



OPEN

## Supervised training of spiking neural networks for robust deployment on mixed-signal neuromorphic processors

Julian Büchel<sup>1,2</sup>, Dmitrii Zendrikov<sup>2</sup>, Sergio Solinas<sup>3</sup>, Giacomo Indiveri<sup>1,2</sup> & Dylan R. Muir<sup>1</sup>✉

Mixed-signal analog/digital circuits emulate spiking neurons and synapses with extremely high energy efficiency, an approach known as “neuromorphic engineering”. However, analog circuits are sensitive to process-induced variation among transistors in a chip (“device mismatch”). For neuromorphic implementation of Spiking Neural Networks (SNNs), mismatch causes parameter variation between identically-configured neurons and synapses. Each chip exhibits a different distribution of neural parameters, causing deployed networks to respond differently between chips. Current solutions to mitigate mismatch based on per-chip calibration or on-chip learning entail increased design complexity, area and cost, making deployment of neuromorphic devices expensive and difficult. Here we present a supervised learning approach that produces SNNs with high robustness to mismatch and other common sources of noise. Our method trains SNNs to perform temporal classification tasks by mimicking a pre-trained dynamical system, using a local learning rule from non-linear control theory. We demonstrate our method on two tasks requiring temporal memory, and measure the robustness of our approach to several forms of noise and mismatch. We show that our approach is more robust than common alternatives for training SNNs. Our method provides robust deployment of pre-trained networks on mixed-signal neuromorphic hardware, without requiring per-device training or calibration.

Dedicated hardware implementations of Spiking Neural Networks (SNNs) are an extremely energy-efficient computational substrate on which to perform signal processing and machine learning inference tasks<sup>1–8</sup>. Optimal energy efficiency is achieved when using mixed-signal analog/digital neuron and synapse circuits following an approach known as “neuromorphic engineering”<sup>9</sup>. In these hardware devices, large arrays of neurons and synapses are physically instantiated in silicon, and coupled with flexible digital routing and interfacing logic in “mixed-signal” designs<sup>2,6</sup>.

However, all analog silicon circuits suffer from process variation across the surface of a chip, changing the operating characteristics of otherwise identical transistors—known as “device mismatch”<sup>10,11</sup>. In the case of spiking neurons implemented using analog or mixed-signal circuits, mismatch is expressed as parameter variation between neurons and synapses that are otherwise configured identically<sup>12–15</sup>. The parameter mismatch on each device appears as frozen parameter noise, introducing variance between neurons and synapses in time constants, thresholds, and weight strength.

Parameter noise in mixed-signal neuromorphic devices can be exploited as a symmetry-breaking mechanism, especially for neural network architectures that rely on randomness and stochasticity as a computational mechanism<sup>16–20</sup>, or can be exploited to improve in-situ training of Bayesian networks via MCMC sampling<sup>21</sup>. However, random architectures can raise problems for commercial deployment of applications on mixed-signal devices: the parameter noise would affect neuronal response dynamics, and these device to device variations could affect and degrade the system performance of individual chips. A possible solution is to perform post-production device calibration or re-training, but this would raise deployment costs significantly and not scale well with deployment to large numbers of devices. In addition to device mismatch, mixed-signal neuromorphic systems also suffer from other sources of noise, such as thermal noise or quantisation noise introduced by restricting synaptic weights to a low bit-depth.

<sup>1</sup>SynSense, Thurgauerstrasse 40, 8050 Zurich, Switzerland. <sup>2</sup>Institute of Neuroinformatics, University of Zurich and ETH Zurich, Winterthurerstrasse 190, 8057 Zurich, Switzerland. <sup>3</sup>Department of Biomedical Science, University of Sassari, Piazza Università, 21, 07100 Sassari, Sardegna, Italy. ✉email: dylan.muir@synsense.ai

In contrast to current mainstream Deep Neural Networks (DNNs), spiking networks suffer from a severe configurability problem. The backpropagation algorithm permits configuration of extremely deep NNs for arbitrary tasks<sup>22</sup>, and is effective also for network models with temporal state<sup>23</sup>, but is difficult to apply to the discontinuous dynamics of SNNs<sup>24–26</sup>. Methods to approximate the gradient calculations by using surrogate functions<sup>27</sup>, eligibility traces<sup>28</sup> or adjoint networks<sup>29</sup> have provided a way to adapt backpropagation for spiking networks. Non-local information is required for strict implementation of the backpropagation algorithm, but random feedback<sup>30</sup> and local losses<sup>31</sup> have been employed with some success to train multi-layer spiking networks. Alternative approaches using initial random dynamics coupled with error feedback and spike-based learning rules can permit recurrent SNNs to mimic a teacher dynamical system<sup>32,33</sup>. Strictly-local spike-timing-based learning rules, inspired by results in experimental neuroscience<sup>34</sup>, have been implemented in digital and mixed-signal neuromorphic devices, as they provide a better match to the distribution of information across neuromorphic chips<sup>35</sup>. Unfortunately, local spike-dependent rules such as Spike-Timing Dependent Plasticity (STDP) are themselves not able to perform supervised training of arbitrary tasks, since they do not permit error feedback or error-based modification of parameters. In both cases, implementing strictly local or backpropagation-based learning infrastructure on-chip adds considerable complexity, size and therefore cost to neuromorphic hardware designs. This cost makes it impractical to use on-chip learning and adaptation to solve the mismatch problem on mixed-signal architectures.

Robustness to noise and variability can be approached from the architectural side. For example, a network architecture search approach can identify networks that are essentially agnostic to precise weight values<sup>36</sup>. However, these networks rely on complex combinations of transfer functions which do not map to neuromorphic SNN designs.

Alternatively, a class of analytically-derived network architectures have been proposed for spiking networks, known as Efficient Balanced Networks<sup>37–43</sup>, relying on a balance between excitation and inhibition to provide robustness to sources of noise including spike-time stochasticity and neuron deletion. These networks can be derived to mimic an arbitrary linear dynamical system through an auto-encoding architecture<sup>38</sup> or can learn to represent and mimic dynamical systems<sup>37,40–42</sup>. We propose to adapt the learning machinery of this spiking architecture to produce deployable SNN-based solutions for arbitrary supervised tasks that are robust to noise and device mismatch.

In this work we present a method for training robust networks of Leaky Integrate and Fire (LIF) spiking neurons that can solve supervised temporal signal regression and classification tasks. We adopt a knowledge distillation approach, by first training a non-spiking Recurrent Neural Network (RNN) to solve the desired supervised task using Back-Propagation Through Time (BPTT)<sup>23</sup>. By then interpreting the activations of the RNN as a teacher dynamical system, we train an SNN using an adaptation of the learning rule from Ref.<sup>41</sup> to mimic the RNN. We show that the resulting trained SNN is robust to multiple forms of noise, including simulated device mismatch, making our approach feasible for deployment on to mixed-signal devices without post-deployment calibration or learning. We compare our method with several other standard approaches for configuring SNNs, and show that ours is more robust to device mismatch.

## Results

We assume a family of tasks defined by mappings  $\mathbf{c}(t) \rightarrow \hat{\mathbf{y}}(t)$ , where  $\mathbf{c}(t) \in \mathbb{R}^{d1}$  and  $\hat{\mathbf{y}}(t) \in \mathbb{R}^{d2}$  are temporal signals with arbitrary dimensionality (Fig. 1a; see “Methods”). For simplicity of notation we do not write the temporal dependency “(t)” for the remainder of the paper. This definition encompasses any form of deterministic temporal signal processing or classification task without loss of generality. We refer to our network architecture as ADS (Arbitrary Dynamical System) spiking networks.

Our approach begins by training a non-spiking rate network to implement the arbitrary task mapping by learning the dynamical system

$$\begin{aligned}\tau \dot{\hat{\mathbf{x}}} &= \hat{\Omega} f(\hat{\mathbf{x}}) + \hat{\mathbf{F}} \mathbf{c} + b \\ \hat{\mathbf{y}} &= \hat{\mathbf{D}} \hat{\mathbf{x}}\end{aligned}$$

through modification of the recurrent weights  $\hat{\Omega} \in \mathbb{R}^{\hat{N} \times \hat{N}}$ ; encoding and decoding weights  $\hat{\mathbf{F}} \in \mathbb{R}^{d1 \times \hat{N}}$  and  $\hat{\mathbf{D}} \in \mathbb{R}^{\hat{N} \times d2}$ ; biases  $b \in \mathbb{R}^{\hat{N}}$ ; time constants  $\tau \in \mathbb{R}^{\hat{N}}$ ; and non-linear transfer function  $f(\cdot) = \tanh(\cdot)$ . BPTT or any other suitable approach can be used to obtain the trained rate network.

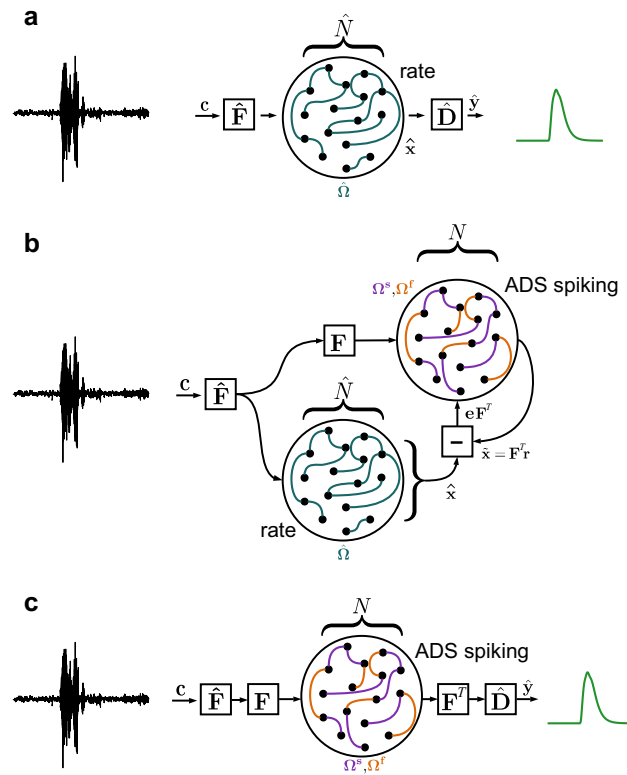
We subsequently train a network of spiking neurons to emulate  $\hat{\mathbf{x}}$ , with leaky membrane dynamics defined by

$$\dot{V} = -\lambda V + \hat{\mathbf{F}} \mathbf{F} \mathbf{c} - \Omega^f \mathbf{o} + \Omega^s \mathbf{o} + k \mathbf{F}^T \mathbf{e}$$

with spike trains  $\mathbf{o} = V > V_{\text{thresh}}$  produced when exceeding threshold voltages  $V_{\text{thresh}}$ ; leak rate  $\lambda$ ; and fast and slow recurrent weights  $\Omega^f$  and  $\Omega^s$  (Fig. 1b; see “Methods”). The decoded dynamics  $\tilde{\mathbf{x}} \approx \hat{\mathbf{x}}$  are obtained from the filtered spiking activity  $\mathbf{r}$  with  $\tilde{\mathbf{x}} = \mathbf{F} \mathbf{r}$ . By feeding back an error signal  $\mathbf{e} = \tilde{\mathbf{x}} - \hat{\mathbf{x}}$  under the control of a decaying feedback rate  $k$ , the spiking network is forced to remain close to the desired target dynamics.  $\Omega^f$  is initialised to provide fast balanced feedback<sup>39</sup>, and  $\Omega^s$  is learned using the rule

$$\dot{\Omega}^s = \eta \mathbf{r} \mathbf{F} \mathbf{e}$$

under learning rate  $\eta$  (see “Methods” and Ref.<sup>41</sup>). Note that we do not require complex multi-compartmental neurons or dendritic nonlinearities in our neuron model, but use a simple leaky integrate-and-fire neuron that is compatible with compact mixed-signal neuromorphic implementation<sup>2</sup>. Once the spiking network has learned to represent  $\tilde{\mathbf{x}} \approx \hat{\mathbf{x}}$  with high accuracy, we replace the rate network entirely with the spiking network (Fig. 1c).

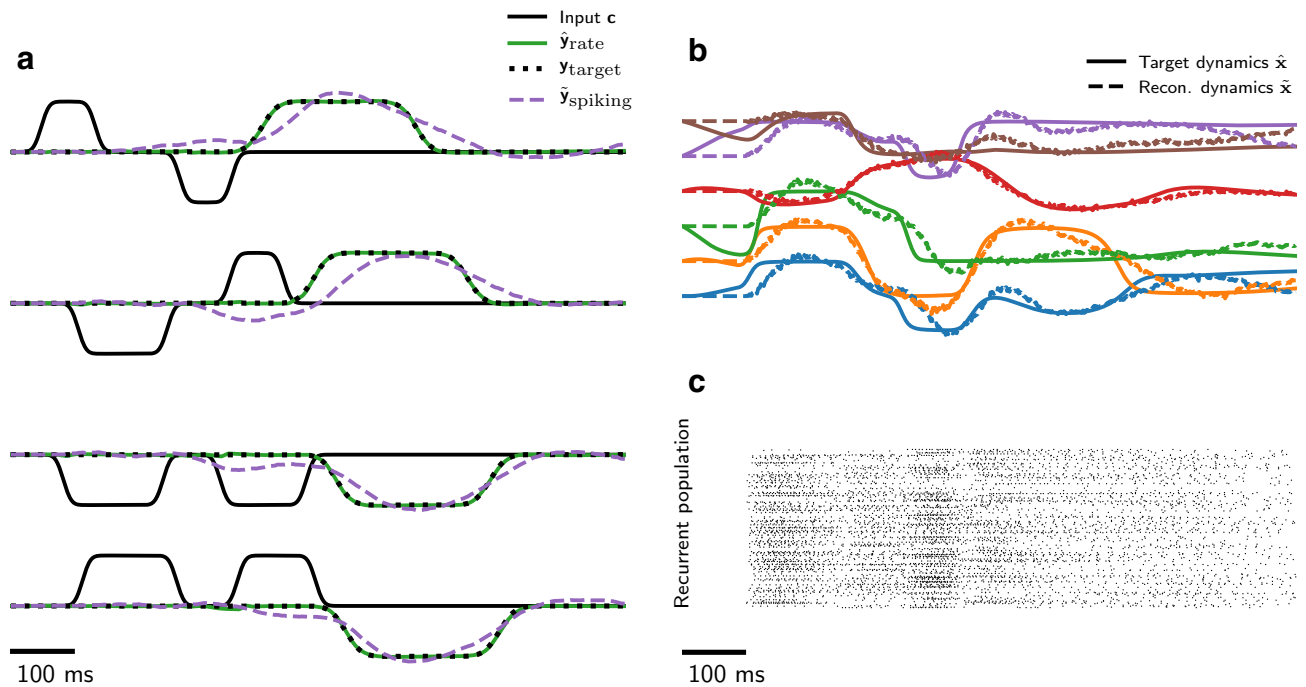


**Figure 1.** Schematic overview of our supervised training approach. **(a)** A recurrent non-spiking neural network with  $\hat{N}$  neurons (“rate”) is trained using BPTT or a similar approach to implement the mapping  $\mathbf{c} \rightarrow \hat{\mathbf{y}}$ , via encoding and decoding weights  $\hat{\mathbf{F}}$  and  $\hat{\mathbf{D}}$ , using the recurrent weights  $\hat{\Omega}$  and resulting in the internal temporal representation of neural activity  $\hat{\mathbf{x}}$ . **(b)** To train a robust spiking network for the task, a network with  $N \neq \hat{N}$  LIF neurons (“ADS spiking”) is initialised with fast balanced feedback connections  $\Omega^f$ , analytically determined from a randomly chosen encoding matrix  $\mathbf{F}$ . The ADS spiking network learns to represent the target signals  $\hat{\mathbf{x}}$  with reference to an error signal  $\mathbf{e} = \tilde{\mathbf{x}} - \hat{\mathbf{x}}$ , by adapting slow feedback connections  $\Omega^s$ . **(c)** For inference, the ADS spiking network replaces the non-spiking rate network, and uses the encoding and decoding weights  $\hat{\mathbf{F}}$  and  $\hat{\mathbf{D}}$  to implement the trained task mapping  $\mathbf{c} \rightarrow \hat{\mathbf{y}}$ .

**Temporal XOR task.** We begin by demonstrating our method using a nonlinear temporal XOR task (Fig. 2; see “Methods”). This task requires memory of past inputs to produce a delayed output, as well as a nonlinear mapping between the memory state and the output variable. A network receives a single input channel where pulses of varying width (100–230 ms) and sign are presented in sequence. The network must report the XOR of the two input pulses by delivering an output pulse of appropriate sign after the second of the two input pulses. A non-spiking RNN ( $N = 64$ ) was trained to perform the temporal XOR task, using BPTT with Mean-Squared Error (MSE) loss against the target output signal (target and output signals shown in Fig. 2a). After 20 epochs of training with 500 samples per epoch, the RNN reached negligible error on 200 test samples ( $\approx 100\%$  accuracy). A spiking ADS network ( $N = 320$ ) was then trained to perform the task, reaching equivalent accuracy (Fig. 2a,b).

**Wake-phrase detection.** The temporal XOR task demonstrates that one-dimensional nonlinear tasks requiring memory can be learned through our method through supervised training. To show that our approach also works on more realistic tasks with complex input dynamics, we implemented an audio wake-phrase detection task (Fig. 3; see “Methods”). Briefly, real-time audio signals were extracted from a database of spoken wake phrases (“Hey Snips” dataset<sup>44</sup>), or from a database of noise samples (“DEMAND” dataset<sup>45</sup>). The target wake phrase data was augmented with noise at an SNR of 10 dB, then passed through a bank of 16 Butterworth filters with central frequencies spaced between 0.4 and 2.8 kHz (Fig. 3b). We trained a non-spiking RNN ( $\hat{N} = 128$ ) to perform the task with high accuracy, using BPTT under an MSE loss function against a smooth target classification signal (Fig. 3d,e). We then trained a spiking ADS network ( $N = 768$ ) to implement the audio classification task. The non-spiking RNN achieved a testing accuracy of  $\approx 90\%$ , and our spiking imitator achieved  $\approx 87\%$  after training for 10 epochs on 1000 training samples.

**Training considerations.** We found that slower input, internal and target dynamics in the RNN were easier for the SNN to reconstruct than very rapid dynamics, depending on the neuron and synaptic time constants in the SNN. Longer and slower target responses yielded smoother ANN dynamics, which were easier for the spik-



**Figure 2.** Our approach implements a supervised temporal non-linear sequence classification task to high accuracy. **(a)** The non-spiking RNN (green;  $\tilde{N} = 64$  neurons) is trained to perform a temporal XOR of the input (black), closely matching the target function (dotted). A spiking ADS network (dashed;  $N = 320$  neurons) is trained to perform the same task. **(b)** The first six internal dynamical variables  $\hat{\mathbf{x}}$  of the RNN are shown (solid), along with their reconstructed equivalents  $\tilde{\mathbf{x}}$  from the spiking ADS network (dashed). **(c)** The spiking activity of the ADS network. Panels **(b)** and **(c)** correspond to the first example in **(a)**.

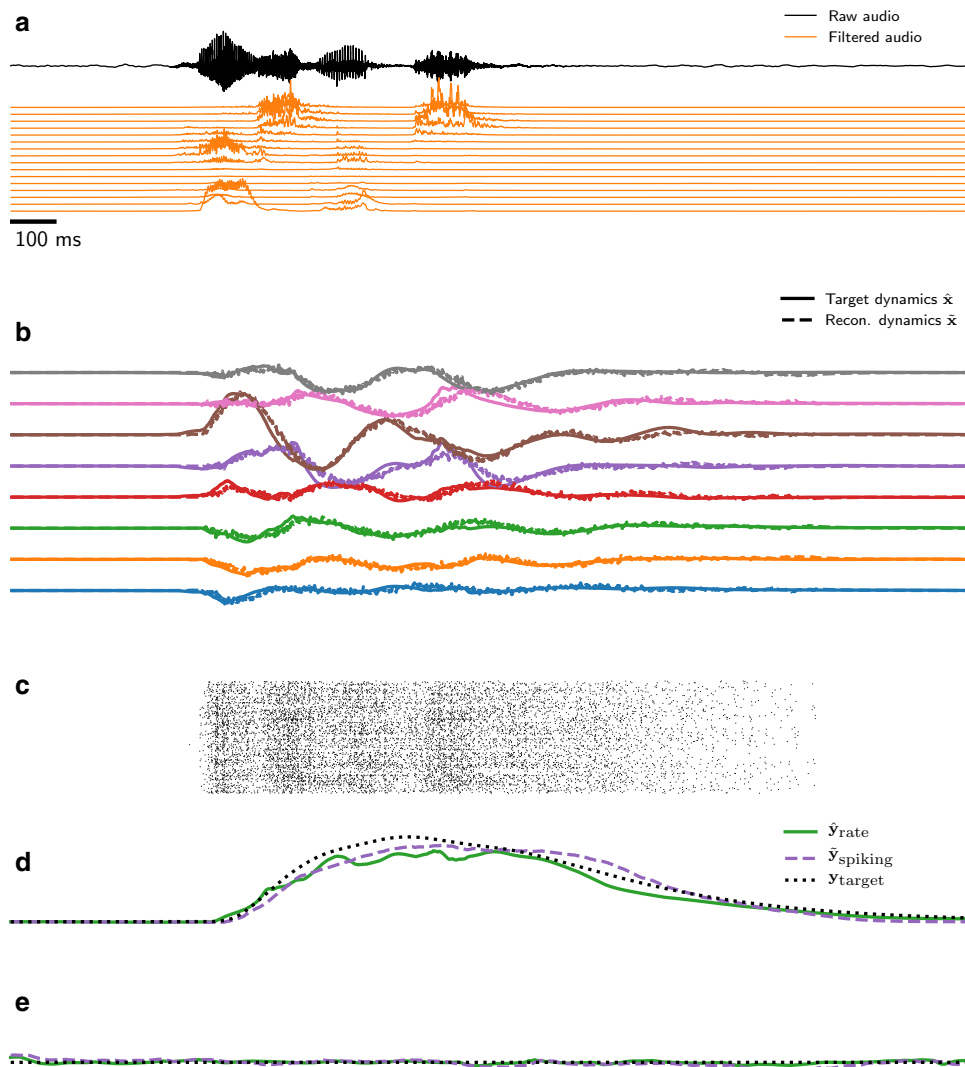
ing ADS network to learn. Our approach did not assume any dendritic non-linearities, or multi-compartmental dendrites with complex basis functions. Instead, the non-linearity of the spiking neuron dynamics is sufficient to learn the dynamics of a non-spiking ANN using the tanh nonlinearity.

We found that including a learning schedule for the error feedback rate  $k$  was important to achieve low reconstruction error. The factor  $k$  must drop to close to zero before the end of training, or else the SNN learns to rely on error feedback for accuracy, and generalisation will be poor once error feedback is removed. Conversely, if  $k$  drops too rapidly during training, the SNN is not held close to the desired target dynamics, and is unable to correctly learn the slow feedback weights  $\Omega^s$ . For these reasons, a well-chosen schedule for  $k$  is important during learning. In this work we chose a progressive stepping function that decrements  $k$  by a fixed amount after some number of signal iterations (see “Methods”). Setting  $k$  to a fixed value for some number of trials enables the SNN to adapt to the corresponding scale of error feedback by updating  $\Omega^s$ .

**Robustness to noise sources.** The slow learned recurrent feedback connections  $\Omega^s$  in the spiking network enable the SNN to reproduce a learned task. In contrast, the balanced fast recurrent feedback connections  $\Omega^f$  are designed to enable the SNN to encode the dynamic variables  $\tilde{\mathbf{x}}$  in a way that is robust to perturbation<sup>38,39</sup>. We examined the robustness of our trained networks to several sources of noise (Fig. 4).

**Device mismatch.** We first introduced frozen parameter noise as a simulation of device mismatch present in mixed-signal neuromorphic implementation of event-driven neuron and synapses. We measured distributions of neuronal and synaptic parameters induced in silicon spiking neurons by device mismatch (see “Methods”; Fig. S1). Measurements were performed on 1 core of 256 analog neurons and synapses, on fabricated mixed-signal neuromorphic DYNAP-SE processors<sup>2</sup>. We observed a consistent relationship between the mean and variance of parameter distributions: the variance of the measured parameters increased linearly with the magnitude of the set parameter. We used this experimentally-recorded relationship to simulate mismatch in our spiking network implementations, simulating deployment of the networks on mixed-signal neuromorphic hardware. Mismatched parameters  $\Theta'$  were generated with  $\Theta' \sim \mathcal{N}(\Theta, \delta\Theta)$ , where  $\delta$  determines the level of mismatch, which we found experimentally to be between 10–20%. Under 20% simulated mismatch on weights, thresholds, biases, synaptic and neuronal time constants, our networks compensated well for the frozen parameter noise present in mixed-signal deployment (Fig. 4b).

**Quantisation noise.** In contrast to 64-bit floating point precision used by the non-spiking RNN, deployment of NN architectures in memory-constrained systems often uses low bit-depth precision for weights and neuron state. Mixed-signal neuromorphic architectures use analog voltages or currents to represent internal neural state,



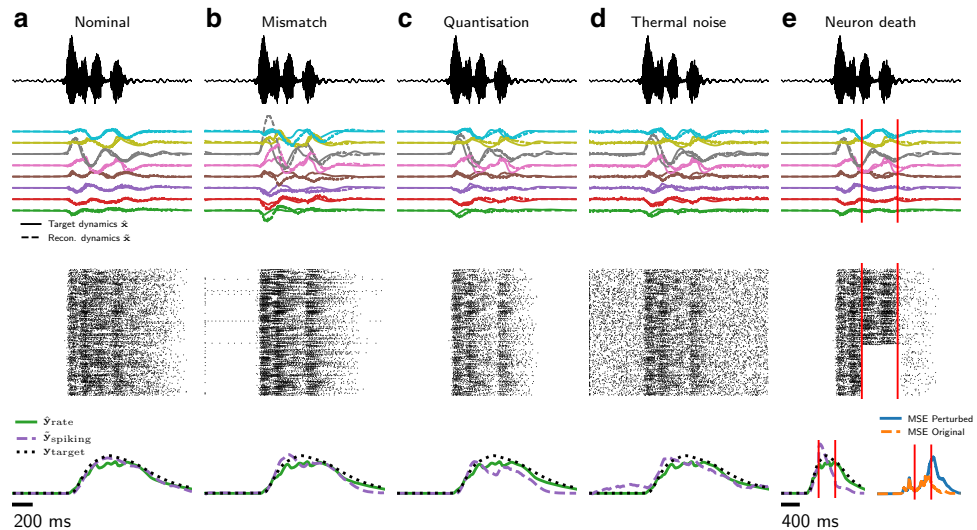
**Figure 3.** Our approach performs a supervised spoken audio multi-dimensional classification task with high accuracy. (a) Audio samples were presented that either matched a spoken target phrase, or consisted of random speech or background noise. Raw audio (black) was filtered into 16 channels (orange) for classification by the network ( $\mathbf{c} \in \mathbb{R}^{16}$ ). (b) Internal RNN dynamics ( $\tilde{\mathbf{x}}$ ; solid) was reconstructed accurately by the trained spiking ADS network ( $\tilde{\mathbf{x}}$ ; dashed) based on the spiking activity (shown in c). An output signal was high (shown in d) when the audio sample matched the target signal, or low (shown in e) when the audio signal was background noise or other speech. Panels (a)–(d) correspond to a single trial.

but can use some form of quantisation for synaptic weights. For example, DYNAP-SE2 processors impose a five-bit representation of synaptic weights, as well as a restricted fan-in of 64 pre-synaptic input sources per neuron<sup>2</sup>. We imposed weight quantisation constraints on our spiking model, and found that our networks compensated well for the resulting frozen quantisation noise (Fig. 4c; see “Methods”).

**Thermal noise.** Due to the analog representation of neuron and synapse states in mixed-signal neuromorphic chips, these state variables are subject to thermal noise. Thermal noise appears as white-noise stochastic fluctuations of all states. We simulated thermal noise by adding noise  $\zeta \sim \mathcal{N}(0, \sigma)$  to membrane potentials  $V$ , with  $\sigma = 1\%$ ,  $5\%$ ,  $10\%$  scaled to the range between reset and threshold potentials  $V_{\text{reset}}$  and  $V_{\text{thresh}}$ . The spiking ADS network performed well in the presence of thermal noise (Fig. 4d).

**Sudden neuron failure.** The fast recurrent feedback connections  $\Omega^f$  present in spiking balanced networks have been shown to be able to compensate for neuron loss, where a subpopulation of spiking neurons is silenced during a trial<sup>38,41,46</sup>. We examined this property in our spiking ADS networks that include fast balanced feedback, and found that indeed our networks compensated well for neuron loss (Fig. 4e). In the absence of fast recurrent feedback (i.e.  $\Omega^f = 0$ ), neuron silencing degraded the performance of the spiking ADS networks (Fig. S3).





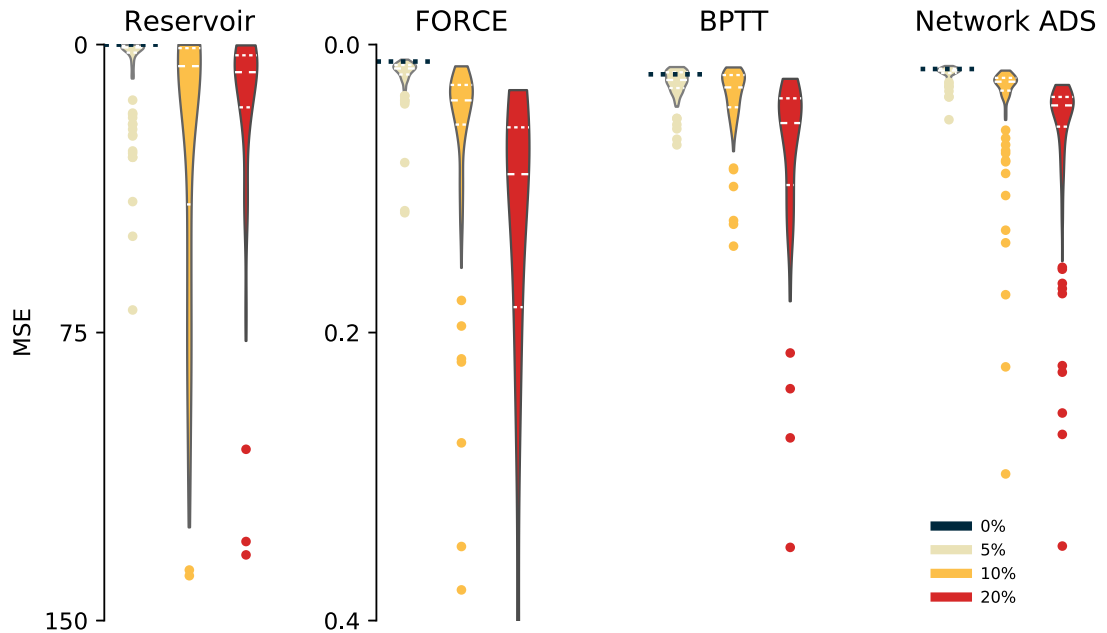
**Figure 4.** Our trained spiking networks are robust to device mismatch and other sources of noise. Each column shows (top to bottom) the raw signal input; the ANN and reconstructed dynamics; the spiking activity of the ADS network; and the task output and target signals. **(a)** The trained ADS network reproduces the ANN internal and target signals with high accuracy. **(b)** In the presence of simulated mismatch in a mixed-signal silicon implementation of LIF neurons (20%; see “Methods”), the ADS network compensates well for the resulting frozen parameter noise. **(c)** Frozen weight noise introduced by quantisation of weights to 4 bits is compensated by the ADS spiking network. **(d)** The spiking network also compensates well in the presence of simulated thermal noise ( $\sigma = 5\%$ ). **(e)** The balanced fast recurrent feedback connections  $\Omega^f$  permit the ADS spiking network to compensate for sudden neuron death (40% of spiking neurons silenced between vertical bars).

**Comparison with alternative architectures.** We have demonstrated that our method produces spiking implementations of arbitrary tasks, defined through supervised training. We compared our approach against several alternative methods for supervised training of SNNs, and evaluated the performance of these methods under simulated deployment on mixed-signal neuromorphic hardware:

- Reservoir Computing, in the form of a Liquid State Machine<sup>18</sup>, relies on the random dynamics of an SNN to project an input over a high-dimensional temporal basis. A readout is then trained to map the random temporal basis to a specified target signal, using regularised linear regression. Since perturbation of the weights and neural parameters will directly modify the temporal basis, we expect the Reservoir approach to perform poorly in the presence of mismatch.
- The spiking FORCE algorithm<sup>32</sup> trains an SNN to mimic a teacher dynamical system. We applied this algorithm to a trained non-spiking RNNs to produce a trained SNN, similarly as in our spiking ADS approach.
- We implemented the BPTT algorithm to train an SNN end-to-end, using a surrogate gradient function similar to Ref.<sup>25</sup>. During training, these networks received input and target functions identical to those presented to the non-spiking RNN.

We first examined simulated deployment of all architectures by simulating parameter mismatch (Fig. 5; see “Methods”). We trained 10 networks for each architecture, and evaluated each network at three levels of mismatch ( $\delta = 5\%, 10\%, 20\%$ ) for 10 random mismatch trials of 500 samples each. We quantified the effect of mismatch on the performance of each network architecture by measuring the MSE between the SNN-generated output  $\tilde{y}$  and the training target for that architecture. For the FORCE and ADS networks the training target was the output of the non-spiking RNN  $y$ . In the case of the Reservoir and BPTT architectures, the training target was the target task output  $\hat{y}$ . Under the lowest level of simulated mismatch (5%), the spiking ADS network showed the smallest degradation of network response (MSE drop  $0.0094 \rightarrow 0.0109$ ;  $p \approx 8 \times 10^{-14}$ ,  $U$  test). The spiking ADS network also showed the smallest mismatched variance in MSE, reflecting that all mismatched networks responses were close to the desired target response (MSE std. dev. ADS 0.0076; Reservoir 11.4; FORCE 0.0244; BPTT 0.0105;  $p < 1 \times 10^{-2}$  in all cases, Levene test). The spiking Reservoir architecture fared the worst, with large degradation in MSE for even 5% mismatch (MSE drop  $0.0157 \rightarrow 1.2523$ ;  $p \approx 4 \times 10^{-51}$ ,  $U$  test). At 10% simulated mismatch, comparable with deployment on mixed-signal neuromorphic devices, our spiking ADS network architecture maintained the best MSE (ADS 0.0161; Reservoir 6.19; FORCE 0.301; BPTT 0.308), performing significantly better than all other architectures ( $p < 1 \times 10^{-6}$  in all cases,  $U$  test). At 20% simulated mismatch the performance of all architectures began to degrade, but our spiking ADS architecture maintained the best MSE (ADS 0.0470; Reservoir 10.5; FORCE 0.953; BPTT 0.565;  $p < 5 \times 10^{-2}$  in all cases,  $U$  test).

We compared the effect of quantisation noise on the four architectures, examining 6–2 bits of weight precision (Fig. S2; see “Methods”). Note that no architectures were trained using quantisation-aware methods, making this



**Figure 5.** Under simulated deployment, our method is more robust to mismatch than standard training approaches. Shown are the distributions of errors (MSE) between the non-spiking teacher RNN and final trained SNN response for 10 random initialisations of training for each architecture and 10 random instantiations for each level of mismatch. Dashed black line: baseline network with no mismatch ( $\delta = 0\%$ ) for each architecture. See text for statistical comparisons.

a direct test of inherent robustness to quantisation noise. The Reservoir architecture broke down for any quantisation level (chance task performance accuracy  $\approx 50\%$ ). The FORCE architecture performed well down to 5 bits (median accuracy 85%), beyond which MSE increased and performance decayed to chance level at 3 bits (med. acc. 50%). Both the ADS spiking network and BPTT architectures maintained good performance down to 4 bits of precision (med. acc. ADS 81%; BPTT 87%), decaying to chance level at 2 bits (med. acc. ADS 52%; BPTT 54%).

We compared the effect of thermal noise of the four architectures, simulated as membrane potential noise (Fig. S4; see “Methods”). The FORCE architecture was most robust to thermal noise, performing best at all noise levels (higher accuracy,  $p < 5 \times 10^{-2}$ ; lower MSE,  $p < 1 \times 10^{-3}$  except for highest level of noise;  $U$  test). All other architectures degraded progressively with increasing noise levels. Our spiking ADS network architecture showed the smallest degradation in general over increasing noise levels (MSE 0.0083–0.115; acc. 82–67%). The BPTT architecture also fared well, while dropping in accuracy for the largest noise level (med. acc. 56%).

**Power comparison for mixed-signal and traditional implementations.** We estimated and compared the power requirements between a direct implementation of the recurrent non-spiking network dynamics on commodity and ASIC hardware, against our mixed-signal spiking implementation of the network dynamics. We performed the power comparison for the real-time audio processing task outlined above, for varying recurrent network dimensions. Computation on the DYNAP-SE1 processor occurs continuously in real-time, with no clock. We selected the slowest clock speeds for the commodity hardware that are sufficient to support real-time operation.

We estimated the power requirements for an ultra-low-power digital microcontroller from ST Microelectronics (STM32L552xx)<sup>47</sup> (see Table S1). When operating at a 16 MHz clock frequency and efficiently implementing only the recurrent dynamics required by a  $\hat{N} = 64$ -neuron non-spiking RNN, the low-power MCU was estimated to require 260  $\mu\text{W}$  when simulating with a time-step  $dt = 10$  ms, increasing to 1130  $\mu\text{W}$  for  $dt = 1$  ms. For the equivalent spiking network with  $N = 768$  spiking neurons the DYNAP-SE1 processor requires 288  $\mu\text{W}$  when fabricated at 180 nm process, and 38  $\mu\text{W}$  when fabricated at 65 nm process. For larger non-spiking RNNs, the DYNAP-SE1 processor has an increasing energy advantage over the low-power MCU.

We also considered the implementation of the non-spiking RNN on an ultra-low-power ASIC, EIE<sup>48</sup>. When implementing the dynamics required by a  $\hat{N} = 64$ -neuron non-spiking RNN, the ASIC required 11  $\mu\text{W}$  when simulating with a time-step  $dt = 10$  ms, increasing to 105  $\mu\text{W}$  for  $dt = 1$  ms. The ASIC displays a power advantage when simulating dynamics for extremely small RNNs with  $\hat{N} < 35$ , or with large time-steps  $dt = 10$  ms and  $\hat{N} < 200$ . For larger networks and with more accurate temporal dynamics, the mixed-signal SNN implementation using our approach is more energy-efficient. For further details of the power estimations see “Methods” and Table S1.

## Discussion

We propose a method for supervised training of spiking neural networks that can be deployed on mixed-signal neuromorphic hardware without requiring per-device retraining or calibration. Our approach interprets the activity of a non-spiking RNN as a teacher dynamical system. Using results from dynamical systems learning theory, our spiking networks learn to copy the pre-trained RNN and therefore perform arbitrary tasks over temporal signals. Our method is able to produce spiking networks that perform both simple and complex non-linear temporal detection and classification tasks. We show that our networks are considerably more robust to several forms of parameter and state noise, compared with several other common techniques for training spiking networks.

Our networks are by design robust to common sources of network and parameter variation, both intra- and inter-chip, which must be compensated for when deploying to mixed-signal neuromorphic hardware. For levels of mismatch measured directly from neuromorphic devices, we show that common SNN network architectures break down badly. Usual approaches for compensating for mismatch-induced parameter variation on neuromorphic hardware employ either on-device training<sup>49–53</sup> or per-device calibration<sup>14,15,53–55</sup>, entailing considerable additional expense in hardware complexity or testing time. In contrast, our method produces spiking networks that do not require calibration or retraining to maintain performance after deployment. As a result, our approach provides a solution for cost-efficient deployment of event-driven neuromorphic hardware.

The coding scheme used by our spiking networks has been shown to promote sparse firing<sup>38</sup>. For mixed-signal neuromorphic hardware, power consumption is directly related to the network firing rate. Our method therefore produces networks that consume little power compared with alternative architectures that use firing-rate encoding or do not promote sparse activity<sup>18,23,32</sup>.

Our approach to obtain high-performing SNNs is at heart a knowledge-transfer approach, relying on copying the dynamics of a highly-performing non-spiking RNN. This two-step approach is needed because the learning rule for our SNN requires a task to be defined in terms of a dynamical system, and is not able to learn the dynamics of an arbitrary input–output mapping (see Supplementary Methods). Consequently, our spiking networks can only perform as well as the pre-trained non-spiking RNN, and require multiple training steps to build a network for a new task. Nevertheless, training non-spiking RNNs is efficient when using automatic differentiation, just-in-time compilation and automatic batching<sup>56</sup>, and can be performed rapidly on GPUs. Our approach trades off between training time on commodity hardware, and immediate deployment on neuromorphic hardware with no per-device training required.

The robustness of our spiking ADS networks comes partially from the fast balanced recurrent feedback connections, which ensure sparse encoding and compensate in real-time for encoding errors<sup>38,39</sup>. These weights also degrade under noise, but can be adapted in a local untrained fashion using local learning rules that are compatible with HW implementation<sup>57</sup>.

Our supervised training approach is designed for temporal tasks, where input and target output signals evolve continuously. This set of tasks encompasses real-time ML-based signal processing and recognition, but is a poorer fit to high-resolution frame-based tasks such as frame-based image processing. These “one-shot” tasks can be mapped into the temporal domain by serialising input frames<sup>58</sup> or by using temporal coding schemes<sup>59</sup>. We found anecdotally that temporal discontinuities in input and target time series made training our ADS networks more difficult, with the implication that a careful matching between task and network time constants is important.

Our approach builds single-population recurrent spiking networks, in contrast to deep non-recurrent network architectures which are common in 2021<sup>60</sup>. Recurrent spiking networks such as Liquid State Machines (LSMs) have been shown to be universal function approximators<sup>61</sup>, but RNNs do not perform the progressive task decomposition that can appear in deep feed-forward networks<sup>62</sup>. Interpretability of the internal state of recurrent networks such as ours is therefore potentially more difficult than for deep feedforward architectures.

Neuromorphic implementation of spiking neural networks has been hailed as the next generation of computing technology, with the potential to bring ultra-low-power non-von-Neumann computation to embedded devices. However, parameter mismatch has been a severe hurdle to large-scale deployment of mixed-signal neuromorphic hardware, as it directly attacks the reliability of the computational elements — a problem that commodity digital hardware generally does not face. Previous solutions to device mismatch have been impractical, as they require expensive per-device calibration or training prior to deployment, or increased hardware complexity (and therefore cost) in the form of on-device learning circuits. We have provided a programming method for mixed-signal neuromorphic hardware that frees application developers from the necessity to worry about computational unreliability, and does not require per-device handling during or after deployment. Our approach therefore removes a significant obstacle to the large-scale and low-cost deployment of neuromorphic devices.

## Methods

We trained and simulated ANNs and SNNs using Rockpool<sup>63</sup>, an open-source Python package for machine learning of SNNs. We implemented a liquid state machine SNN<sup>18</sup>, spiking FORCE network<sup>32</sup>, and a BPTT-trained SNN<sup>23</sup> using Jax<sup>56</sup> and custom-written forward-Euler solvers. Parameters for all architectures are given in the Supplementary Material. Code to generate all models, analysis and figures in this paper are available from <https://github.com/synsense/Robust-Classification-EBN>.

**Temporal XOR task.** We created signals of a total duration of 1 second, of which the first two thirds were dedicated to the input and the last third to the target (Fig. 2). During the input time-frame, two activity bumps were created on a single input channel representing the binary inputs to the logical XOR operation. The bumps had varying length (uniformly drawn between 66–157 ms) and magnitude  $\pm 1$ , and were smoothed with a Gaussian filter to produce smooth activity transitions. In the final third of the signal we defined a target bump of



magnitude  $\pm 1$ , indicating the true output of the XOR operation. The target bump was also smoothed with a Gaussian filter. We trained a rate network ( $\hat{N} = 64$ ) to high performance on the XOR task, then subsequently trained a spiking model ( $N = 320$ ) to follow the dynamics of the trained rate network. We used a fixed learning rate  $\eta = 1 \times 10^{-5}$  and fixed error feedback rate  $k = 75$  during SNN training. Output classification from both networks was determined by the network output passing the thresholds  $\pm 0.5$ .

**Speech classification task.** We drew samples from the “Hey Snips” dataset<sup>44</sup>, augmented with noise samples from the DEMAND dataset<sup>45</sup>, with a signal-to-noise ratio of 10 dB. Each signal had a fixed length of 5 s and was pre-processed using a 16-channel bank of 2nd-order Butterworth filters with evenly-spaced centre frequencies ranging 0.4–2.8 kHz. The output of each filter was rectified with  $\text{abs}(\cdot)$ , then smoothed with a 2nd order Butterworth low-pass filter with cut-off frequency 0.3 kHz to provide an estimate of the instantaneous power in each frequency band. The rate network for the speech classification task ( $\hat{N} = 128$ ) was trained for 1 epoch on 10 000 samples to achieve roughly the same performance as the spiking network trained with BPTT. We trained spiking networks ( $N = 768$ ;  $\tau_{\text{mem}} = 50$  ms;  $\tau_{\text{fast}} = 1$  ms;  $\tau_{\text{slow}} = 70$  ms) for 5 epochs on 1000 training samples, validated on 500 validation samples and 1000 test samples. To perform a classification we integrated the output of the network when it passed a threshold of 0.5. We then applied a subsequent threshold on this integral, determined by a validation set, to determine the final prediction. We used a fixed learning rate  $\eta = 1 \times 10^{-4}$  and a decaying step function for the error feedback factor  $k$  (from 200–25 in 8 evenly-spaced steps).

**Spiking neuron model and initialisation.** We used an LIF neuron model with a membrane time constant  $\tau_{\text{mem}} = 50$  ms; reset potential  $V_{\text{reset}} = 0$ ; resting potential  $V_{\text{rest}} = 0.5$ ; and spiking threshold  $V_{\text{thresh}} = 1$ . The membrane potential dynamics for the neuron model were given by

$$\tau_{\text{mem}} \frac{\partial V}{\partial t} = V_{\text{rest}} - V + I_{\text{inp}} + I_{\text{fast}} + I_{\text{slow}} + I_{\text{e}} + \eta_n$$

with input current  $I_{\text{inp}}$ ; fast and slow recurrent post-synaptic potentials (PSPs)  $I_{\text{fast}}$  and  $I_{\text{slow}}$ ; error current  $I_{\text{e}} = k\mathbf{D}^T \mathbf{e}$ ; and noise current  $\eta$ . Output spikes from a neuron are given by  $o(t) = V > V_{\text{thresh}}$ . Synaptic dynamics were described by

$$\tau_{\text{syn}} \frac{\partial I_{*}}{\partial t} = -I_{*} + (W\mathbf{o}(t)\tau_{\text{syn}})/\Delta t$$

with input synaptic weights  $W$ ; synaptic time constants  $\tau_{\text{syn}} = 1$  ms and 70 ms for fast and slow synapses, respectively; and simulation time step  $\Delta t$ . Feed-forward and decoding weights were initialised using a standard normal distribution scaled by the number of input/output dimensions ( $\hat{N}$ ). Fast balanced recurrent feedback connections were initialised and rescaled according to the threshold and reset potential, as described in Ref.<sup>38</sup>. The spiking network was simulated using a forward Euler solver with a simulation time step of 1 ms.

**Non-spiking network.** The dynamics of a neuron in the non-spiking RNN were described by

$$\tau_j \dot{x}_j = -x_j + \hat{\mathbf{F}}c_j(t) + \hat{\Omega}f(\mathbf{x}) + b_j + \epsilon_j$$

with input  $c(t)$ ; encoding weights  $\hat{\mathbf{F}}$ ; recurrent weights  $\hat{\Omega}$ ; non-linearity  $f(\cdot) = \tanh(\cdot)$ ; bias  $b$ ; and noise term  $\epsilon$ . Time constants  $\tau$  were initialised with linearly spaced values (10 – 100 ms). The trainable parameters in this network are the time constants  $\tau$ ; the encoding and recurrent weights  $\hat{\mathbf{F}}$  and  $\hat{\Omega}$ ; and the biases  $b$ . No noise was applied during training or inference ( $\epsilon = 0$ ).

**Measurements of parameter mismatch.** Using recordings from fabricated mixed-signal neuromorphic chips we measured levels of parameter mismatch (i.e. fixed substrate noise pattern) present in hardware. In particular, for DYNAP-SE<sup>2</sup>, a neuromorphic processor which emulates LIF neuron, AMPA and NMDA synapse models with analog circuits, we measured neuron and synaptic time constants, and synaptic weights for individual neuron units, by recording and analysing the voltage traces produced by these circuits. We observed levels of mismatch in the order of 10–20% for individual parameters, with widths of the distributions being proportional to the means (see Fig. S1).

**Power estimates.** Since the input and output weighting differs between the spiking and non-spiking network, and comprises only a small portion of the parameters, we limited our power comparison to the recurrent portion of the network. Updating the recurrent dynamics for the non-spiking rate network requires multiply-accumulate operations for the recurrent input  $\mathbf{r}_t = \hat{\Omega}f(\mathbf{x}_t)$  (neglecting the transfer function  $f(\cdot)$ ); multiply-accumulate operations for the Euler solver update  $\mathbf{x}_{t+1} = \mathbf{x}_t + \dot{\mathbf{x}}_t * dt/\tau$ ; and accumulate operations for  $\dot{\mathbf{x}}_t = -\mathbf{x}_t + \mathbf{i}_t + \mathbf{b} + \mathbf{r}_t$ . With  $\hat{N} = 64$  neurons, these amount to 8576 OPs, with MACs counted as two OPs. With a time-step of  $dt = 1$  ms, this corresponds to 8.58 GOPS (Giga-OPs per second). We estimated the power to implement our RNN on non-neuromorphic NN accelerators by using previously reported power as GOPS/W. We examined only chips with published data for total power, and where we could identify the fabrication node for the published chip. We re-scaled power estimates to normalise against the fabrication node, providing estimates for 65 nm nodes in all cases. For the ultra-low-power microcontroller (STM32L552xx), we assumed that the MCU switched to a low-power sleep mode once the dynamics for a given time-step were computed. This

permits the MCU to save power when only a portion of computing resources is required to simulate real-time dynamics.

Again neglecting synaptic operations required for input and output, we estimate the energy for routing a single recurrent spike on the DYNAP-SE1 mixed-signal neuromorphic processor as 3.3 nJ. We found that the firing rate of the spiking population is upper-bounded by approximately 3 Hz per neuron during simulation. For the spiking recurrent population with  $N = 768$ , this corresponds to energy usage of  $7.6 \mu\text{W}$  dynamic power consumption. Static power consumption for the DYNAP-SE1 processor is estimated at  $30 \mu\text{W}$ . Table S1 compares the energy consumption of running the ANN on an efficient ASIC<sup>48</sup>, and a low-power general purpose MCU<sup>64</sup> to the energy consumption of the DYNAP-SE1 using the spiking network with 12 times more neurons.

**Simulated mismatch.** To simulate parameter mismatch in mixed-signal neuromorphic hardware we derived a model where the values for each parameter follow a normal distribution with the standard deviation depending linearly on the mean. The mismatched parameters  $\Theta'$  are obtained with  $\Theta' \sim \mathcal{N}(\Theta, \delta\Theta)$  where  $\delta$  determines the level of mismatch. We considered three levels of mismatch: 5, 10 and 20%.

**Quantisation noise.** We introduced quantisation noise by reducing the bit-precision of all weights post-training to 2, 3, 4, 5 and 6 bits. The weights were quantised by setting  $\mathbf{W}_{\text{disc}}^s = \rho \lfloor W/\rho \rfloor$  where  $\rho = (\max(\mathbf{W}_{\text{full}}^s) - \min(\mathbf{W}_{\text{full}}^s))/(2^b - 1)$  and  $\lfloor \cdot \rfloor$  is the rounding operator.

**Simulated thermal noise.** Thermal noise is inherent in neuromorphic devices and can be modeled by Gaussian noise on the input currents. We applied three different levels of thermal noise ( $\sigma = 0.01, 0.05, 0.1$ ) that was scaled according to the difference between  $V_{\text{reset}}$  and  $V_{\text{thresh}}$  to assure equal amounts of noise for neuron model and network architecture.

**Neuron silencing.** We created four network instances grouped into two pairs: One pair was trained with the fast recurrent feedback connections  $\Omega^f$  as described above, and the other pair with  $\Omega^f = 0$ . We then clamped 40% of the neurons of one instance of both pairs to  $V_{\text{reset}}$  while evaluating 1000 test samples.

**Benchmark network architectures.** We investigated the robustness to simulated noise for four different learning paradigms, including the FORCE method<sup>32</sup>, BPTT<sup>23</sup> and reservoir computing<sup>18</sup>. All parameters for these methods are given in Supplementary Material.

**Statistical tests.** All statistical comparisons were double-sided Mann–Whitney  $U$  tests unless stated otherwise.

## Data availability

Code to generate all models, analysis and figures in this paper are available from <https://github.com/synsense/Robust-Classification-EBN>.

Received: 7 September 2021; Accepted: 22 November 2021

Published online: 03 December 2021

## References

- Corradi, F. & Indiveri, G. A neuromorphic event-based neural recording system for smart brain–machine-interfaces. *IEEE Trans. Biomed. Circ. Syst.* **9**, 699–709 (2015).
- Moradi, S., Ning, Q., Stefanini, F. & Indiveri, G. A scalable multi-core architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (dynaps). *CoRR* (2017). [arXiv:1708.04198](https://arxiv.org/abs/1708.04198)
- Cassidy, A. S. *et al.* Truenorth: A high-performance, low-power neurosynaptic processor for multi-sensory perception, action, and cognition (2016).
- Indiveri, G. *et al.* Neuromorphic silicon neuron circuits. *Front. Neurosci.* **5**, 73. <https://doi.org/10.3389/fnins.2011.00073> (2011).
- O'Connor, P., Neil, D., Liu, S.-C., Delbruck, T. & Pfeiffer, M. Real-time classification and sensor fusion with a spiking deep belief network. *Front. Neurosci.* **7**, 178. <https://doi.org/10.3389/fnins.2013.00178> (2013).
- Schemmel, J. *et al.* A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *2010 IEEE International Symposium on Circuits and Systems (ISCAS)*, 1947–1950 (2010).
- Davies, M. *et al.* Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro* **38**, 82–99 (2018).
- Painkras, E. *et al.* Spinnaker: A 1-w 18-core system-on-chip for massively-parallel neural network simulation. *IEEE J. Solid-State Circ.* **48**, 1943–1953 (2013).
- Mead, C. Neuromorphic electronic systems. *Proc. IEEE* **78**, 1629–1636 (1990).
- Pelgrom, M., Duinmaijer, A. & Welbers, A. Matching properties of MOS transistors. *IEEE J. Solid-State Circ.* **24**, 1433–1440 (1989).
- Tuinhout, H. & Wils, N. Parametric mismatch characterization for mixed-signal technologies. In *2009 IEEE Bipolar/BiCMOS Circuits and Technology Meeting*, 107–114 (2009).
- Qiao, N. & Indiveri, G. Scaling mixed-signal neuromorphic processors to 28 nm FD-SOI technologies. In *2016 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, 552–555 (2016).
- Indiveri, G. & Sandamirskaya, Y. The importance of space and time for signal processing in neuromorphic agents: The challenge of developing low-power, autonomous agents that interact with the environment. *IEEE Signal Process. Mag.* **36**, 16–28 (2019).
- Neftci, E. & Indiveri, G. A device mismatch compensation method for VLSI neural networks. In *2010 Biomedical Circuits and Systems Conference (BioCAS)*, 262–265 (2010).
- Aamir, S. A. *et al.* An accelerated lif neuronal network array for a large-scale mixed-signal neuromorphic architecture. *IEEE Trans. Circ. Syst. I Regul. Pap.* **65**, 4299–4312 (2018).

16. Sheik, S., Chicca, E. & Indiveri, G. Exploiting device mismatch in neuromorphic vlsi systems to implement axonal delays. In *IEEE International Joint Conference on Neural Networks (IJCNN) 2012*, Proceedings of the International Joint Conference on Neural Networks, 1–6 (IEEE, 2012). <https://doi.org/10.5167/uzh-75361>.
17. Yao, E., Hussain, S., Basu, A. & Huang, G.-B. Computation using mismatch: Neuromorphic extreme learning machines, 294–297 (2013).
18. Maass, W., Natschlager, T. & Markram, H. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Comput.* **14**, 2531–2560. <https://doi.org/10.1162/089976602760407955> (2002).
19. Eliasmith, C. A unified approach to building and controlling spiking attractor networks. *Neural Comput.* **17**, 1276–1314. <https://doi.org/10.1162/0899766053630332> (2005).
20. Neckar, A. *et al.* Braindrop: A mixed-signal neuromorphic architecture with a dynamical systems-based programming model. *Proc. IEEE* **107**, 144–164 (2019).
21. Dalgaty, T. *et al.* In situ learning using intrinsic memristor variability via Markov chain Monte Carlo sampling. *Nat. Electron.* **4**, 151–161. <https://doi.org/10.1038/s41928-020-00523-3> (2021).
22. Krizhevsky, A., Sutskever, I. & Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Commun. ACM* **60**, 84–90. <https://doi.org/10.1145/3065386> (2017).
23. Werbos, P. J. Backpropagation through time: What it does and how to do it. *Proc. IEEE* **78**, 1550–1560 (1990).
24. Lee, J. H., Delbruck, T. & Pfeiffer, M. Training deep spiking neural networks using backpropagation. *Front. Neurosci.* **10**, 508. <https://doi.org/10.3389/fnins.2016.00508> (2016).
25. Neftci, E. O., Augustine, C., Paul, S. & Detorakis, G. Event-driven random back-propagation: Enabling neuromorphic deep learning machines. *Front. Neurosci.* **11**, 324. <https://doi.org/10.3389/fnins.2017.00324> (2017).
26. Bellec, G. *et al.* A solution to the learning dilemma for recurrent networks of spiking neurons. *Nat. Commun.* **11**, 3625. <https://doi.org/10.1038/s41467-020-17236-y> (2020).
27. Neftci, E. O., Mostafa, H. & Zenke, F. Surrogate gradient learning in spiking neural networks. *CoRR* (2019). [arXiv:1901.09948](https://arxiv.org/abs/1901.09948)
28. Gerstner, W., Lehmann, M., Liakoni, V., Corneil, D. & Brea, J. Eligibility traces and plasticity on behavioral time scales: Experimental support of Neohebbian three-factor learning rules. *Front. Neural Circ.* **12**, 53. <https://doi.org/10.3389/fncir.2018.00053> (2018).
29. Wunderlich, T. C. & Pehle, C. Eventprop: Backpropagation for exact gradients in spiking neural networks (2020). [arXiv:2009.08378](https://arxiv.org/abs/2009.08378)
30. Lillcrap, T. P., Cownden, D., Tweed, D. B. & Akerman, C. J. Random synaptic feedback weights support error backpropagation for deep learning. *Nat. Commun.* **7**, 13276. <https://doi.org/10.1038/ncomms13276> (2016).
31. Kaiser, J., Mostafa, H. & Neftci, E. Synaptic plasticity dynamics for deep continuous local learning (Decolle). *Front. Neurosci.* **14** (2020). <https://doi.org/10.3389/fnins.2020.00424>
32. Nicola, W. & Clopath, C. Supervised learning in spiking neural networks with force training. *Nat. Commun.* **8**, 2208. <https://doi.org/10.1038/s41467-017-01827-3> (2017).
33. Gilra, A. & Gerstner, W. Predicting non-linear dynamics by stable local learning in a recurrent spiking neural network. *eLife* **6** (2017).
34. Markram, H., Gerstner, W. & Sjöström, P. J. Spike-timing-dependent plasticity: A comprehensive overview. *Front. Synapt. Neurosci.* **4**, 2 (2012).
35. Brader, J. M., Senn, W. & Fusi, S. Learning real-world stimuli in a neural network with spike-driven synaptic dynamics. *Neural Comput.* **19**, 2881–2912 (2007).
36. Gaier, A. & Ha, D. Weight agnostic neural networks. In *Advances in Neural Information Processing Systems*, 5364–5378 (2019).
37. Bourdoukan, R., Barrett, D. G. T., Machens, C. K. & Denève, S. Learning optimal spike-based representations. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, Vol. 2, NIPS'12, 2285–2293 (Curran Associates Inc., 2012). <http://dl.acm.org/citation.cfm?id=2999325.2999390>
38. Boerlin, M., Machens, C. K. & Denève, S. Predictive coding of dynamical variables in balanced spiking networks. *PLoS Comput. Biol.* **9**, 1–16 (2013). <https://doi.org/10.1371/journal.pcbi.1003258>
39. Denève, S. & Machens, C. K. Efficient codes and balanced networks. *Nat. Neurosci.* **19**, 375–382. <https://doi.org/10.1038/nn.4243> (2016).
40. Denève, S., Alemi, A. & Bourdoukan, R. The brain as an efficient and robust adaptive learner. *Neuron* **94**, 969–977 (2017). <http://www.sciencedirect.com/science/article/pii/S0896627317304178>
41. Alemi, A., Machens, C. K., Denève, S. & Slotine, J.-J. E. Learning nonlinear dynamics in efficient, balanced spiking networks using local plasticity rules. In *AAAI*, 588–595 (2018). <https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17438>
42. Brendel, W., Bourdoukan, R., Vertechi, P., Machens, C. K. & Denève, S. Learning to represent signals spike by spike. *PLoS Comput. Biol.* **16**, e1007692 (2020).
43. Calaim, N., Alexander Dehmelt, F., Gonçalves, P. J. & Machens, C. K. Robust coding with spiking networks: A geometric perspective. *bioRxiv* (2020).
44. Coucke, A. *et al.* Efficient keyword spotting using dilated convolutions and gating. *CoRR* (2018). [arXiv:1811.07684](https://arxiv.org/abs/1811.07684).
45. Thiemann, J., Ito, N. & Vincent, E. DEMAND: A collection of multi-channel recordings of acoustic noise in diverse environments (2013). <https://doi.org/10.5281/zenodo.1227121>. Supported by Inria under the Associate Team Program VERSAMUS.
46. Barrett, D. G., Denève, S. & Machens, C. K. Optimal compensation for neuron loss. *eLife* **5**, e12454 (2016). <https://doi.org/10.7554/eLife.12454>
47. STM32L552xx Ultra-low-power Arm® Cortex®-M33 32-bit MCU+TrustZone®+FPU, 165 DMIPS, up to 512 KB Flash memory, 256 KB SRAM, SMPS (2020). <https://www.st.com/resource/en/datasheet/stm32l552cc.pdf>
48. Han, S. *et al.* Eie: Efficient inference engine on compressed deep neural network. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, 243–254 (2016)
49. Fusi, S., Annunziato, M., Badoni, D., Salamon, A. & Amit, D. J. Spike-driven synaptic plasticity: Theory, simulation, VLSI implementation. *Neural Comput.* **12**, 2227–2258. <https://doi.org/10.1162/089976600300014917> (2000).
50. Cameron, K. & Murray, A. Minimizing the effect of process mismatch in a neuromorphic system using spike-timing-dependent adaptation. *IEEE Trans. Neural Netw.* **19**, 899–913 (2008).
51. Mitra, S., Fusi, S. & Indiveri, G. Real-time classification of complex patterns using spike-based learning in neuromorphic VLSI. *IEEE Trans. Biomed. Circ. Syst.* **3**, 32–42 (2009).
52. Pfeil, T., Scherzer, A., Schemmel, J. & Meier, K. Neuromorphic learning towards nano second precision. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, 1–5 (2013).
53. Wunderlich, T. *et al.* Demonstrating advantages of neuromorphic computation: A pilot study. *Front. Neurosci.* **13**, 260. <https://doi.org/10.3389/fnins.2019.00260> (2019).
54. Costas-Santos, J., Serrano-Gotarredona, T., Serrano-Gotarredona, R. & Linares-Barranco, B. A spatial contrast retina with on-chip calibration for neuromorphic spike-based AER vision systems. *IEEE Trans. Circ. Syst. I Regul. Pap.* **54**, 1444–1458 (2007).
55. Neftci, E., Chicca, E., Indiveri, G. & Douglas, R. A systematic method for configuring VLSI networks of spiking neurons. *Neural Comput.* **23**, 2457–2497. [https://doi.org/10.1162/NECO\\_a\\_00182](https://doi.org/10.1162/NECO_a_00182) (2011).
56. Bradbury, J. *et al.* JAX: composable transformations of Python+NumPy programs (2018). <http://github.com/google/jax>
57. Büchel, J., Kakon, J., Perez, M. & Indiveri, G. Implementing efficient balanced networks with mixed-signal spike-based learning circuits (2020). [arXiv:2010.14353](https://arxiv.org/abs/2010.14353)
58. Le, Q. V., Jaitly, N. & Hinton, G. E. A simple way to initialize recurrent networks of rectified linear units (2015). [arXiv:1504.00941](https://arxiv.org/abs/1504.00941)

59. Thorpe, S., Delorme, A. & Van Rullen, R. Spike-based strategies for rapid processing. *Neural Netw.* **14**, 715–725 (2001). <http://www.sciencedirect.com/science/article/pii/S0893608001000831>
60. Sengupta, S. *et al.* A review of deep learning with special emphasis on architectures, applications and recent trends. *Knowl. Based Syst.* **194**, 105596 (2020). <http://www.sciencedirect.com/science/article/pii/S095070512030071X>
61. Maass, W. & Markram, H. On the computational power of circuits of spiking neurons. *J. Comput. Syst. Sci.* **69**, 593–616 (2004). <http://www.sciencedirect.com/science/article/pii/S002200004000406>
62. Montavon, G., Samek, W. & Müller, K.-R. Methods for interpreting and understanding deep neural networks. *Digital Signal Process.* **73**, 1–15 (2018). <http://www.sciencedirect.com/science/article/pii/S1051200417302385>
63. Muir, D., Bauer, F. & Weidel, P. *Rockpool documentation*. <https://doi.org/10.5281/zenodo.4045345> (2019).
64. ST. *STM32L552xx and STM32L562xx advanced Arm\*-based 32-bit MCUs* (2020). [https://www.st.com/resource/en/reference\\_manual/dm00346336-stm32l552xx-and-stm32l562xx-advanced-arm-based-32-bit-mcus-stmicroelectronics.pdf](https://www.st.com/resource/en/reference_manual/dm00346336-stm32l552xx-and-stm32l562xx-advanced-arm-based-32-bit-mcus-stmicroelectronics.pdf)

## Acknowledgements

This project has received funding in part by the European Union's Horizon 2020 ERC project NeuroAgents (Grant No. 724295); from the European Union's Horizon 2020 research and innovation programme for ECSEL grants ANDANTE (grant agreement No. 876925), TEMPO (grant agreement No. 826655), and SYNCH (grant agreement No. 824162); and from "Fondo di Ateneo per la ricerca 2020" (FAR2020) of the University of Sassari (grant to S. Solinas).

## Author contributions

D.R.M. and G.I. conceived and designed the research. J.B. and D.R.M. developed software and simulations. J.B., S.S. and D.Z. performed experiments and collected data. J.B. and D.R.M. analysed and interpreted the data. J.B. and D.R.M. drafted the manuscript. D.R.M., J.B., D.Z., S.S. and G.I. performed critical revision of the manuscript. D.R.M. approved the final version for publication.

## Competing interests

The authors declare no competing interests.

## Additional information

**Supplementary Information** The online version contains supplementary material available at <https://doi.org/10.1038/s41598-021-02779-x>.

**Correspondence** and requests for materials should be addressed to D.R.M.

**Reprints and permissions information** is available at [www.nature.com/reprints](http://www.nature.com/reprints).

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

© The Author(s) 2021