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

Yin, Bojian; Corradi, Federico; Bohté, Sander M.

Published in:
Nature Machine Intelligence

DOI:
10.1038/s42256-021-00397-w

nature machine intelligence

ARTICLES

https://doi.org/10.1038/s42256-021-00397-w

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]

Inspired by detailed modelling of biological neurons, spiking neural networks (SNNs) are investigated as biologically plausible and high-performance models of neural computation. The sparse and binary communication between spiking neurons potentially enables powerful and energy-efficient neural networks. The performance of SNNs, however, has remained lacking compared with artificial neural networks. Here we demonstrate how an activity-regularizing surrogate gradient combined with recurrent networks of tunable and adaptive spiking neurons yields the state of the art for SNNs on challenging benchmarks in the time domain, such as speech and gesture recognition. This also exceeds the performance of standard classical recurrent neural networks and approaches that of the best modern artificial neural networks. As these SNNs exhibit sparse spiking, we show that they are theoretically one to three orders of magnitude more computationally efficient compared to recurrent neural networks with similar performance. Together, this positions SNNs as an attractive solution for AI hardware implementations.

The success of brain-inspired deep learning in AI is naturally refocusing 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 analogue value (Fig. 1a). Spiking neurons, as more detailed neural abstractions, are theoretically more powerful than analogue neural units[2] as they allow the relative timing of individual spikes to carry important information. A real-world example in nature is the efficient sound localization in animals such as Barn Owls 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].

Numerous approaches to learning in spiking neural networks (SNNs) have been developed since their introduction[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 the loss function with respect to the weights have to deal with the discontinuous nature of the spiking mechanism inside neurons. Local linearized approximations such as 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 such as CIFAR10 and (Fashion) MNIST, and demonstrating improved performance on temporal tasks such as TIMIT speech recognition[12]. Still, SNNs have struggled to demonstrate a clear advantage over classical artificial neural networks (ANNs)[13,14].

Here we introduce a novel approach to spiking recurrent neural networks (SRNNs)[15], networks that include recurrently connected layers of spiking neurons (Fig. 1b). We demonstrate how these networks can be trained to high performance on hard benchmarks to exceed the current state of the art in SNNs for all but one benchmark, and approaching or surpassing the state of the art in classical recurrent artificial neural networks. High performance in SRNNs is achieved by applying backpropagation through time (BPTT)[16] to spiking neurons using a novel multi-Gaussian surrogate gradient and 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 (similar to the gradient of the sigmoid-like dSilu activation function[17,18]); we find that it consistently outperforms other existing surrogate gradients. Similarly, co-training the internal time-constants of adaptive spiking neurons always proved to be beneficial. We demonstrate that these ingredients jointly improve performance to a competitive level while maintaining sparse average network activity.

We demonstrate the superior performance of SRNNs for well-known benchmarks that have an inherent temporal dimension, such as electrocardiogram (ECG) wave-pattern classification, speech (Google Speech Commands, GSC; TIMIT), radar gesture recognition (project Soli) and classical hard benchmarks such as 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 cheap accumulate operations dominate over more expensive multiply-and-accumulate (MAC) operations in SRNNs. Based on relative MAC versus accumulate 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 the network and task complexity.

SRNNs. We focus here on multilayer networks of recurrently connected spiking neurons, as illustrated in Fig. 1b; variations include spiking recurrent neural networks that receive bidirectional input (bi-SRNNs; Extend Data Fig. 2a).

[1]CWI, Machine Learning group, Amsterdam, the Netherlands. [2]Stichting Interuniversitair Micro-Elektronica Centrum (IMEC) Nederland, Eindhoven, the Netherlands. [3]Univ of Amsterdam, Faculty of Science, Amsterdam, the Netherlands. [4]Rijksuniversiteit Groningen, Faculty of Science and Engineering, Groningen, the Netherlands. ✉e-mail: byin@cwi.nl

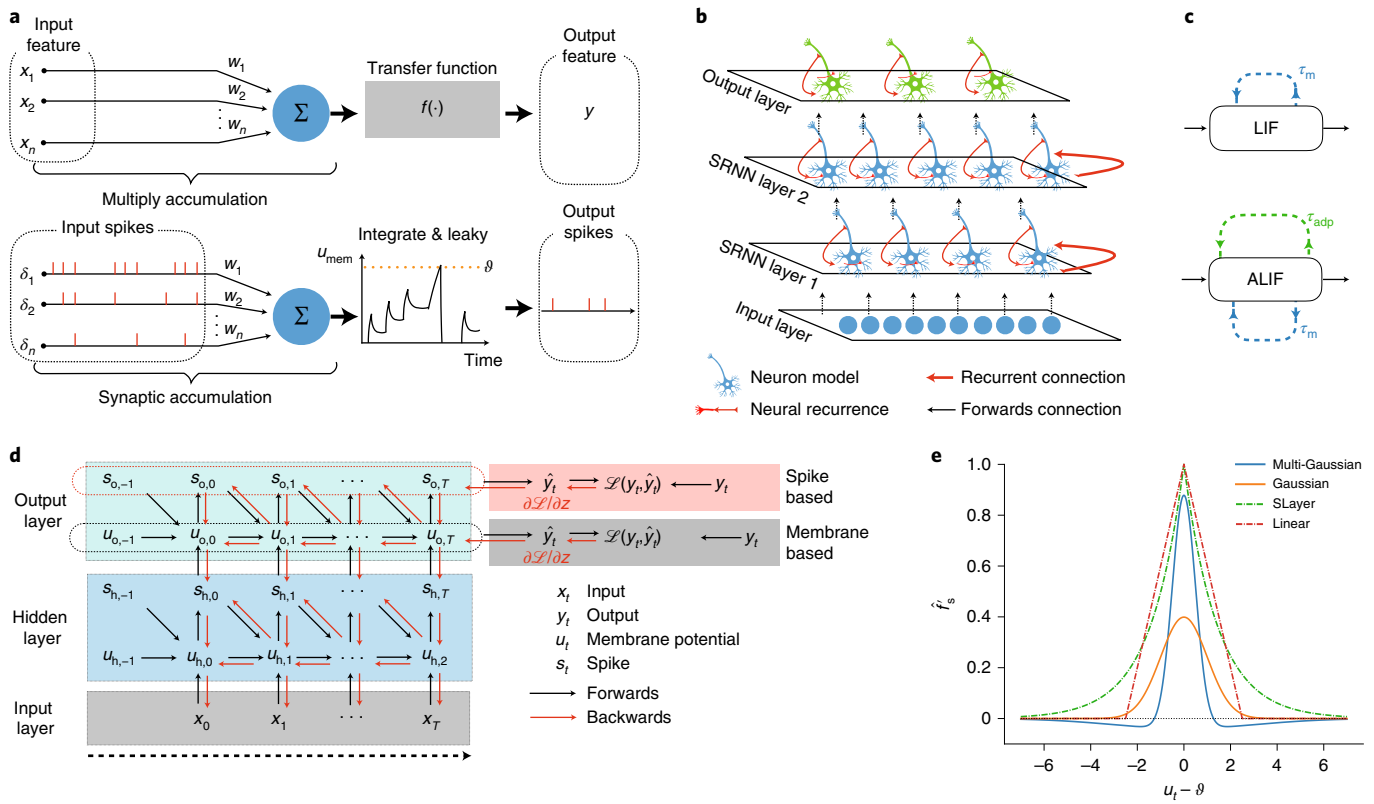


Fig. 1 | Illustration of processing and learning in networks of spiking neurons. a, Top: a classical artificial neural unit computes a weighted sum over input activations and then computes an output activation from this sum using a non-linear transfer function. Time is modelled as iterated recomputation 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 an SRNN: 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 feedforwards fashion, which is shown here for two recurrent layers. **c**, The decaying threshold and membrane potential of the LIF and ALIF neurons can be modelled as an internal state induced by self-recurrency. **d**, Roll-out of the computational graph of a spiking neuron as used for BPTT for a sequence $t = 0 \ldots T$. **e**, An illustration of different surrogate gradient functions $\hat{f}_s'$ as a function of the neuron's membrane potential and threshold, where the multi-Gaussian is parameterized as in the experiments below ($s = 6, h = 0.15$).

Spiking neurons are derived from models that capture the behaviour of real biological neurons[19]. Although biophysical models such as the Hodgkin–Huxley model are accurate, they are also costly to compute[20]. Phenomenological models such as the leaky integrate-and-fire (LIF) neuron model trade levels of biological realism for interpretability and reduced computational cost: LIF 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 behaviour of real neurons, including responses that exhibit longer history dependency such as spike-rate adaptation[20]. Bellec and colleagues[21] 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 increases after each emitted spike and 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 Fig. 1c.

BPTT, surrogate-gradient and multi-Gaussian. Given a loss-function $\mathcal{L}(t|\theta)$ defined over neural activity at a particular time t, the BPTT algorithm[16] updates network parameters θ in the direction that minimizes the loss by computing the partial gradient $\partial\mathcal{L}(t)/\partial\theta$ using the chain-rule. Here θ includes 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 the current loss is accounted for. The roll-out of network activity, through which the gradient is computed, is illustrated in Fig. 1d.

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[10,12,22] and has the added benefit of allowing the mapping of SNNs to RNNs in optimized deep learning frameworks such as PyTorch and Tensorflow[10]. Multiple surrogate gradient functions have been proposed and evaluated, including Gaussian, linear[21] and SLayer[6] functions; however, no notable differences in performance are reported for these functions[10].

We here define the multi-Gaussian as a novel surrogate gradient $\hat{f}_s'(\cdot)$ comprising a weighted sum of multiple Gaussians $\mathcal{N}$, where the hyperparameters h and s are chosen such that the multi-Gaussian contains negative parts:

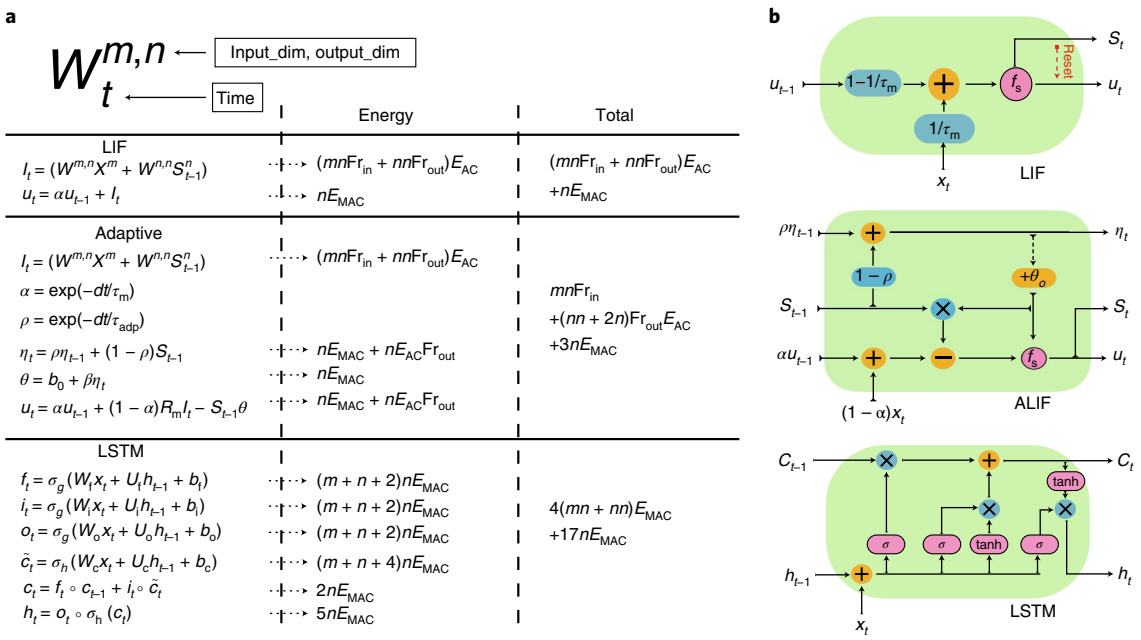


Fig. 2 | Complexity calculation. a, Theoretical energy computation of different layers. The computational complexity calculation follows Hunger[50]. Given the equations for evaluating a single neural LIF, ALIF or LSTM unit (left), complexity is computed for a single recurrently connected layer where each neuron receives n feedforward inputs with average spike probability, Fr_{in}, and m recurrent inputs with average spike probability, Fr_{out}; E_{AC} and E_{MAC} denote the energy cost for accumulate (AC) and MAC operations, respectively. **b**, LIF, ALIF and LSTM internal operation schematic.

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

where u_t is the spiking neuron's membrane potential, σ is the width of the Gaussian and ϑ is its internal threshold. The multi-Gaussian surrogate gradient is inspired by the dSilu[17] activation function, which was shown to outperform the standard sigmoidal activation function both for accuracy and learning speed, and which has a derivative similar to the multi-Gaussian. The negative parts of multi-Gaussian gradient effectively regularize activity, as they penalize both relatively large inputs and small inputs[18]. The gradient function thus aids the SNN in achieving high accuracy with sparse neural activity. The shape of the multi-Gaussian and various other surrogate gradient functions is illustrated in Fig. 1e.

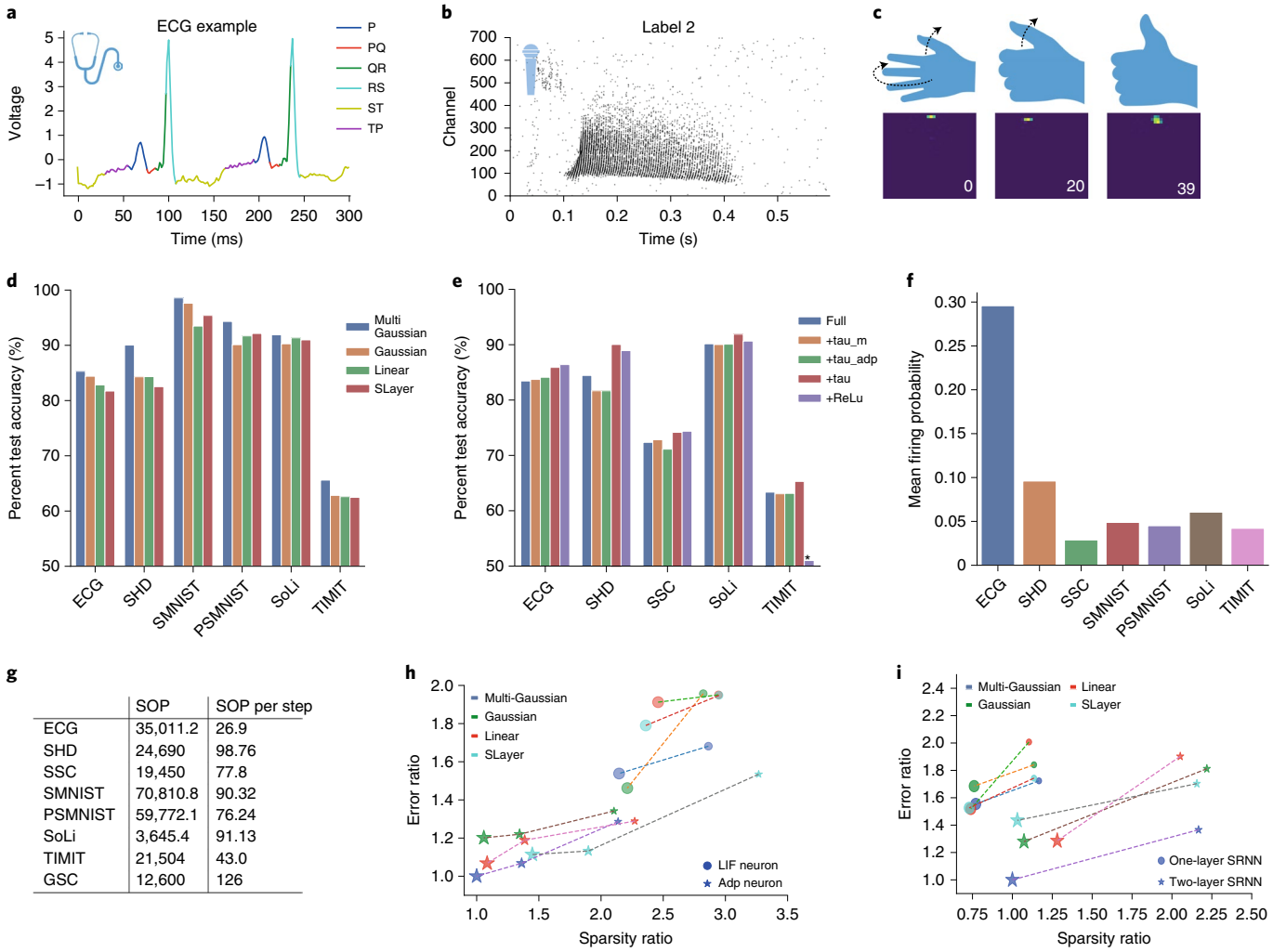
Computational cost. To estimate the efficiency of SNNs and compare them with ANNs, we calculate the number of computations required in terms of accumulation (AC) and MAC operations[23]. We do this for an SRNN network with LIF or ALIF neurons and compare it with a complex recurrent ANN structure such as a long short-term memory (LSTM) network[24] (Fig. 2; see Extended Data Fig. 2b for other ANNs). 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 it to the internal sum. By contrast, for a spiking neuron, a transmitted spike requires only an accumulate at the target neuron, adding the weight to the potential, and where spikes may be quite sparse. Furthermore, the spiking neuron's internal state needs to be updated at every timestep at the cost of several MACs depending on the spiking neuron model complexity[14]. As it is much more energetically expensive to calculate MACs than ACs (for example, 31-times more expensive on a 45 nm complementary metal–oxide–semiconductor[25]), the relative efficiency of SNNs is determined by the number of connections multiplied by activity sparsity and the spiking neuron model complexity. We also remark that on digital hardware, multiplication circuits require substantially more die area than addition circuits[26].

Results

Tasks. Recurrent neural networks provide state of the art performance in various sequential tasks that require memory[27] (typically in small and compact networks) and can operate in an online fashion. We distinguish two kinds of sequential tasks: (1) streaming tasks, where many inputs map to many specified outputs (many-to-many); and (2) classification tasks, where an input sequence maps to a single output value (many-to-one). Sequential classification tasks can also be computed in an online fashion, where classification is determined for each timestep.

We selected benchmark tasks with an inherent temporal dimension that can also be computed with RNNs of modest size to fit the dynamics and constraints of SNNs. For these tasks, we trained several different SRNN network architectures with various gradients, hyperparameters and spiking neuron models and compared them with classical and state-of-the-art RNN architectures. Hyperparameters were selected using threefold cross-validation on the training data.

The ECG[28] 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[28]. The ECG-wave labelling 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 a time. The permuted version also first permutes each digit-class removing spatial information. The Spiking Heidelberg Digits (SHD) and Spiking Speech Command (SSC) datasets[29] are SNN-specific sequence classification benchmarks that comprise audio data converted into spike trains based on a detailed ear model.



g

	SOP	SOP per step
ECG	35,011.2	26.9
SHD	24,690	98.76
SSC	19,450	77.8
SMNIST	70,810.8	90.32
PSMNIST	59,772.1	76.24
SoLi	3,645.4	91.13
TIMIT	21,504	43.0
GSC	12,600	126

Fig. 3 | Details of task performance. a–c, Examples of a single ECG signal channel labelled for each timestep (**a**); the input spike-trains for the spoken number seven in the SHD dataset (**b**); and an example of gesture data (**c**), in which the top row shows the temporal evolution of the gesture and the bottom row shows the corresponding range-doppler image. **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, whereas ReLU denotes the non-spiking analogue SRNN. **f**, The per-timestep spike probability of the SRNNs on various tasks. **g**, Total average spike operations (SOPs) per sample and SOPs per sample per step (timesteps per frame). **h**, The effect of neuron types in terms of test accuracy and sparsity with various gradients (shown for the Soli dataset). The size of the nodes indicates the network size, whereas the colours of the nodes represent the gradient type. **i**, Effect of the number of hidden recurrent layers on test accuracy and sparsity with various gradients (shown for SHD dataset).

The Soli dataset[30] gesture recognition task comprises a set of gestures where each gesture is measured as a sequence of radar returns collected from the solid-state millimetre-wave radar sensor (Soli). We treat the Soli task as both an online streaming and classification task by processing frames sequentially; we thus obtain two measures, per-frame accuracy for streaming and whole sequence accuracy for classification.

Both the GSC[31] and TIMIT datasets[32] are classical speech recognition benchmarks where, for TIMIT, we compute the frame error rate (FER) and where, similar to ref. [12], we apply a bidirectional architecture such that future information is also used to classify each frame (illustrated in Extended Data Fig. 2a). Samples from the ECG, SHD and Soli datasets are shown in Fig. 3a–c.

As shown in Table 1, we find that these SRNNs achieve state of the art performance for SNNs on all but one task, exceed conventional RNNs such as LSTM models, and approach or exceed the state of the art for modern RNNs. For GSC, we exceed the SNN state of the art for recurrent and online processing and approach the non-streaming result of ref. [33]. Moreover, we see that SRNNs considerably close the accuracy gap (SHD, SSC, GSC) on non-recurrent architectures such as convolutional neural networks (CNNs) and attention-based networks; these networks, however, typically comprise 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 Fig. 3d: although we see that there is little difference between previously developed gradients such as Gaussian, Linear and SLayer, we find that the multi-Gaussian function consistently outperforms these gradients. To better understand why the multi-Gaussian is beneficial, we removed either the left or right negative part of the gradient for comparison. We found consistently that both performance and sparseness improved for both parts (Extended Data Table 2, and Extended Data Figs. 3 and 4), demonstrating that the negative parts of the multi-Gaussian act as effective regularizers.

Table 1 | Comparisons of SRNN performances with 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[29]	73.1%
	SRNN	**This work**	**85.9%**		CNN-SoTa	CNN[29]	77.7%
S-MNIST	RNN-SoTa	IndRNN[44]	99.5%		SNN-base	LIF[29]	50.1%
	RNN	LSTM[45]	98.2%		SRNN-SoTA	SNN[46]	60.1%
	SRNN-SoTa	LSNN[21]	96.4%		**SRNN**	**This work**	**74.2%**
	SRNN	**This work**	**98.7%**	Soli	CNN-SoTa	CNN[30]	77.7%
PS-MNIST	RNN-SoTa	IndRNN[44]	97.2%		RNN-SoTa	CNN+LSTM[30]	87.2%
	RNN	LSTM[45]	88%		**SRNN**	**This work**	**91.9%**
	SRNN	**This work**	**94.3%**	GSC	RNN-SoTa	Att RNN[47]	95.6%
SHD	RNN-SoTa	Bi-LSTM	87.2%		CNN-SoTa	SCNN[33]	94.5%
	CNN-SoTa	CNN[29]	92.4%		SNN-SoTa	LSNN[12]	91.2 %
	SNN-base	LIF[29]	71.4%		**SRNN**	**This work**	**92.1%**
	SNN	SNN[48]	82.2%	TIMIT	RNN-SoTa	Bi-LSTM[49]	68.9%
	SNN-SoTa	SNN[46]	82.7%		SNN-SoTa	LSNN[21]	65.4%
	SRNN	**This work**	**90.4%**		**Bi-SRNN**	**This work**	**66.1%**

Bold font indicates this work.

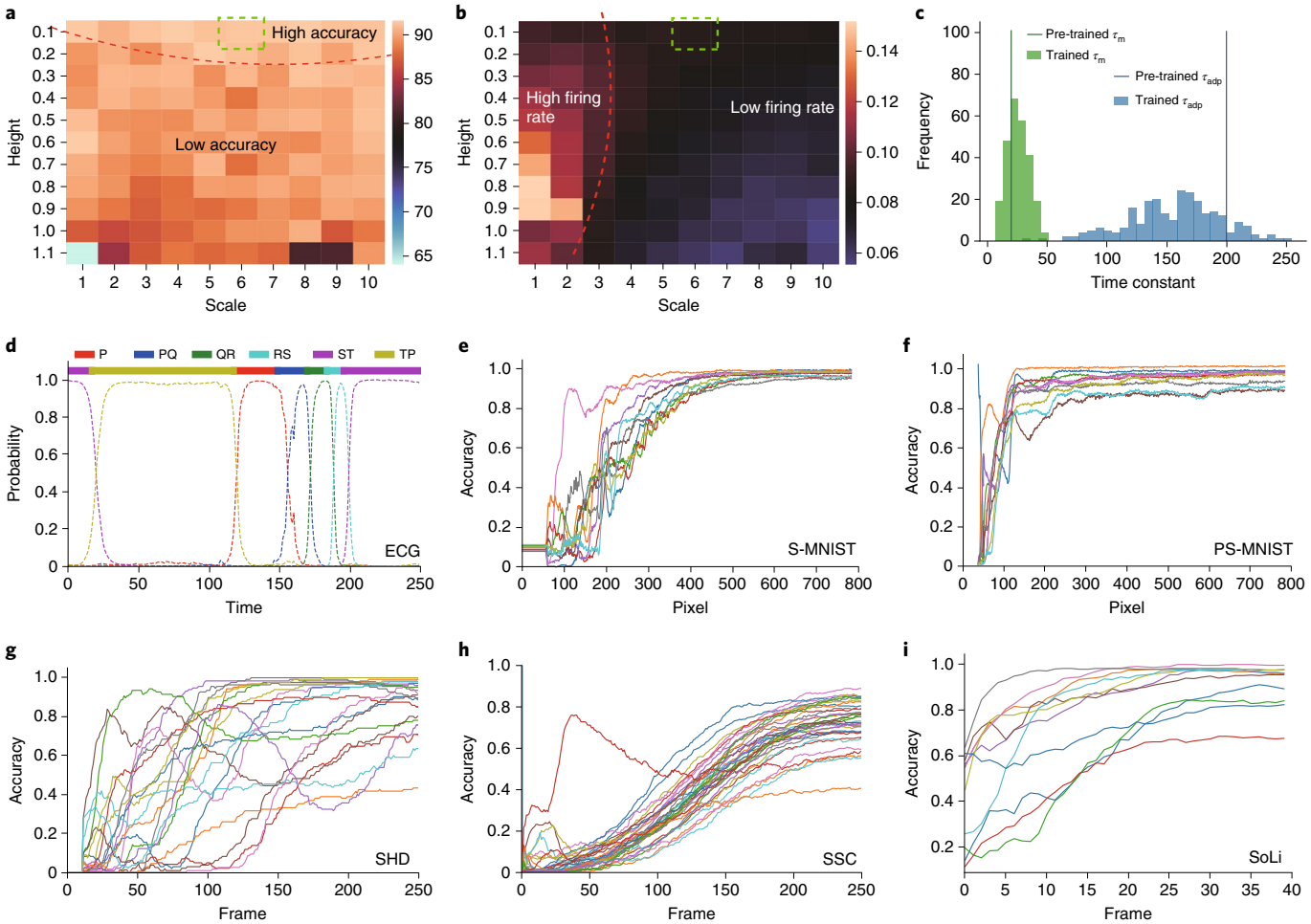


Fig. 4 | Learning trade-offs and classification latency. a,**b**, Grid search for *h* (Height) and *s* (Scale) for the multi-Gaussian surrogate gradient on the Soli dataset. The dotted line demarcates the top-left area of solutions with high accuracy (>0.91) (**a**) and high firing sparsity (>0.09) (**b**). The dashed green box denotes the selected *h* and *s* values. **c**, Evolution of spiking neuron time constants evolving before and after training. **d**, An example of ECG streaming classification: the prediction probability of each output label is calculated from the normalized output neurons's membrane potential (dashed lines, bottom). Top, the colour-coded true labels. **e**–**i**, Temporal evolution of classification accuracy for the S-MNIST recognition (**e**), PS-MNIST (**f**) and SHD recognition (**g**) tasks, the SSC dataset (**h**) and for the Soli dataset (**i**).

As shown in the ablation study in Fig. 3e, we also find that, independent of the surrogate gradient used, training the time-constants in the ALIF neurons consistently improves performance; not training τ_m or τ_{adp}—or training neither—reduces performance. Much of the power of the SRNNs seem to derive from their multilayer recurrent and self-recurrent architecture. When we make the spiking neurons non-spiking by eliminating the spiking mechanism and communicating the rectified linear unit (ReLU) value of the membrane potential, for almost all tasks we achieve performances that slightly exceed that of the spiking SRNNs.

The trained SRNNs communicate sparingly: most networks exhibit sparseness of 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—which is expressed as the average firing probability per timestep per neuron—is plotted in Fig. 3f. In general we find that increasing network size improves accuracy while decreasing the average sparsity (Fig. 3h,i), although 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 per step) for the highest performing SRNNs are given in Fig. 3g. We also evaluated to what degree the internal recurrency of spiking neurons contributes compared to the intralayer recurrent connectivity: we find that the addition of intralayer recurrent connections consistently improves accuracy (see Extended Data Table 2 in the Supplementary Information).

Plotting the performance of networks using either ALIF or LIF neurons, we find that ALIF neurons consistently improve both performance and activity sparseness in the networks (Fig. 3h). 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 (Fig. 3i), with similar improvements in the other tasks.

We performed 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 using cross-validation. We find that there is a range of values where we can obtain both competitive accuracy and high sparsity (areas to the top left of the orange dotted line in Fig. 4a,b). We used a similar hyperparameter search for the other tasks using selected values only from the high-accuracy/low activity area identified here; the training procedure also learns the time constants for the respective tasks. As shown in Fig. 4c 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).

The streaming and online nature of several of the tasks allows the network to make any-time decisions. Figure 4d shows the classification of 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 Fig. 4e–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, whereas 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 4i 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 favourably with ref. [30]: the SRNN allows decisions to be made earlier and with better accuracy.

Given the relative accumulate and MAC energy cost from refs. [14,25,34] and the computational complexity calculations from Fig. 2a, 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 59-times more energy efficient than RNNs at equivalent performance levels, where for most tasks the non-spiking (ReLU) SRNN compares most favourably. More classical RNN structures such as LSTMs require many more parameters and operations, often being 1,000-times less efficient—we also calculate similar estimates for other RNN structures in Extended Data Table 1.

Discussion

We showed how multilayered 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 BPTT using a novel surrogate gradient, the multi-Gaussian, where the multi-Gaussian gradient consistently outperformed the other surrogate gradients. These results approach or equal the accuracy of conventional RNNs, where the non-spiking ReLU-SRNNs consistently slightly outperformed the spiking version, demonstrating the effectiveness of the SRNN network architecture. 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.

The multi-Gaussian gradient was inspired by a sigmoid-style saturating activation function developed for standard artificial neurons, the dSilu, which has a similarly shaped gradient. As with the dSilu, we also find that the negative parts of the gradient help improve accuracy, and in the SRNN also sparseness. The latter suggests that the negative parts of the gradient act as effective regularizers.

Neither the SRNNs nor the presented RNNs were optimized beyond accuracy and (for the SRNNs) sparsity: no optimizations such as pruning and quantization were applied. When we compare the SRNN for the GSC task with the attention-based CNN-network TinySpeech[23] (the recent state of the art in efficiency-optimized speech recognition), we find that at an equivalent performance level, the SRNN still requires 19.6-times fewer MACs, and where, unlike TinySpeech, the SRNN operates in an online and streaming fashion (Extended Data Table 1).

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. We showed that most classification decisions could be made early with near-optimal accuracy.

The datasets discussed here were all selected for being amenable to streaming and online processing by SRNNs with very limited preprocessing; for example, calculating log Mel filters. In preliminary work, the use of conventional convolutional network layers to extract useful features proved helpful for simple subsequent layers of spiking neurons[35]. We similarly find (with a hybrid CNN-SRNN we obtained an accuracy of 97.91% on the DVS128 dataset and 96.5% on the GSC dataset, with the CNN-SRNN code available at https://github.com/byin-cwi/Efficient-spiking-networks/tree/main/DVS128) that deep preprocessing improves accuracy considerably for tasks such as GSC and also the DVS128 dataset[36], where SRNNs exhibited scores exceeding those reported by refs. [33,35]. This suggests that for even larger problems than those studies here, deep preprocessing holds much promise when balanced against the impact on complexity and energy requirement and also on the ability to process event-based streaming data.

Using surrogate gradients, the BPTT gradient in the SRNNs can be computed using standard deep learning frameworks, where we used PyTorch[37]. The code is available at https://github.com/

Table 2 | Comparison of SRNN energy consumption with respective RNN and SNN state-of-the-art accuracy

Task	Method	Network	Acc.	Energy/step		Ratio	Task	Method	Network	Acc	Energy/step		Ratio
				MAC	AC×fr						MAC	AC×fr	
ECG	Bi-LSTM	290[a]	80.8	181,800		×1,700	SHD	Bi-LSTM	128+128+100	87.2	1,100,00		×1,700
	ReLU	4+36+6	86.4	1,900		×18		ReLU	128+128	88.9	142,600		×125
	Ours (LIF)	4+36+6	49.7	42	500	×0.5		Ours (ALIF)	128+128	90.4	788	10,700	**×1**
	Ours	4+36+6	85.9	90	500	**×1**	Soli	LSTM	512+512	77.7	2,700,00		×604
S-MNIST	ReLU	64+256+256	99.0	157,300		×59		ReLU	512+512	79.6	1,100,00		×246
	Ours	64+256+256	98.7	2,000	20,000	**×1**		Ours	512+512	79.8	3,100	42,400	×1
PS-MNIST	ReLU	64+256+256	93.5	157,300		×63	GSC	ReLU	300+300		222,600		×167
	Ours	64+256+256	94.3	2,000	15,300	**×1**		Ours	300+300	92.2	1,00	10,100	**×1**
SSC	ReLU	400+400	74.4	766,600		×236	TIMIT[b]	Ours	256+61	66.1	1,600	56,700	**×1**
	Ours	400+400	74.2	2,400	26,100	**×1**							

The relative energy cost is calculated using the number of MACs and ACs required during inference, where Ratio, the relative energy ratio, is computed using 1MAC = 31AC (ref. [25]). The average spiking probability in the SRNNs per timestep is denoted by fr. The accuracy in Soli dataset is per frame accuracy. For GSC, the ReLU SRNN did not converge. [a]The bidirectional-LSTM network contains 290 LSTM units. [b]For TIMIT, the complexity of comparably accurate networks was not available.

byin-cwi/Efficient-spiking-networks. 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, precluding online and on-chip learning. Furthermore, 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 considerably larger networks, either dedicated hardware and/or sparsity optimized frameworks are needed[38]. Approximations to BPTT such as eProp[12] or alternative recurrent learning methods such as RTRL[39] may also help alleviate this limitation.

We remark that the energy advantage of SRNNs we computed is theoretical: although the computational cost in terms of MACs is well-accepted[23,34], this measure ignores real-world realities such as 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[40] 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[19], 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 modelled as:

$$u_{t-1} = u_{t-1}(1 - S_{t-1}) + u_r S_{t-1} \tag{2}$$

$$u_t = u_{t-1}(1 - 1/\tau_{\mathrm{m}}) + R_{\mathrm{m}} I_t/\tau_{\mathrm{m}} \tag{3}$$

$$S_t = f_{\mathrm{s}}(u_t, \vartheta) \tag{4}$$

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

$$u_t = \alpha u_{t-1} + (1 - \alpha) R_{\mathrm{m}} I_t - \vartheta S_{t-1} \tag{5}$$

$$\eta_t = \rho \eta_{t-1} + (1 - \rho) S_{t-1} \tag{6}$$

$$\vartheta = b_0 + \beta \eta_t \tag{7}$$

$$S_t = \hat{f}_{\mathrm{s}}(u_t, \vartheta), \tag{8}$$

where α, γ are parameters related to the temporal dynamics ($\alpha = \exp(-dt/\tau_{\mathrm{m}})$ and $\rho = \exp(-dt/\tau_{\mathrm{adp}})$), ϑ is a dynamical threshold comprising 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. The standard

validation sets were used where available to determine overfitting; for SHD we held out 5% of the training data and for (P)S-MNIST 10%. Apart from the SSC and SHD datasets, analogue 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). We used one of two methods to decode the output of the network: 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 $\hat{\mathbf{y}} = \text{softmax}\left(\sum_t S^t_{i,\text{out}}\right)$ where $S^t_{i,\text{out}}$ 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's membrane potential $u_{\text{out},t}$ at each timestep as $\hat{\mathbf{y}}_t = \text{softmax}\left(u_{\text{out},t}\right)$. 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 in which classification switches within the sample (the spiking then functions effectively as resets). We use a standard BPTT approach[12] to minimize the cross-entropy or negative-log-likelihood loss for each task using the Adam[41] 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. The BPTT for the spiking neurons is calculated retrogradely along with the self-recurrence circuits. As shown in Fig. 1d, 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 $\{u_{\text{h},t}, S_{\text{h},t}, u_{\text{o},t}, S_{\text{o},t}\}$, where $S_{\text{h},t}$ refers to a neuron firing-or-not in a hidden layer and $S_{\text{o},t}$ to an output neuron (if spiking), and $u_{\text{h},t}$ and $u_{\text{o},t}$ denote hidden and output neurons's membrane potentials. We then obtain a classification $\hat{\mathbf{y}}(t)$ either for each timestep or for the whole sequence $\hat{\mathbf{y}}$ and an associated loss. In classification tasks with C classes, the prediction probability of class $c - \hat{\mathbf{y}}_c$ is computed after having read the whole sequence, and then the loss of the network is calculated as $\mathcal{L} = \sum_{c=1}^{C} y_c \log \hat{\mathbf{y}}_c$, 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, $\mathcal{L} = \sum_{t=1}^{T} \mathcal{L}_t$. For the BPTT-derived gradient, we compute $\frac{\partial \mathcal{L}}{\partial z} = \hat{\mathbf{y}} - y$ and for recurrent weights, W_{h2o}, we compute $\frac{\partial \mathcal{L}}{\partial W_{\text{h2o}}} = \frac{\partial \mathcal{L}}{\partial z} \sum_{t'}^{T} \frac{\partial S_{\text{o},t'}}{\partial W_{\text{h2o}}}$, where each term can be computed at each timestep t' as $\frac{\partial S_{\text{o},t'}}{\partial W_{\text{h2o}}} = \frac{\partial S_{\text{o},t'}}{\partial u_{\text{o},t'}} \frac{\partial u_{\text{o},t'}}{\partial W_{\text{h2o}}} + \sum_{\xi=0}^{t'-1} \frac{\partial S_{\text{o},t'}}{\partial u_{t'}} \frac{\partial u_{\text{o},t'}}{\partial u_{\text{o},\xi}} \frac{\partial u_{\text{o},\xi}}{\partial W_{\text{h2o}}}$ and $\frac{\partial S_{\text{o},t'}}{\partial W_{\text{h2h}}} = \sum_{\xi=0}^{t'} \frac{\partial S_{\text{o},t'}}{\partial u_{\text{h},\xi}} \frac{\partial u_{\text{h},\xi}}{\partial W_{\text{h2h}}}$, and where W_{h2h} refers to weights between neurons in the hidden layers, and W_{h2o} to weights 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 $s = 6.0$ based on a grid search, and we set σ to 0.5. The standard surrogate gradients were defined following ref. [10], with the linear surrogate gradient as $\hat{f}_s'(u_t|\vartheta) = \text{ReLU}(1 - \alpha_{\text{linear}}|u_t - \vartheta|)$; the SLayer[6] gradient as $\hat{f}_s'(u_t|\vartheta) = \exp(-\alpha_{\text{slayer}}|u_t - \vartheta|)$, and the Gaussian surrogate gradient as $\hat{f}_s'(u_t|\vartheta) = \mathcal{N}(u_t|\vartheta, \sigma_{\text{G}})$; for all gradients, α is positive. We optimized all surrogate gradients hyperparameters in the experiments using grid searches; in the experiments we used $\alpha_{\text{linear}} = 1.0$, $\alpha_{\text{slayer}} = 5.0$ and $\sigma_{\text{G}} = 0.5$.

Network initialization. Compared with ANNs, SRNNs require initializing both weight and the spiking neurons's hyperparameters (that is, neuron type, time constants, thresholds, starting potential). We randomly initialize the time constants following a tight normal distribution (μ, σ) with per-layer specific parameters given in Supplementary Table 1. 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 feedforwards 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 (Extended Data Fig. 1).

For the various tasks, the loss-function, sequence length, maximum number of epochs, learning rate and decay schedule, and minibatch size are specified in Supplementary Table 1. Validation showed that the SRNNs were not prone to overfitting and test accuracy was measured at the last epoch. 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 Fig. 1b. For the ECG task, the QTDB dataset [28] consists of two channels of ECG signals. We apply a variant of the level-crossing encoding[42] threshold on the derivative of the normalized ECG signal to convert the original continuous values x into a spike train: each channel was transformed into two separate spike trains representing value increasing events and value decreasing events, respectively. The level crossing encoding we used is defined as

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

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

For the SHD, the audio records were aligned to 1 s by cutting or completing with zeros. As in a work by Cramer and colleagues[29], 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 comprises 8,156 samples, whereas the test dataset contains 2,264 samples. For the SSC dataset, the speech commands were also uniformly aligned to 1 s with a 250 Hz 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 was fed into the model frame-by-frame as input and split into training and testset as in Wang and colleagues[30]. The original range-doppler images have four channels, but we found empirically that using one channel was sufficient. For the Soli task, the first layer of the SRNN, we use a feedforwards spiking dense layer, followed by a recurrent layer. As in Wang and colleagues[30], 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 four 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 GSC v.1 (ref. [31]). For preprocessing, log Mel filters and their first and second-order derivatives are extracted from raw audio signals using Librosa[43]. For the FFTs, a window of 30 ms and a hop of 10 ms is used. The timestep of the simulation is 10 ms. We calculate the logarithm of 40 Mel filters coefficients using the Mel scale between 20 Hz and 4 kHz. Furthermore, 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 40×3 matrix.

The TIMIT database contains 3,696 and 192 samples in training and test data, respectively. We preprocessed the original audio data as in Bellec and colleagues[12], using Mel-frequency cepstral coefficient (MFCC) encoding; 10% of the training dataset was randomly selected as validation dataset, and the network was trained on the remainder. Similar to bidirectional LSTMs, we use a bidirectional adaptive SRNN for this task (see Extended Data Fig. 2a); we use two SRNN layers in the network, reading the sequence from the forwards and backwards directions, respectively. The mean of these layers's output is then fed into the last layer, an integrator, to generate the class prediction.

Data availability

The data analysed during this study are open source and publicly available. The dataset for ECG streaming dataset is derived from original QTDB dataset (https://physionet.org/content/qtdb/1.0.0/). Spiking datasets (SHD and SSC) belong to Spiking Heidelberg Datasets, which are available at https://zenkelab.org/resources/spiking-heidelberg-datasets-shd/. The MNIST dataset can be downloaded from http://yann.lecun.com/exdb/mnist/. The Soli dataset can be downloaded at https://polybox.ethz.ch/index.php/s/wG93iTUdvRU8EaT. TIMIT Acoustic-Phonetic Continuous Speech Corpus are available on request via https://doi.org/10.35111/17gk-bn40. Further information can be found in our repository (see the Code Availability section). Source data are provided with this paper.

Code availability

The code used in the study is publicly available from the GitHub repository (https://github.com/byin-cwi/Efficient-spiking-networks).

Received: 23 March 2021; Accepted: 1 September 2021;
Published online: 14 October 2021

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 Netw.* **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 (ESANN, 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 (NeurIPS, 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 Netw.* **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 (IEEE, 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. Event-based backpropagation can compute exact gradients for spiking neural networks. *Sci. Rep.* **11**, 12829 (2021).
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. Yin, B., Corradi, F. & Bohté, S. M. Effective and efficient computation with multiple-timescale spiking recurrent neural networks. In *International Conference on Neuromorphic Systems 2020* 1–8 (ACM, 2020).
16. Werbos, P. J. Backpropagation through time: what it does and how to do it. *Proc. IEEE* **78**, 1550–1560 (1990).
17. Elfwing, S., Uchibe, E. & Doya, K. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Netw.* **107**, 3–11 (2018).
18. Elfwing, S., Uchibe, E. & Doya, K. Expected energy-based restricted boltzmann machine for classification. *Neural Netw.* **64**, 29–38 (2015).
19. Gerstner, W. & Kistler, W. M. *Spiking Neuron Models: Single Neurons, Populations, Plasticity* (Cambridge Univ. Press, 2002).
20. Izhikevich, E. M. Simple model of spiking neurons. *IEEE Trans. Neural Netw.* **14**, 1569–1572 (2003).
21. 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 (NeurIPS, 2018).
22. Bohte, S. M. Error-backpropagation in networks of fractionally predictive spiking neurons. In *International Conference on Artificial Neural Networks (ICANN)* 60–68 (Springer, 2011).
23. Wong, A., Famouri, M., Pavlova, M. & Surana, S. Tinyspeech: attention condensers for deep speech recognition neural networks on edge devices. Preprint at https://arxiv.org/abs/2008.04245 (2020).
24. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Comput.* **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. 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).
28. 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).
29. 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 and Learning Systems* 1–14 (IEEE, 2020); https://doi.org/10.1109/TNNLS.2020.3044364
30. 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 *Proc. 29th Annual Symposium on User Interface Software and Technology* 851–860 (ACM, 2016).
31. Warden, P. Speech commands: a dataset for limited-vocabulary speech recognition. Preprint at https://arxiv.org/abs/1804.03209 (2018).
32. Garofolo, J. S. *TIMIT Acoustic Phonetic Continuous Speech Corpus* (Linguistic Data Consortium, 1993).
33. Pellegrini, T., Zimmer, R. & Masquelier, T. Low-activity supervised convolutional spiking neural networks applied to speech commands recognition. In *2021 IEEE Spoken Language Technology Workshop (SLT)* 97–103 (IEEE, 2021).
34. 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 *Proc. IEEE/CVF Winter Conference on Applications of Computer Vision* 3953–3962 (IEEE, 2021).
35. Fang, W. et al. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. Preprint at https://arxiv.org/abs/2007.05785 (2020).
36. Amir, A. et al. A low power, fully event-based gesture recognition system. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition* 7243–7252 (IEEE, 2017).
37. Paszke, A. et al. Pytorch: an imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems* Vol. 32, 8024–8035 (NeurIPS, 2019).
38. Zenke, F. et al. Visualizing a joint future of neuroscience and neuromorphic engineering. *Neuron* **109**, 571–575 (2021).
39. Zenke, F. & Neftci, E. O. Brain-inspired learning on neuromorphic substrates. *Proc. IEEE* Vol. 109, 1–16 (IEEE, 2021).
40. Keijser, J. & Sprekeler, H. Interneuron diversity is required for compartment-specific feedback inhibition. Preprint at https://doi.org/10.1101/2020.11.17.386920 (2020).
41. Kingma, D. P. & Ba, J. Adam: a method for stochastic optimization. In *3rd International Conference on Learning Representations* (DBLP, 2015).
42. Lichtsteiner, P., Posch, C. & Delbruck, T. A 128 × 128 120 db 15 μs latency asynchronous temporal contrast vision sensor. *IEEE J. Solid-State Circuits* **43**, 566–576 (2008).
43. McFee, B. et al. librosa: audio and music signal analysis in Python. In *Proc.14th Python in Science Conference* Vol. 8, 18–25 (SciPy, 2015).
44. Li, S., Li, W., Cook, C., Zhu, C. & Gao, Y. Independently recurrent neural network (indrnn): building a longer and deeper RNN. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 5457–5466 (IEEE, 2018).
45. Arjovsky, M., Shah, A. & Bengio, Y. Unitary evolution recurrent neural networks. In *International Conference on Machine Learning* 1120–1128 (ACM, 2016).
46. 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 (2021).
47. Perez-Nieves, N., Leung, V. C., Dragotti, P. L. & Goodman, D. F. Neural heterogeneity promotes robust learning. Preprint at https://www.biorxiv.org/content/10.1101/2020.12.18.423468v2.full (2021).
48. de Andrade, D. C., Leo, S., Viana, M. L. D. S. & Bernkopf, C. A neural attention model for speech command recognition. Preprint at https://arxiv.org/abs/1808.08929 (2018).
49. Graves, A. & Schmidhuber, J. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Netw.* **18**, 602–610 (2005).
50. Hunger, R. *Floating Point Operations in Matrix-Vector Calculus* (Munich Univ. Technology, 2005).

Acknowledgements

B.Y. is funded by the NWO-TTW Programme 'Efficient Deep Learning' (EDL) P16-25. We gratefully acknowledge the support from the organizers of the Capo Caccia Neuromorphic Cognition 2019 workshop and Neurotech CSA, as well as J. Wu and S. S. Magraner for helpful discussions.

Author contributions

B.Y., F.C. and S.B. conceived the experiments, B.Y. conducted the experiments, B.Y., F.C. and S.B. analysed the results. All authors reviewed the manuscript.

Competing interests

The authors declare no competing interests.

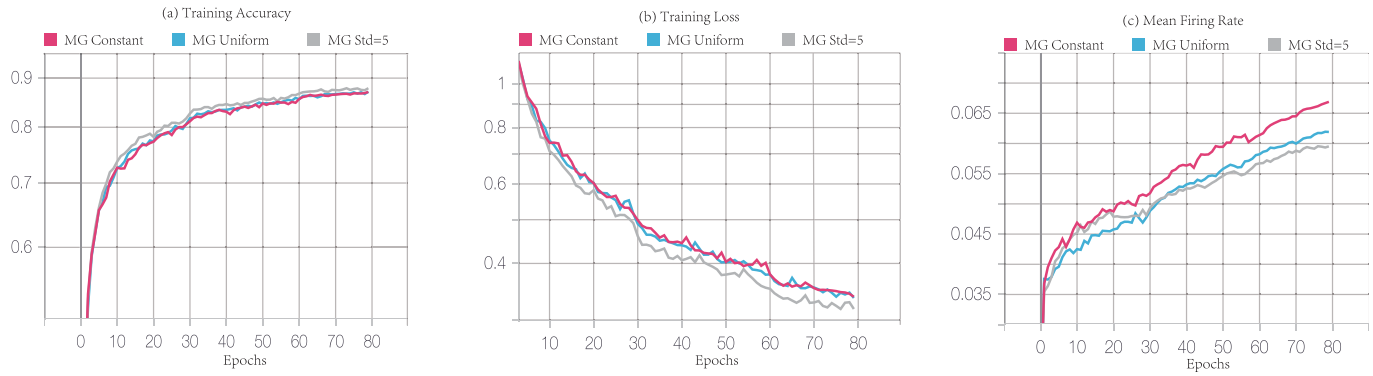
Additional information

Extended data is available for this paper at https://doi.org/10.1038/s42256-021-00397-w.

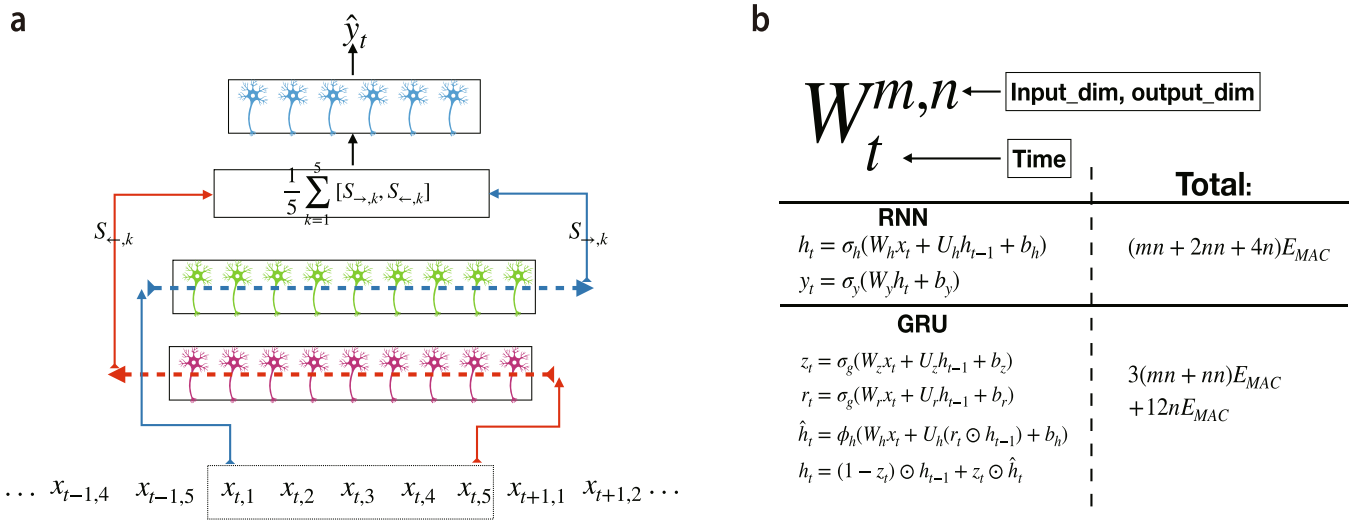
Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s42256-021-00397-w.

Correspondence and requests for materials should be addressed to Bojian Yin.

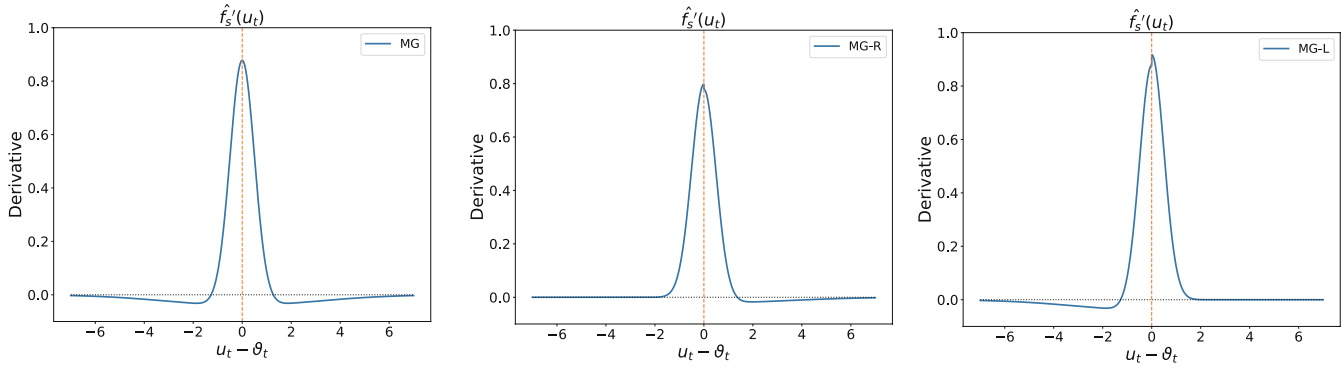
Peer review information *Nature Machine Intelligence* thanks Thomas Nowotny and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.



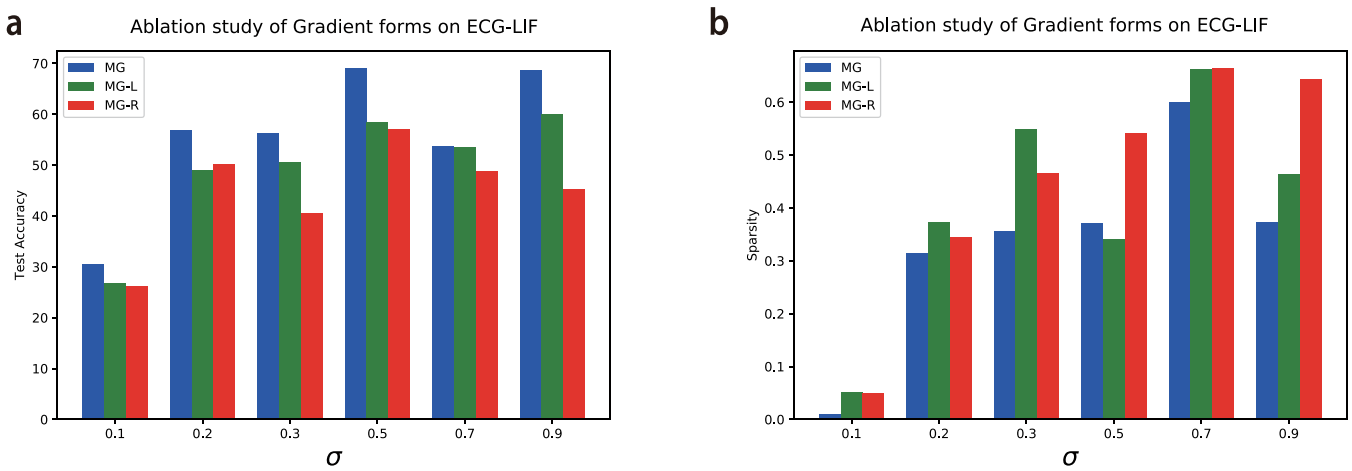
Extended Data Fig. 1 | Effects of different time constant initialization schemes on network training and performance on the SoLi dataset. a, Training accuracy **b**, Training Loss **c**, Mean Firing rate of the network. The $MG_{constant}$ is the network where τ is initialized with a single value; for $MG_{uniform}$ the network is initialized with uniformly distributed time-constants near the single value of $MG_{constant}$; for MG_{std5}, a normal distribution with std 5.0 is used near the same single value.



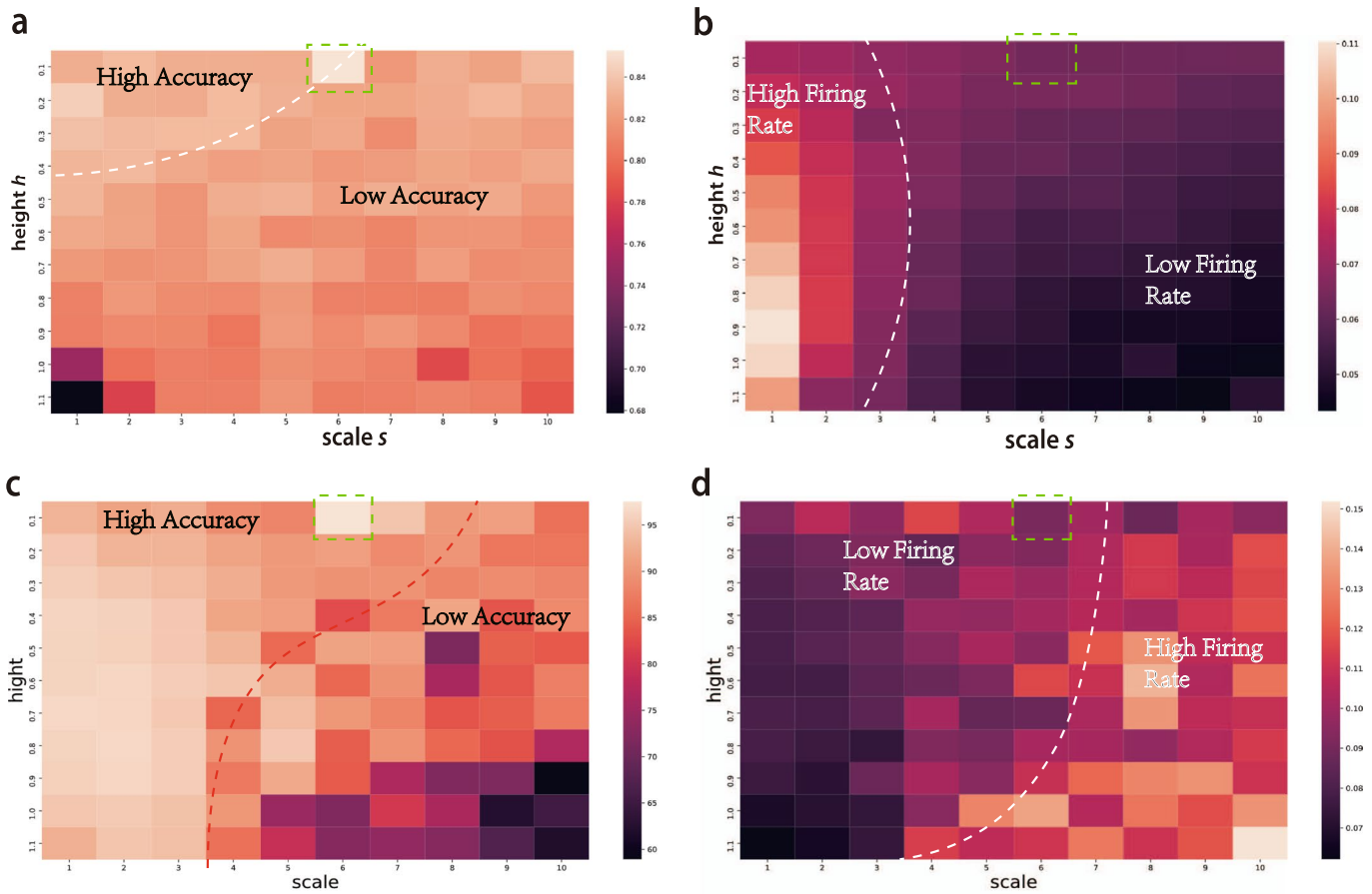
Extended Data Fig. 2 | SI-panel. a, Bi-directional SRNN architecture. **b**, Computational cost computation of different layers for regular RNNs and GRU units. The computational complexity calculation follows[50].



Extended Data Fig. 3 | Variants of Multi-Gaussian gradient. As illustrated, we remove either the left(MG-R) or right(MG-L) negative part of the Multi-Gaussian gradient for comparison, leaving on the ablated part the positive Gaussian gradient.



Extended Data Fig. 4 | Study of different forms of gradients on ECG-LIF. (a,b) shows the result of the using various Multi-Gaussian negative gradient ablations on the ECG-LIF task where the σ of the central (positive) Gaussian as defined in Eq (1) is varied. The effect of varying σ is shown for test accuracy (a) and sparsity (b). We find that also then, the standard Multi-Gaussian outperforms variations in terms of accuracy and sparsity.



Extended Data Fig. 5 | A grid search was performed on the SoLi dataset and SHD for the *h* and *s* parameters of the multi-Gaussian surrogate gradient. In the grid search, we calculated the performance of each pair of parameters by averaging the test accuracy and firing rate over tri-folder cross-validation. The white dashed line delineates the upper left region for models with high accuracy (> 0.91) in **(a)** and high firing rate (> 0.09) in **(b)**. The red lines in **(c)** approximately delineate regions with accuracy above and below 0.87, and the white curve in **(d)** approximately demarcates models with an average firing rate above or below 0.1.