr{L}}{\partial z} = \hat{y} - y$, and for recurrent weights W_{h2o}, we compute $\frac{\partial \mathscr{L}}{\partial W_{h2o}} = \frac{\partial \mathscr{L}}{\partial z} \sum_{t'}^{T} \frac{\partial S_{o,t'}}{\partial W_{h2o}}$, where each term can be computed at each timestep t' as $\frac{\partial S_{o,t'}}{\partial W_{h2o}} = \frac{\partial S_{o,t'}}{\partial u_{o,t'}} \frac{\partial u_{o,t'}}{\partial W_{h2o}} + \sum_{\xi=0}^{t'-1} \frac{\partial S_{o,t'}}{\partial u_{t'}} \frac{\partial u_{o,t'}}{\partial u_{o,\xi}} \frac{\partial u_{o,\xi}}{\partial W_{h2o}}$ and

$\frac{\partial S_{o,t'}}{\partial W_{h2h}} = \sum_{\xi=0}^{t'} \frac{\partial S_{o,t'}}{\partial u_{h,\xi}} \frac{\partial u_{h,\xi}}{\partial W_{h2h}}$, and where W_{h2h} refers to weights between neurons in the hidden layers, and W_{h2o} to weight between hidden and output neurons. The discontinuous spiking function enters the gradient as the term $\frac{\partial S}{\partial u}$, and here we use the differentiable surrogate gradients[10].

For the Multi-Gaussian surrogate gradient, we found effective parameter values $h = 0.15$ and $s = 6$. based on a grid search, and we set σ to 0.5. The standard surrogate gradients were defined following[10], with the Linear surrogate gradient as $\hat{f_s}'(u_t|\vartheta) = ReLU(1 - \alpha_{linear}|u_t - \vartheta|)$; the SLayer[6] gradient as $\hat{f_s}'(u_t|\vartheta) = \exp(-\alpha_{slayer}|u_t - \vartheta|)$, and the Gaussian surrogate gradient as $\hat{f_s}'(u_t|\vartheta) = \mathscr{N}(u_t|\vartheta, \sigma_G)$; for all gradients, α is positive. We optimized all surrogate gradients hyperparameters in the experiments using grid searches; in the experiments we used $\alpha_{linear} = 1.0$, $\alpha_{slayer} = 5.0$, and $\sigma_G = 0.5$.

Network initialization. Compared to ANNs, SRNNs require initializing both weight and also the spiking neurons' hyperparameters (i.e, neuron type, time constants, thresholds, starting potential). We randomly initialize the time constants following the normal distribution (μ, σ) with per-layer specific parameters given in Table S1. For all neurons, the starting value of the membrane potential is initialized with a random value distributed uniformly in the range$[0, \vartheta]$. The bias weights of the network are initialized as zero and all feedforward weights are initialized using Xavier-uniform initialization; weights for recurrent connections are initialized as orthogonal matrices. We compared networks with constant, uniform, and normal initializers for the time-constants and found that the normal initializer achieved the best performance (Figure S1).

For the various tasks, the loss-function, sequence length, maximum number of epochs, learning rate and decay schedule, and minibatch-size are specified in Table S1. Unless specified otherwise, the network architecture consists of inputs densely connected to one or more fully recurrently connected layers of spiking neurons connected to a layer of output neurons, as illustrated in Figure 1b. For the **ECG** task, the QTDB dataset[36] consists of two channels of ECG signals. We apply level-crossing encoding[45] on the normalized ECG signal to convert the original continuous values into a spike train, where each channel was transformed into two spike trains representing a value increasing or decreasing event respectively. The level crossing encoding we used is defined as

$$S_+ = \begin{cases} 1, & \text{if } x_t - x_{t-1} \geq L_+ \\ 0, & \text{otherwise} \end{cases}, \qquad S_- = \begin{cases} 1, & \text{if } x_{t-1} - x_t - \geq L_- \\ 0, & \text{otherwise} \end{cases}$$

where S_+, S_- denote spikes for the positive and negative spike-train respectively and we used $L_+ = 0.3$ and $L_- = 0.3$.

For the **Spiking Heidelberg Dataset (SHD)**, the audio records were aligned to 1s by cutting or completing with zeros. As in[27], two speakers were held out for the test dataset, and 5% of samples from other speakers were also added into the test dataset. The training dataset thus comprised of 8156 samples and test dataset contains 2264 samples. For the **Spiking Speech Command Dataset**, the speech commands were also uniformly aligned to 1s with a 250hz sampling frequency, and the dataset was randomly split into training, validation and test dataset with a ratio of 72%-8%-20% respectively. For the **SoLi dataset**, the sequence of 40 Range-Doppler images (RDI) was fed into the model frame-by-frame as input and split into training and testset as in[31]. The original RDIs have 4 channels, but we found empirically that using one channel was sufficient. For the SoLi task, the first layer of the SRNN, we use a feedforward spiking dense layer, followed by a recurrent layer. As in[31], separate networks were trained for per-frame accuracy (Acc_s) and per-sequence accuracy (Acc_c), for the streaming and classification version of the task respectively. In the **S-MNIST** tasks, the network read the image pixel by pixel; for the **PS-MNIST** task, pixels are read into the network using a sliding window of size 4 with stride 1. For both tasks, the pixel value is fed into the network directly as injected current into the neurons of the first hidden layer as a fully connected layer with its own weights. We use the **Google Speech Command** v1[37]. For preprocessing, Log Mel filters and their first and second-order derivatives are extracted from raw audio signals using Librosa[46]. For the FFTs, a window of 30ms and a hop of 10ms is used. The timestep of the simulation is 10 ms. We calculate the logarithm of 40 Mel filters coefficient using the Mel scale between 20 Hz and 4kHz. Additionally, spectrograms are normalized to ensure that the signal in each frequency has a variance of 1 across time; we then selected the first three derivative orders as three distinct input channels. The input to the SRNN is thus a sequence of 101 frames, where each frame comprises of a 40-by-3 matrix.

The **TIMIT** database contains 3696 and 192 samples in training and test data respectively. We preprocessed the original audio data as in[12] using MFCC encoding; 10% of the training dataset was randomly selected as validation dataset, and the network was trained on the remainder. Similar to bi-directional LSTMs, we use a bi-directional Adaptive SRNN for this task, illustrated in Figure S2**a**: we use two SRNN layers in the network, reading the sequence from the forward and backward direction respectively. The mean of these layer's output is then fed into the last layer, an integrator, to generate the class prediction.

References

1. Hassabis, D., Kumaran, D., Summerfield, C. & Botvinick, M. Neuroscience-inspired artificial intelligence. *Neuron* **95**, 245–258 (2017).
2. Maass, W. Networks of spiking neurons: the third generation of neural network models. *Neural Networks* **10**, 1659–1671 (1997).
3. Gerstner, W., Kempter, R., Van Hemmen, J. L. & Wagner, H. A neuronal learning rule for sub-millisecond temporal coding. *Nature* **383**, 76–78 (1996).
4. Davies, M. *et al.* Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro* **38**, 82–99 (2018).
5. Bohte, S. M., Kok, J. N. & La Poutré, J. A. Spikeprop: backpropagation for networks of spiking neurons. In *European Symposium on Artificial Neural Networks (ESANN)*, vol. 48, 17–37 (2000).
6. Shrestha, S. B. & Orchard, G. Slayer: Spike layer error reassignment in time. In *Advances in Neural Information Processing Systems*, vol. 31, 1412–1421 (2018).
7. Zenke, F. & Ganguli, S. Superspike: Supervised learning in multilayer spiking neural networks. *Neural Comput.* **30**, 1514–1541 (2018).
8. Kheradpisheh, S. R., Ganjtabesh, M., Thorpe, S. J. & Masquelier, T. Stdp-based spiking deep convolutional neural networks for object recognition. *Neural Networks* **99**, 56–67 (2018).
9. Falez, P., Tirilly, P., Bilasco, I. M., Devienne, P. & Boulet, P. Multi-layered spiking neural network with target timestamp threshold adaptation and stdp. In *International Joint Conference on Neural Networks (IJCNN)*, 1–8 (2019).
10. Neftci, E. O., Mostafa, H. & Zenke, F. Surrogate gradient learning in spiking neural networks. *IEEE Signal Process. Mag.* **36**, 61–63 (2019).
11. Wunderlich, T. C. & Pehle, C. Eventprop: Backpropagation for exact gradients in spiking neural networks. *arXiv preprint arXiv:2009.08378* (2020).
12. Bellec, G. *et al.* A solution to the learning dilemma for recurrent networks of spiking neurons. *Nat. Commun.* **11**, 1–15 (2020).
13. Sengupta, A., Ye, Y., Wang, R., Liu, C. & Roy, K. Going deeper in spiking neural networks: VGG and residual architectures. *Front. Neurosci.* **13**, 95 (2019).
14. Roy, K., Jaiswal, A. & Panda, P. Towards spike-based machine intelligence with neuromorphic computing. *Nature* **575**, 607–617 (2019).
15. Werbos, P. J. Backpropagation through time: what it does and how to do it. *Proc. IEEE* **78**, 1550–1560 (1990).
16. Clevert, D., Unterthiner, T. & Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). In *International Conference on Learning Representations (ICLR)* (2016).
17. Gerstner, W. & Kistler, W. M. *Spiking neuron models: Single neurons, populations, plasticity* (Cambridge university press, 2002).
18. Izhikevich, E. M. Simple model of spiking neurons. *IEEE Transactions on Neural Networks* **14**, 1569–1572 (2003).
19. Bellec, G., Salaj, D., Subramoney, A., Legenstein, R. & Maass, W. Long short-term memory and learning-to-learn in networks of spiking neurons. In *Advances in Neural Information Processing Systems*, 787–797 (2018).
20. Hunger, R. *Floating point operations in matrix-vector calculus* (Munich University of Technology, Inst. for Circuit Theory and Signa, 2005).
21. Bohte, S. M. Error-backpropagation in networks of fractionally predictive spiking neurons. In *International Conference on Artificial Neural Networks (ICANN)*, 60–68 (Springer, 2011).

22. Lu, L., Shin, Y., Su, Y. & Karniadakis, G. E. Dying relu and initialization: Theory and numerical examples. *arXiv preprint arXiv:1903.06733* (2019).

23. Wong, A., Famouri, M., Pavlova, M. & Surana, S. Tinyspeech: Attention condensers for deep speech recognition neural networks on edge devices. *arXiv preprint arXiv:2008.04245* (2020).

24. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural computation* **9**, 1735–1780 (1997).

25. Horowitz, M. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, 10–14 (IEEE, 2014).

26. Ludgate, P. E. On a proposed analytical machine. In *The Origins of Digital Computers*, 73–87 (Springer, 1982).

27. Cramer, B., Stradmann, Y., Schemmel, J. & Zenke, F. The heidelberg spiking data sets for the systematic evaluation of spiking neural networks. *IEEE Transactions on Neural Networks Learn. Syst.* 1–14, DOI: 10.1109/TNNLS.2020.3044364 (2020).

28. Li, S., Li, W., Cook, C., Zhu, C. & Gao, Y. Independently recurrent neural network (indrnn): Building a longer and deeper rnn. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 5457–5466 (2018).

29. Arjovsky, M., Shah, A. & Bengio, Y. Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*, 1120–1128 (2016).

30. Perez-Nieves, N., Leung, V. C., Dragotti, P. L. & Goodman, D. F. Neural heterogeneity promotes robust learning. *bioRxiv* 2020–12 (2021).

31. Wang, S., Song, J., Lien, J., Poupyrev, I. & Hilliges, O. Interacting with soli: Exploring fine-grained dynamic gesture recognition in the radio-frequency spectrum. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, 851–860 (2016).

32. de Andrade, D. C., Leo, S., Viana, M. L. D. S. & Bernkopf, C. A neural attention model for speech command recognition. *arXiv preprint arXiv:1808.08929* (2018).

33. Zenke, F. & Vogels, T. P. The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks. *Neural Comput.* **0**, 1–27, DOI: 10.1162/neco_a_01367 (2021).

34. Graves, A. & Schmidhuber, J. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks* **18**, 602–610 (2005).

35. Shewalkar, A., Nyavanandi, D. & Ludwig, S. A. Performance evaluation of deep neural networks applied to speech recognition: Rnn, lstm and gru. *J. Artif. Intell. Soft Comput. Res.* **9**, 235–245 (2019).

36. Laguna, P., Mark, R. G., Goldberg, A. & Moody, G. B. A database for evaluation of algorithms for measurement of qt and other waveform intervals in the ecg. In *Computers in cardiology 1997*, 673–676 (IEEE, 1997).

37. Warden, P. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209* (2018).

38. Garofolo, J. S. Timit acoustic phonetic continuous speech corpus. *Linguist. Data Consortium, 1993* (1993).

39. Kundu, S., Datta, G., Pedram, M. & Beerel, P. A. Spike-thrift: Towards energy-efficient deep spiking neural networks by limiting spiking activity via attention-guided compression. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 3953–3962 (2021).

40. Paszke, A. *et al.* Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, 8024–8035 (2019).

41. Zenke, F. *et al.* Visualizing a joint future of neuroscience and neuromorphic engineering. *Neuron* **109**, 571–575 (2021).

42. Zenke, F. & Neftci, E. O. Brain-inspired learning on neuromorphic substrates. *Proc. IEEE* 1–16 (2021).

43. Keijser, J. & Sprekeler, H. Interneuron diversity is required for compartment-specific feedback inhibition. *bioRxiv* DOI: 10.1101/2020.11.17.386920 (2020).

44. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR* (2015).

45. Lichtsteiner, P., Posch, C. & Delbruck, T. A 128*times*128 120 db 15μ s latency asynchronous temporal contrast vision sensor. *IEEE journal solid-state circuits* **43**, 566–576 (2008).

46. McFee, B. *et al.* librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, vol. 8, 18–25 (2015).

Acknowledgements (not compulsory)

BY is funded by the NWO-TTW Programme “Efficient Deep Learning” (EDL) P16-25. The authors gratefully acknowledge the support from the organizers of the Capo Caccia Neuromorphic Cognition 2019 workshop and Neurotech CSA, as well as Jibin Wu and Saray Soldado Magraner, for helpful discussions.

Author contributions statement

B.Y., F.C. and S.B. conceived the experiment(s), B.Y. conducted the experiment(s), B.Y., F.C. and S.B. analysed the results. All authors reviewed the manuscript.

Additional information

Competing interests. The authors declare no competing interests.

Supplementary Information

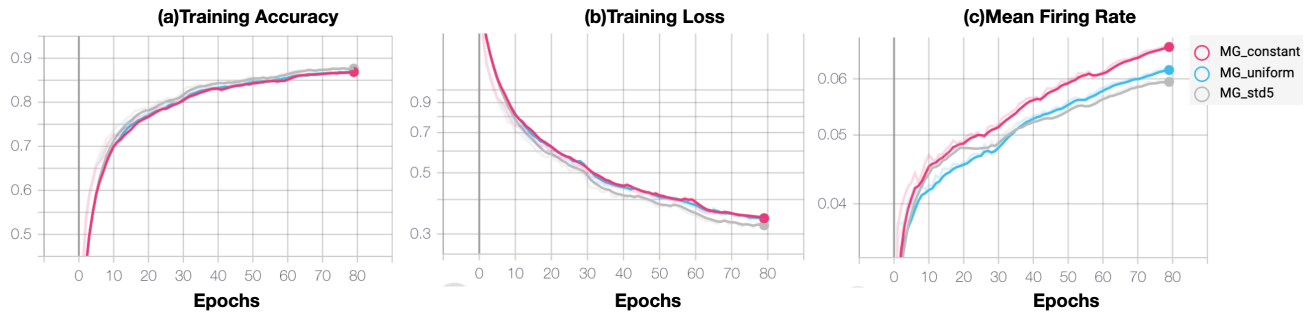


Figure S1. Effects of different time constant initialization schemes on network training and performance on the SoLi dataset. a, Training accuracy **b**, Training Loss **c**, Mean Firing rate of the network. The $\text{MG}_{constant}$ is the network where τ is initialized with a single value; for $\text{MG}_{uniform}$ the network is initialized with uniformly distributed time-constants near the single value of $\text{MG}_{constant}$; for MG_{std5}, a normal distribution with std 5.0 is used near the same single value.

Table S1. Details of parameters used for initialization and optimization for each task. Time constants refer to successive layers in the architecture; the layer output is last numbered layer – the numbers between brackets denote the mean and std used for initializing the time constants with Gaussian distribution $N(\mu;\sigma)$ where the μ is the mean of the time constant and σ is std. LIF non-spiking output layers use only the τ_m parameter. For the Loss, NLL denotes negative log-likehood and CE cross-entropy. The learning rate decay schedules denote the amount of decay (multiplicative factor) and the epoch when the decay is applied (in case of Step decays); for SMNIST and PSMNIST, a linear decay to zero is applied.

task		ECG	SHD	SSC	GSC	SMNIST	PSMNIST	SoLi	TIMIT
seq length		1301	250	250	101	784	784	40	500
time constant	$\tau_m 1$	(20;.5)	(20;5)	(20;5)	(20;5)	(20;5)	(20;5)	(20;5)	(20;5)
	$\tau_{adp} 1$	(7;.2)	(150;10)	(150;50)	(150;50)	(200;50)	(200;50)	(20;5)	(200;5)
	$\tau_m 2$	(20;.5)	(20;5)	(20;5)	(20;5)	(20;5)	(20;5)	(20;5)	(20;5)
	$\tau_{adp} 2$	(100;1)	(150;10)	(150;50)	(150;50)	(200;50)	(200;50)	(20;5)	(200;50)
	$\tau_m 3$	-	(20;5)	(20;5)	(20;5)	(20;5)	(20;5)	(20;5)	(3;1)
	$\tau_{adp} 3$	-	-	-	-	(200;50)	(200;50)	-	-
learning rate		1e-2	1e-2	1e-2	1e-2	1e-2	1e-2	2e-2	1e-2
Loss		NLL	CE	CE	CE	CE	CE	NLL	CE
minibatch size		64	64	32	64	256	256	128	16
epochs		400	20	150	70	300	300	150	200
lr decay		.5per50	.5per20	.5per10	.75per20			.75per50	.5per100
lr decay type		Step	Step	Step	Step	Linear	Linear	Step	Step

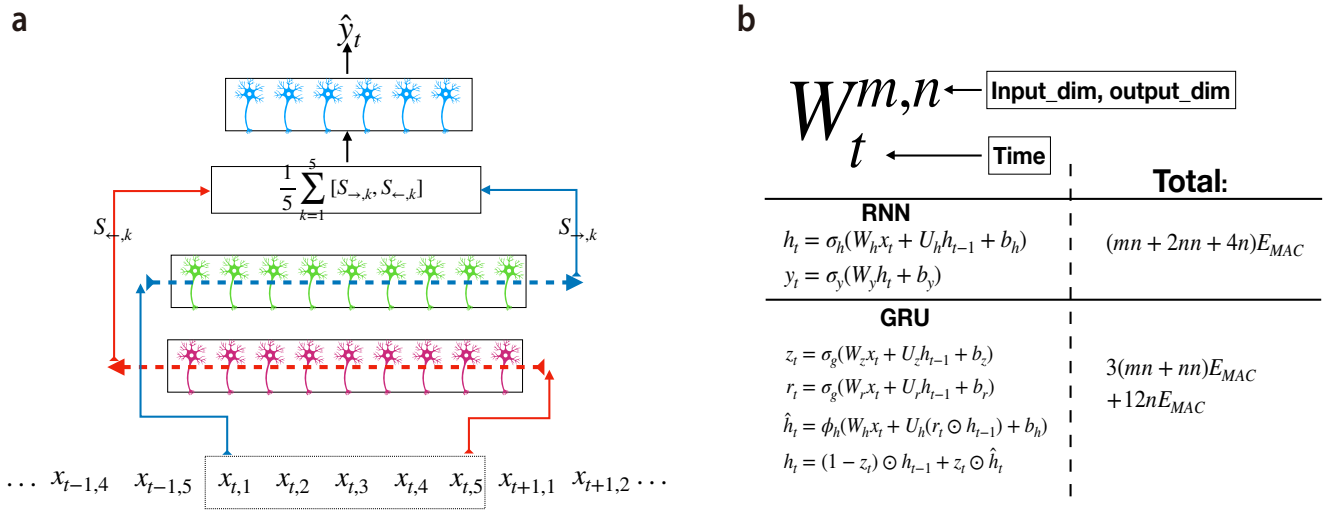


Figure S2. **a**, Bi-directional SRNN architecture. **b**, Computational cost computation of different layers for regular RNNs and GRU units. The computational complexity calculation follows[20].

Task	Method	Accuracy	Energy_{nn}/s	Energy ratio	Error ratio	Efficiency
ECG-qtdb	Adaptive SRNN	***85.9%***	325.7	1x	1x	1x
	LIF SRNN	75.5%	179.9	.55 x	1.67x	.91x
	RELU SRNN	86.4%	5784.6	17.8x	.93x	16.5x
	LSTM	78.9%	20422.8	62.7x	1.43x	89.6x
	GRU	77.3%	15400	47.2x	1.54x	72.7x
	Vanilla RNN	74.8%	9597.6	29.5x	1.71x	50.5x
	Bidirectional LSTM_{290}	80.76%	563580	1729.9x	1.31x	2266.2x
SMNIST	Adaptive SRNN	***98.7%***	8250.3	1x	1x	1x
	RELU SRNN	98.99%	487623.8	59.1x	.74x	43.4x
PSMNIST	Adaptive SRNN	***94.32%***	7775.1	1x	1x	1x
	RELU SRNN	93.47%	487623.8	62.7x	1.1x	69.0x
SHD	Adaptive SRNN	***87.81%***	3515.7	1x	1x	1x
	RELU SRNN	88.93%	442097.2	125.8x	.91x	114.5x
	Bidirectional LSTM	87.2%	3468215.52	986.5x	1.05x	1035.8x
	CNN[27]	92.4%	–	–x	–x	–x
SSC	Adaptive SRNN	74.18%	10154.1	1x	1x	1x
	RELU SRNN	74.36%	2373918.	234.0x	.99x	231.7x
	LSTM[27]	73.1%	–	–x	–x	–x
	CNN[27]	77%	–	–x	–x	–x
SoLi	Adaptive SRNN	79.8%	13,804	1x	1x	1x
	Vanilla RNN	63.6%	4101324.8	297.1x	1.80x	524.8x
	GRU	79.20%	6580531.2	476.7x	1.03x	491.1x
	LSTM	79.99%	8360972.8	605.7x	0.99x	599.6x
	RELU SRNN	79.8%	3283950.9	237.9x	1.x	237.9x
TIMIT	Adaptive SRNN	66.13%	10626.8	1x	1x	1x
	RELU SRNN	–%	11861974	175.2x	–x	–x
GSC	Adaptive SRNN	92.12%	4120.3	1x	1x	1x
	RELU SRNN	–%	11861974	167.5x	–x	–x
	CNN[23]	92.4%	80600	19.6x	0.96x	18.8x

Table S2. Performance and relative energy consumption for various models. For comparison, the average energy consumption of each input timestep and error rate of Adaptive SRNN was set to unit value. The energy/error ratio is defined as the ratio of the energy/error compared to the adaptive SRNN. We define the efficiency as the product of energy and error ratio. In the ECG task, the Bidirectional LSTM_{290} is a network with 2 bidirectional LSTM with 120 and 40 hidden neurons respectively followed by three dense layer with 100, 20 and 6 hidden neurons. For the ECG tasks, the comparison networks (LSTM, GRU, vanilla RNN and LIF SRNN) share the same architecture with 36 hidden neurons each. In the SHD task, the bidirectional LSTM is a 2-layer bidirectional network with 128 neurons each, followed by a dense layer with 100 neurons. In the SHD and SSC tasks, the temporal bin-width for the LSTM and CNN networks[27] is set to 10ms with a sampling frequency of 100Hz. For SoLi, all networks (vanilla RNN, GRU, and LSTM) use the same network structure as the SRNN, with a recurrent layer with 512 neurons followed by a dense layer with 512 neurons. All CNNs in the table read the whole sequence at once, and their computation cost is computed as the average cost per timestep. Dashes denote where values are not available either because of lacking network architecture details or lack of convergence (ReLu SRNN for TIMIT and GSC).