Fluctuation-driven initialization for spiking neural network training

NEUROMORPHIC
Computing and Engineering

OPEN ACCESS

RECEIVED
21 June 2022

REVISED
30 September 2022

ACCEPTED FOR PUBLICATION
5 October 2022

PUBLISHED
8 December 2022

PAPER

Fluctuation-driven initialization for spiking neural network training

Julian Rossbroich[1,2,3], Julia Gygax[1,2,3] and Friedemann Zenke[1,2,*]

[1] Friedrich Miescher Institute for Biomedical Research, Basel, Switzerland
[2] Faculty of Science, University of Basel, Switzerland
[3] These authors contributed equally to this work.
[*] Author to whom any correspondence should be addressed.

E-mail: friedemann.zenke@fmi.ch

Keywords: spiking neural networks, surrogate gradients, initialization strategies, recurrent neural networks, Dale's law, homeostatic plasticity, balanced networks

Supplementary material for this article is available online

Abstract

Spiking neural networks (SNNs) underlie low-power, fault-tolerant information processing in the brain and could constitute a power-efficient alternative to conventional deep neural networks when implemented on suitable neuromorphic hardware accelerators. However, instantiating SNNs that solve complex computational tasks *in-silico* remains a significant challenge. Surrogate gradient (SG) techniques have emerged as a standard solution for training SNNs end-to-end. Still, their success depends on synaptic weight initialization, similar to conventional artificial neural networks (ANNs). Yet, unlike in the case of ANNs, it remains elusive what constitutes a good initial state for an SNN. Here, we develop a general initialization strategy for SNNs inspired by the fluctuation-driven regime commonly observed in the brain. Specifically, we derive practical solutions for data-dependent weight initialization that ensure fluctuation-driven firing in the widely used leaky integrate-and-fire neurons. We empirically show that SNNs initialized following our strategy exhibit superior learning performance when trained with SGs. These findings generalize across several datasets and SNN architectures, including fully connected, deep convolutional, recurrent, and more biologically plausible SNNs obeying Dale's law. Thus fluctuation-driven initialization provides a practical, versatile, and easy-to-implement strategy for improving SNN training performance on diverse tasks in neuromorphic engineering and computational neuroscience.

1. Introduction

Spiking neurons communicate through discrete action potentials, or spikes, thereby enabling energy efficient and reliable information processing in neurobiological and neuromorphic systems [1, 2]. Before using a spiking neural network (SNN) for any application, its connections need to be task-optimized. In conventional artificial neural networks (ANNs) this step is accomplished through direct end-to-end optimization using back-propagation in combination with suitable parameter initialization [3]. However, the lack of smooth derivatives of neuronal spiking dynamics precludes using gradient-based optimization in SNNs. One increasingly common approach to overcome this issue is surrogate gradient (SG) learning [4–6] which relies on continuous relaxations of the actual gradients for parameter updates. While SGs are a powerful tool for building functional SNN models, they can be adversely affected by poor initial parameter choices. In deep ANNs, suboptimal weight initialization can lead to vanishing or exploding gradients [7–9], thereby creating a major impediment to their use. Optimal weight initialization [10–12] combined with suitable architectural choices such as skip connections [11, 13] largely avoid this issue in ANNs. Similarly, the problem of vanishing gradients has been suggested to affect deep SNNs [14, 15]. However, we still lack a principled strategy for SNN initialization.

Here, we close this gap by introducing a practical weight initialization strategy for SNNs. Specifically, we draw inspiration from neurobiology, where neuronal dynamics commonly exhibit fluctuation-driven firing [16, 17]. Since neurons in the fluctuation-driven regime are more sensitive to small changes in the input [18] and thus also to changes in their synaptic weights, we hypothesized that this regime could be advantageous for subsequent SG learning. In the following, we develop a general, yet simple initialization theory for SNNs consisting of integrate-and-fire (LIF) neurons, and empirically demonstrate its effectiveness for task-optimizing SNNs using SG techniques.

2. Results

Neurons in biological SNNs commonly exhibit irregular and asynchronous firing dynamics [16, 17, 19]. Such dynamics can often be attributed to large sub-threshold fluctuations that can naturally arise through excitatory–inhibitory balance commonly observed in neurobiology [19, 20]. To test whether this fluctuation-driven regime could constitute a suitable initial state for subsequent learning, we proceeded in two steps. First, we derived a set of compact analytical expressions that link the initial synaptic weight distribution with the magnitude of sub-threshold fluctuations. Second, we numerically tested whether initializing SNNs in the fluctuation-driven regime would allow us to rapidly train these networks to high accuracy using SGs.

To arrive at analytical expressions, we note that there are primarily three factors that contribute to the membrane potential fluctuations (figure 1(a)). These are, first, the number and firing statistics of the input neurons, second, the synaptic weight distribution, and third, the postsynaptic and neuronal parameters that govern temporal integration of the inputs. For simplicity, we assume that the presynaptic input arrives from a homogeneous population of independent Poisson neurons and that the initial weight distribution is given by a Gaussian. Further, we limited our derivation to current-based LIF neurons, which are commonly used in SNN models.

To derive an expression that links the synaptic weight distribution to the fluctuation magnitude, we consider a current-based LIF neuron with membrane potential U, whose sub-threshold dynamics are given as the sum of weighted filtered presynaptic spike trains S_j:

$$U(t) = \sum_j w_j \epsilon * S_j(t), \tag{1}$$

where $S_j = \sum_k \delta(t - t_j^k)$ denotes the output spike train of the presynaptic neuron j with firing times t_j^k and $*$ is a temporal convolution of the spike train $S_j(t)$ with ϵ, a linear filter kernel with the shape of an evoked postsynaptic potential (PSP). Specifically, we assume a synaptic model with exponentially decaying currents and, therefore, the shape of ϵ is fully characterized by the synaptic and membrane time constants τ_{syn} and τ_{mem} (see methods and supplementary material S1). Since we have many statistically independent inputs, the central limit theorem guarantees that U approaches a normal distribution which is fully specified by its mean μ_U and variance σ_U^2. Further assuming that presynaptic spikes are generated by homogeneous Poisson processes with associated firing rates $\nu_j = \langle S_j \rangle$, yields the following expressions for the mean and the variance

$$\mu_U \equiv \langle U \rangle = \sum_j w_j \nu_j \int_{-\infty}^{\infty} \epsilon(s) \mathrm{d}s = \sum_j w_j \nu_j \bar{\epsilon} \tag{2}$$

$$\sigma_U^2 \equiv \langle U^2 \rangle - \mu_U^2 = \sum_j w_j^2 \nu_j \int_{-\infty}^{\infty} \epsilon(s)^2 \mathrm{d}s = \sum_j w_j^2 \, \nu_j \hat{\epsilon}, \tag{3}$$

in which $\bar{\epsilon}$ and $\hat{\epsilon}$ correspond to definite integrals of the filter kernel and squared filter kernel respectively which can be obtained analytically for many common neuron models (supplementary material S1). For n inputs with equal firing rates $\nu_j = \nu$ and independently drawn normally distributed weights $W \sim \mathcal{N}(\mu_W, \sigma_W^2)$, the above expressions further simplify to

$$\mu_U = n \mu_W \nu \bar{\epsilon} \tag{4}$$

$$\sigma_U^2 = n(\sigma_W^2 + \mu_W^2) \nu \hat{\epsilon}. \tag{5}$$

Finally, rewriting equations (4) and (5) yields the desired expressions linking the synaptic weight distribution with the magnitude of the membrane potential fluctuations:

$$\mu_W = \frac{\mu_U}{n \nu \bar{\epsilon}} \tag{6}$$

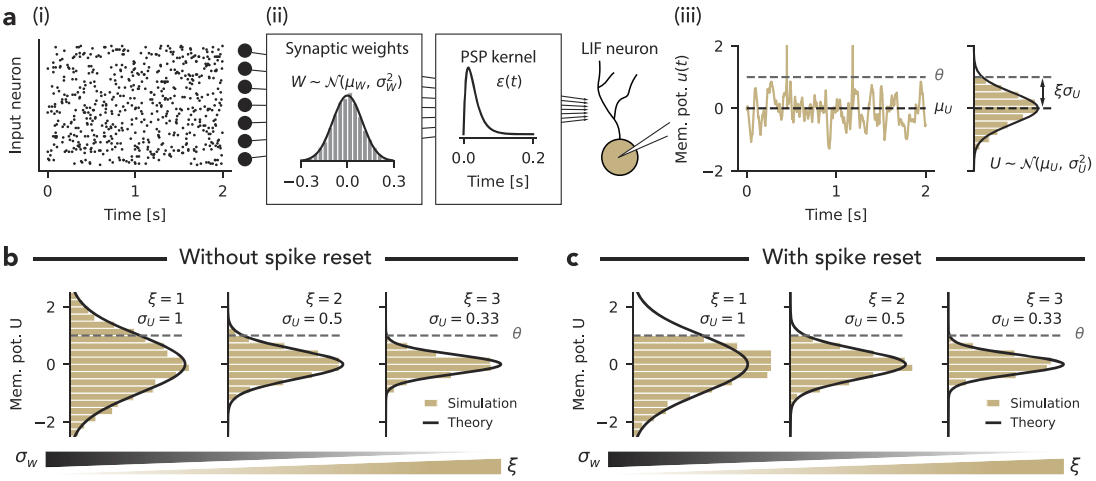

Figure 1. Parameterization of fluctuation-driven spiking serves as an initialization strategy for SNNs. (a) Incoming presynaptic Poisson spike trains (i) are weighted by synaptic strengths w_j and filtered through a PSP kernel $\epsilon(t)$ (ii) to yield membrane fluctuations $u(t)$ in a postsynaptic neuron (iii). In the fluctuation-driven regime, the membrane potential crosses the firing threshold θ stochastically, resulting in irregular output spike trains. Because the magnitude of membrane potential fluctuations, σ_U, is determined by the parameters of the presynaptic weight distribution, μ_W and σ_W, synaptic weights can be initialized from a target value for the fluctuation magnitude. (b) Expected and observed distributions of the membrane potential without considering spike-reset dynamics for different target fluctuation strengths expressed in terms of σ_U and ξ. (c) As panel (b), but considering the spike reset dynamics in the numerical simulations.

$$\sigma_W^2 = \frac{\sigma_U^2}{n\nu\hat{\epsilon}} - \mu_W^2 = \frac{1}{n\nu\hat{\epsilon}}\left(\frac{\theta - \mu_U}{\xi}\right)^2 - \mu_W^2. \tag{7}$$

For a neuron to be in the fluctuation-driven regime requires the bulk of the Gaussian distribution to lie below the firing threshold (figure 1(b)). At the same time, we require a non-vanishing probability to cross the threshold to ensure some baseline levels of spiking activity. To formalize these requirements, we introduced the target parameter ξ as

$$\xi \equiv \frac{\theta - \mu_U}{\sigma_U}, \tag{8}$$

which describes the distance between the mean membrane potential μ_U and the spike threshold θ in units of the standard deviation σ_U (figure 1(a) and supplementary figure S1). To satisfy the above requirements, ξ should be on the order of one. Concretely, we consider the range $1 \leqslant \xi \leqslant 3$ (figure 1(b)). For zero-mean weight distributions, this directly translates into a desired fluctuation amplitude range $\frac{1}{3} \leqslant \sigma_U \leqslant 1$, which, given the above assumptions, is achieved by

$$\sigma_W^2 = \frac{\sigma_U^2}{n\nu\hat{\epsilon}} \tag{9}$$

at initialization.

The expressions are based on a no-spiking assumption. Hence, we expect systematic deviations from the derived membrane potential distribution in the presence of spiking. However, for small sub-threshold fluctuations ($\sigma_U \ll 1$), the systematic contribution of the spike reset becomes negligible (figure 1(c)). Exact membrane potential distributions that take into consideration spike reset dynamics could be obtained using the Fokker–Planck equation [21, 22], however, such an approach does not yield compact analytic expressions and is, thus, less practical for our purposes.

Because equations (4) and (5) are based on an independence assumption that is violated by real-world data, we expected further deviations in numerical simulations with real-world data. To quantify the magnitude of these deviations, we compared the predictions of equations (4) and (5) with observed membrane potential fluctuations in a single LIF neuron exposed to inputs from two realistic datasets. For simplicity, we assumed a zero-mean weight distribution and used equation (9) to obtain its standard deviation for different target fluctuation magnitudes σ_U.

First, we considered a synthetic classification dataset based on random manifolds that can flexibly generate SNN benchmarks of arbitrary complexity [5] (Randman; see methods). We generated a dataset with $n_{\text{Randman}} = 20$ input neurons and 10 classes in which spike times belonging to the same class are drawn from a smooth random manifold (figure 2(a)) all the while different classes correspond to different manifolds. For

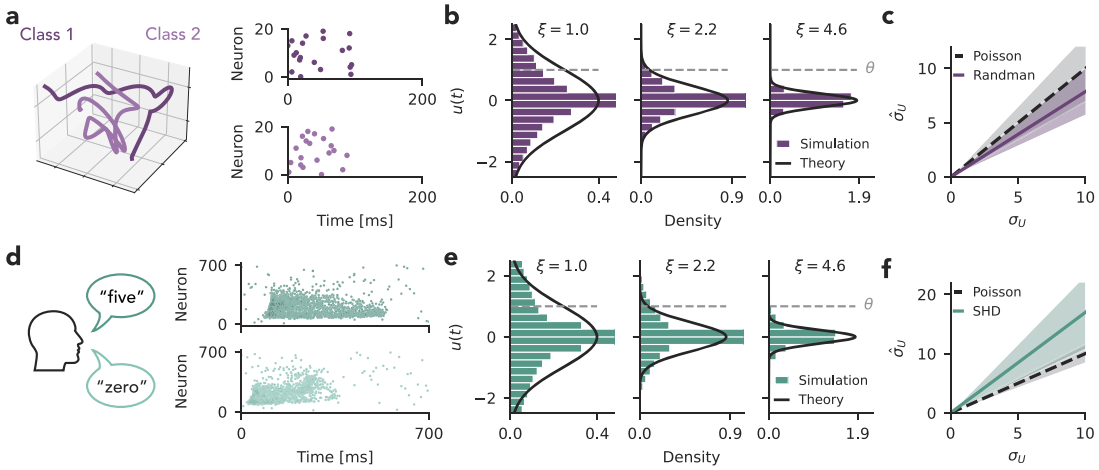

Figure 2. Real-world datasets induce small systematic biases in fluctuation strength at initialization. (a) Two one-dimensional example manifolds from the Randman dataset, embedded into a three-dimensional space (left) and example spike raster plots corresponding to a sample from each class (right). (b) Theoretically expected distribution and numerically obtained density histogram of the membrane potential of a single neuron without spike reset in response to the Randman dataset. Because of large peaks at $u(t) = 0$, the x-axes in the first and middle panels have been truncated to 45% and 80% of their maximum, respectively. (c) Numerically observed $\hat{\sigma}_U$ as a function of the target σ_U for the Randman dataset. The expected relationship corresponds to homogeneous and independent Poisson neurons. Shaded regions indicate standard deviation across neurons. (d) Two spike rasters that correspond to two example inputs from the SHD dataset. Input spikes are obtained by filtering recordings of spoken digits with a biologically inspired cochlear model [23]. (e) As panel (b), for the SHD dataset. X-axes in the first and middle panels have been truncated to 58% and 90% of their maximum, respectively. (f) As panel (c), for the SHD dataset.

each input pattern, each neuron fires precisely one spike during a 100 ms interval. Each 100 ms input interval was followed by 100 ms of inactivity in the input layer to allow for a propagation delay in the hidden layer (figure 2(a)). We then recorded the membrane potential distribution and found, as expected, that it deviated from a Gaussian (figure 2(b)), due to the temporal non-stationarity and structure. Next, we measured the observed membrane potential fluctuations $\hat{\sigma}_U$ for varying target values of σ_U (figure 2(c)). We found that $\hat{\sigma}_U$ was systematically smaller than σ_U. However, the magnitude of bias was comparable to the expected variability in the case of Poisson inputs (see supplementary material S2).

Next, we considered the Spiking Heidelberg Digits (SHD) speech dataset (figure 2(d)), an SNN benchmark based on real-world auditory data, which consists of approximately 10 000 spoken digits in German and English that have been converted into spikes using a biologically plausible cochlear model [23]. Importantly, SHD has a larger number of input neurons ($n_{\text{SHD}} = 700$) which typically fire more than one spike with an average input firing rate of $\nu_{\text{SHD}} = 15.8$ Hz. Again, we measured the membrane potential distribution and observed deviations from a Gaussian (figure 2(e)). In contrast to the Randman data, the observed fluctuations $\hat{\sigma}_U$ were systematically larger than their target σ_U due to heavy tails in the distribution (figure 2(f)). Not surprisingly, real-world data causes systematic deviations from equations (4) and (5), but these differences were on the same order as expected fluctuations due to the finite sample size of the weight and Poisson variability. Hence, we reasoned that our simple theory provides a reasonable approximation for initializing SNNs in the fluctuation-driven regime even when using real-world data.

2.1. Initialization of shallow SNNs

We sought to evaluate whether the fluctuation-driven regime constitutes a good initialization strategy for SNN training. To this end, we trained a fully connected SNN with one hidden layer with 128 units on the Randman dataset (see table 4; methods). We initialized the weights using the parameters $\mu_W = 0$ and σ_W given by our theory (equation (9) with target $\sigma_U = 1$). This choice resulted in asynchronous irregular firing activity consistent with the fluctuation-driven regime (supplementary figure S2(a-d)). Subsequently, we trained the network in a supervised fashion using SGs with previously established parameters [5], back-propagation through time (BPTT), a maximum-over-time loss defined on ten readout units, and weak spiking activity regularization in the form of a soft upper bound on the population firing rate at the hidden layer (figure 3(a); methods). Training resulted in an SNN that accurately solved the task (test accuracy: $97.3\% \pm 0.2$; train accuracy: $99.6\% \pm 0.0$; figure 3(b)).

To test whether our weight initialization strategy confers an advantage over other choices of μ_W and σ_W, we performed an extensive parameter search and measured validation accuracy after 200 training epochs. The network achieved the best validation accuracy when μ_W was zero or negative and σ_W was close to one (figure 3(c)), well within our suggested regime of $1 \leqslant \xi \leqslant 3$. Further, we found a large parameter regime that supported

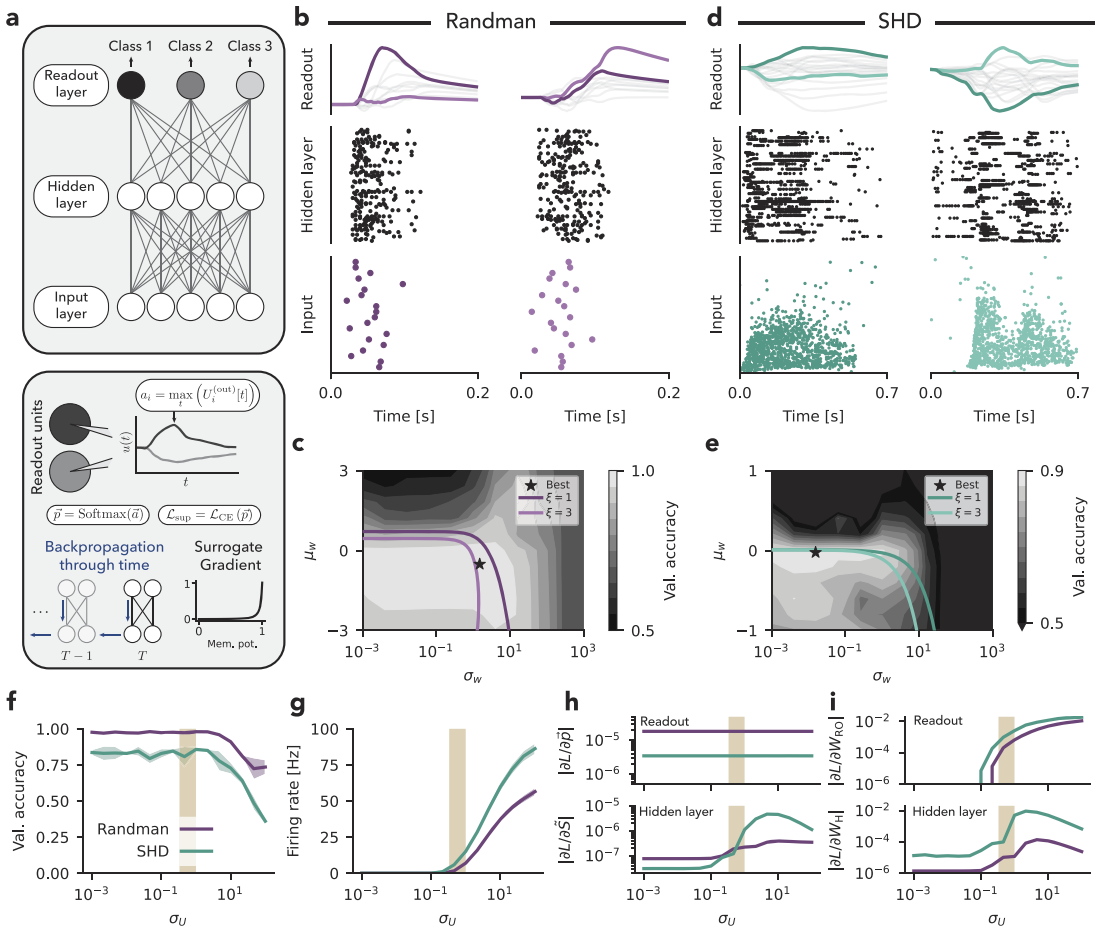

Figure 3. Initialization in the fluctuation-driven regime results in optimal learning performance. (a) Top: schematic of the SNN used for training. Bottom: illustration of the learning dynamics. The supervised loss function $\mathcal{L}_{\text{sup}}$ relies on the maximum membrane potential over time of readout units $U_i^{(\text{out})}[t]$, to which a Softmax and cross-entropy loss $\mathcal{L}_{\text{CE}}$ is applied. All networks were trained by minimizing $\mathcal{L}_{\text{sup}}$ in the direction of negative SGs, computed with BPTT. (b) Snapshot of network activity over time after training on the Randman dataset. Bottom: spike raster of input layer activity from two different samples corresponding to two different classes is shown. Middle: spike raster of hidden layer activity. Top: membrane potential of readout units. The readout units corresponding to the two input classes are highlighted in different shades. (c) Heatmap showing validation accuracy after training on the Randman dataset as a function of the parameters of the synaptic weight distribution at initialization. (d) Same as in panel (b), but for a network trained on the SHD dataset. (e) Same as panel (c), but for the SHD dataset. (f) Validation accuracy as a function of target fluctuation magnitude σ_U for initializations in the balanced state with $\mu_U = \mu_W = 0$. The shaded region around the lines indicates the range of values across five random seeds. The sand-colored shaded region corresponds to our suggested target fluctuation magnitude $\frac{1}{3} \leqslant \sigma_U \leqslant 1$. (g) Average hidden layer firing rate as a function of σ_U. (h) Average magnitude of SGs with respect to the output in the readout layer (top) and hidden layer (bottom) as a function of σ_U. (i) Same as panel (h), but for the average magnitude of SGs with respect to the synaptic weights.

learning at close-to-optimal accuracy for $-2 \leqslant \mu_W \leqslant 0$ and $\sigma_W < 10$ which extends beyond the parameter regime suggested by our theory.

To test whether these results would change on a more complex task, we trained a similar SNN on the SHD dataset [23] with weight parameters $\mu_W = 0$ and $\sigma_W^{(\text{SHD})} = 0.23$ as suggested by our theory (equation (9) with target $\sigma_U = 1$). Due to differences of the number of input neurons and firing rates between the two datasets our theory predicts $\sigma_W^{(\text{SHD})} \approx 10^{-1}\sigma_W^{(\text{Randman})}$. After training, the network accurately classified spoken digits (test accuracy: $65.5\% \pm 0.7$; train accuracy: $100.0\% \pm 0.0$; figure 3(d)). As before, we performed an extensive parameter search over different initializations and found that networks initialized in the fluctuation-driven regime ($1 \leqslant \xi \leqslant 3$) showed close-to-optimal performance (figures 3(e) and (f)). Unlike in the Randman case, the parameter regime with good performance was much smaller and tightly constrained around $\mu_W \approx 0$. Finally, even though our initialization strategy posits that neurons be in the fluctuation-driven regime, we observed a sizeable fraction of hidden layer neurons with regular firing activity both before (supplementary figure S2e) and after learning (figure 3(d)). We found that our theory predicts these cases (supplementary figures S2(d) and (h)) due to the inherent variability in the sampling of synaptic weights (supplementary material S2 and supplementary figure S3).

For both datasets, we found that initialization with $\sigma_W \ll 1$ and $\mu_W \approx 0$ supported close-to-optimal learning. This result surprised us because the ensuing vanishing membrane potential fluctuations should lead to

quiescent hidden layer activity. To check whether this is indeed the case, we initialized networks with different target values for σ_U and recorded their hidden layer activity. As expected, we found that fluctuation magnitudes $\sigma_U \ll 1$ still supported close-to-optimal learning performance (figure 3(f)), despite an absence of spikes in the hidden layer at the time of initialization (figure 3(g)).

Because vanishing spiking activity should influence gradient magnitudes during backpropagation, we recorded the magnitude of the SG with respect to the output at the readout and hidden layers at the time of the first training epoch. Due to the nature of the loss function, initialization does not affect the magnitude of the gradient in the readout layer but can change the magnitude of the gradient by two orders of magnitude in the hidden layer (figure 3(h)). Consequently, the absolute magnitude of weight changes is also amplified in the hidden layer when fluctuations are large (figure 3(i)). Since the synaptic weight update depends on presynaptic activity, initializations resulting in quiescent hidden layers (figure 3(g)) lead to an absence of weight updates in the readout layer (figure 3(i)). However, as long as SGs do not vanish in the first layer, the network can recover spike propagation and therefore gradient flow during training. That the network is able to learn without problems in this regime may seem surprising at first and is indeed a peculiarity of SGs.

In addition to classification accuracy, the sparsity of neuronal activity is a key SNN performance indicator. To limit firing rates in the hidden layers to a sensible regime, we optimized all networks with activity regularization. Specifically, we added a soft upper bound on the population firing rate (see methods). This regularization punishes population firing rates in the hidden layers exceeding 10 Hz. To investigate the effect of weight initialization on sparsity, we systematically recorded population firing rates of the above network trained with or without activity regularization. As expected, activity regularization resulted in average population firing rates of $<$10 Hz following training, independent of the target σ_U at initialization. In contrast, networks trained without activity regularization exhibited population firing rates exceeding 60 Hz in the hidden layer and only weak dependence on the target σ_U at initialization (supplementary figures S4(a) and (b)). Next, we wanted to ensure that activity regularization does not result in a substantial loss in classification accuracy. To that end, we compared the accuracy of networks trained with and without activity regularization for the given threshold and strength parameters. We found that regularized networks performed only slightly worse than their unregularized counterparts albeit with vastly reduced average firing rates (supplementary figure S4(c)). Based on these findings, we used activity regularization on the population firing rates in all subsequent experiments.

Thus far, we studied strictly feed-forward SNNs without recurrent hidden layer connections. Recurrent SNNs typically perform better than feed-forward networks on tasks requiring memory such as SHD [5]. To that end, we extended our initialization strategy to networks with recurrent connections (see methods) and applied it to recurrent SNNs with one hidden layer. As in the case of feed-forward networks, we found recurrent SNNs trained well with sufficiently small target fluctuations σ_U (supplementary figures S5(a) and (b)). In summary, shallow SNNs are surprisingly robust to initialization when the absolute magnitude of the weights is small. In practice, initialization with $\mu_U = 0$ and a target fluctuation magnitude $\sigma_U \leqslant 1$ can be used to achieve close-to-optimal learning performance.

2.2. Initialization of deep SNNs

We hypothesized that deep SNNs are more sensitive to initialization, as is the case with deep ANNs [11]. To test this hypothesis, we first extended our initialization strategy to deep and recurrent convolutional spiking neural network (CSNN) architectures (see methods). We then initialized several CSNN architectures with different numbers of recurrently connected hidden layers according to equations (37)–(39) with target $\mu_U = 0$ and different targets σ_U. Subsequently, we trained the resulting networks and measured validation accuracy on held-out data. As expected, sensitivity to the fluctuation magnitude at initialization increased with network depth (figure 4(a)). As in shallow fully connected networks, CSNNs with a single hidden layer were remarkably robust to initialization and close-to-optimal training performance was achieved for $\sigma_U \leqslant 1$. In contrast, deep networks with three hidden layers performed well when the fluctuation magnitude fell into the regime $0.05 \leqslant \sigma_U \leqslant 3$. This regime was narrowed further in deeper networks with seven hidden layers, which only achieved high validation accuracy for initializations in the range $0.5 \leqslant \sigma_U \leqslant 2$. Like in the shallow case, activity regularization ensured sparse activity with a negligible effect on classification accuracy (supplementary figures S4(d) and (e)). Finally, although a seven-layer CSNN did not improve classification performance on this task over the three-layer network, we wanted to know whether initialization with $\sigma_U = 1$ would be conducive for training even deeper networks. To get at this question, we extended our network to ten hidden layers, the deepest possible architecture afforded by our GPU memory while keeping all other training parameters equal to networks with seven layers, and found that initialization with $\sigma_U = 1$ resulted in reliable training (test accuracy: $81.1\% \pm 1.6$; validation accuracy: $93.6\% \pm 2.9$). Crucially, when we instead trained with Kaiming

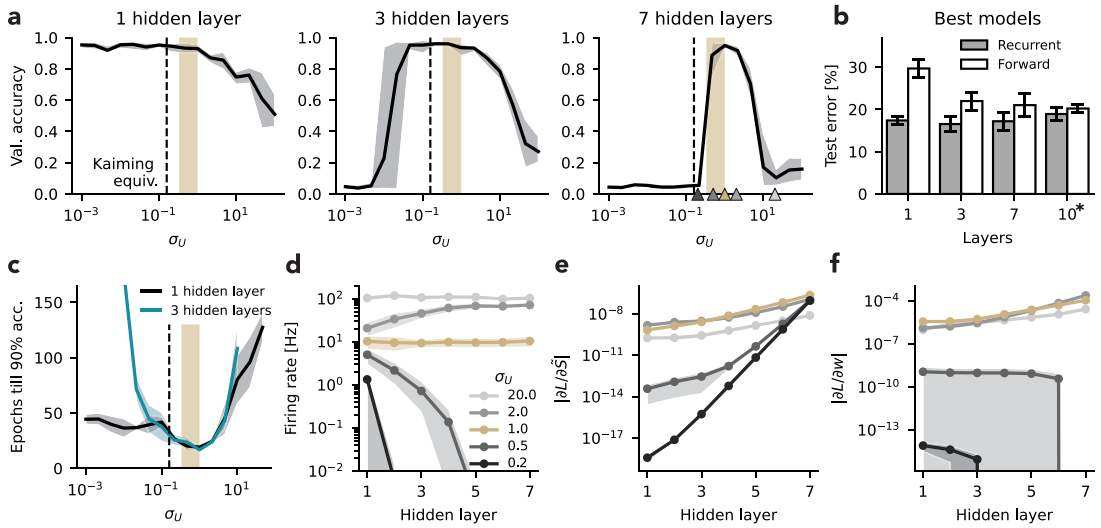

Figure 4. Deep CSNNs are sensitive to initialization due to vanishing SGs. (a) Validation accuracy as a function of target fluctuation strength σ_U for recurrent CSNNs of increasing depth. All networks were trained on the SHD dataset. The triangular markers in the right plot correspond to the values of σ_U plotted in panels (d)–(f). The shaded region around the lines indicates the range of values across five random seeds. The sand-colored shaded region corresponds to our suggested target fluctuation magnitude $\frac{1}{3} \leqslant \sigma_U \leqslant 1$. The dashed line corresponds to Kaiming initialization. (b) Test error of the five best-performing models in terms of validation accuracy, for different numbers of hidden layers and for networks with and without recurrent connections in the hidden layers. *No initialization parameter sweeps were performed for networks with ten hidden layers. Instead, the data depict results obtained from five networks initialized with target $\sigma_U = 1$. (c) Training speed of CSNNs, as illustrated by the number of required epochs to reach 90% training accuracy on the SHD dataset. (d) Population firing rate at initialization (before training) as a function of hidden layers in a CSNN with seven hidden layers, for different values of σ_U. (e) As panel (d), but displaying the magnitude of SGs. (f) As panel (d), but displaying the magnitude of the synaptic weight update. When membrane potential fluctuations are so small that neurons in the previous layer do not spike, the weight update equals zero.

initialization [11], the standard initialization for non-spiking rectified linear unit (ReLU) networks, learning failed in CSNNs with seven or more hidden layers. In summary, we observed that fluctuation-driven initialization with $\sigma_U = 1$ supports learning in deep CSNNs.

To check whether depth increases the generalization performance of trained networks, we compared the test error of successfully trained CSNNs with one, three, seven, and ten hidden layers. We found that deeper networks did not show better generalization performance than one-layer networks (figure 4(b)). These findings suggest that the addition of multiple hidden layers does not provide an advantage in recurrently connected CSNNs on the SHD dataset. Since recurrently connected networks can be considered as deep in time, we were wondering whether strictly feed-forward SNNs would benefit from increasing depth. To that end, we repeated training of deep CSNNs with corresponding layer sizes but without recurrent hidden layer connections on the SHD dataset (see methods). Indeed, we found that deep feed-forward SNNs performed better than shallow feed-forward SNNs (figure 4(b) and supplementary figure S6).

In addition to the classification accuracy, weight initialization affects training speed. To check whether fluctuation-driven initialization is conducive to fast training, we measured the number of required epochs to reach 90% accuracy on the training dataset in CSNNs with one and three hidden layers. We found that networks initialized in the fluctuation-driven regime ($\sigma_U = 1$) trained fastest (figure 4(c) and supplementary figure S7). On average, CSNNs with one hidden layer initialized with target $\sigma_U = 1$ reached 90% training accuracy after 19.2 epochs, and CSNNs with three hidden layers required 16.8 epochs to reach 90% training accuracy. Thus, fluctuation-driven initialization is conducive to fast training.

Vanishing SGs impair learning in deep SNNs. In deep ANNs initialization is closely related to the problem of vanishing or exploding gradients [7–9]. We wondered whether this mechanism, i.e., vanishing or exploding SGs, prevented training in deep SNNs when σ_U falls outside the optimal regime. To test this idea, we initialized seven-layer CSNNs with different targets σ_U and recorded the neuronal activity in hidden layers. Like in shallow SNNs (figure 3(g)), initializations with small σ_U led to quiescent hidden layers in deep CSNNs, which impaired the activity propagation to deeper layers (figure 4(d)). Specifically, in networks initialized with $\sigma_U = 0.5$, only the first four hidden layers exhibited spiking activity. This effect was amplified in networks initialized with $\sigma_U = 0.2$, in which all but the first hidden layer were quiescent. In contrast, networks initialized with $\sigma_U = 2$ exhibited a strong increase in firing rates in deeper layers, and initializations with $\sigma_U = 20$ caused firing rates to saturate in all layers of the network. Only initializations with $\sigma_U = 1$ led to stable activity propagation with a firing rate of $\approx$10 Hz throughout the network.

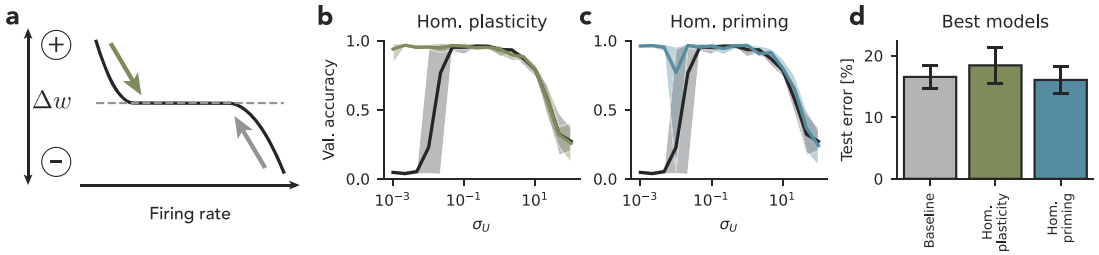

Figure 5. Homeostatic plasticity increases the robustness to initialization in deep SNNs. (a) Illustration of the homeostatic activity mechanism as a firing rate regularizer. Homeostatic plasticity (green) prevents neurons from remaining silent by increasing the synaptic weights when the firing rate is low. In all our simulations, a complementary upper bound activity regularizer (gray), that acts on the population-level, prevents neurons from spiking incessantly. (b) Validation accuracy after training a deep convolutional SNN with three hidden layers on the SHD dataset as a function of σ_U. The colored line corresponds to networks trained with an active homeostatic plasticity mechanism. The black line corresponds to the baseline without homeostatic plasticity. The shaded region around the lines indicates the range of values across five random seeds. (c) As panel (b), for networks that were primed for 10 epochs with a homeostatic plasticity mechanism prior to supervised learning. During supervised learning, the homeostatic mechanism was inactive. (d) Test error of the five best-performing models in terms of validation accuracy for models trained with homeostatic plasticity, homeostatic priming, and the baseline model.

We next investigated how impaired activity propagation influenced SG magnitudes. To that end, we recorded SG magnitudes at each hidden layer during training. In networks initialized with $\sigma_U = 0.5$ and $\sigma_U = 0.2$, in which spiking activity vanished in deep layers, each quiescent layer decreased SGs by approximately two orders of magnitude (figure 4(e)). As a result, the magnitude of weight updates in early layers decreased by several orders of magnitude consistent with the numerical value of the surrogate derivative for neurons at rest (0.023 for $\beta = 20$; see methods). Moreover, weight updates vanished in deeper layers, caused by the lack of presynaptic activity (figure 4(f)). In contrast, initializations with $\sigma_U \geqslant 1$ led to relatively stable SG and weight update magnitudes across all layers (figures 4(e) and (f)). Notably, gradients were consistently one to two orders of magnitude smaller in networks initialized with $\sigma_U = 20$ compared to networks initialized with $\sigma_U = 1$ or $\sigma_U = 2$ (figures 4(e) and (f)).

In summary, the sensitivity to initialization in deep SNNs is caused by impaired activity propagation to deeper layers and associated vanishing SGs. Empirically we found that only initializations with $\sigma_U \approx 1$ supported both propagation of sparse population activity and stable magnitudes of back-propagating SGs in deep networks.

Since the surrogate derivative used to compute SGs is to some extent freely tunable [5], one might argue that re-scaling it could provide a potential solution to vanishing SGs by ensuring stable gradient magnitudes during back-propagation (see methods). We tested this approach and found that a re-scaled SG can only prevent vanishing gradients in the absence of spiking at the cost of exploding gradients when the network does exhibit spiking which emerges over training (supplementary figures S8(a)–(c)). In strictly feed-forward networks, we found that the gradients were less prone to exploding, hence re-scaling the SG could potentially alleviate the problem of vanishing gradients (supplementary figures S8(d)–(f)) and therefore increase robustness to initialization. However, with increasing depth, exploding gradients would likely prevent successful training even in deep feed-forward SNNs.

Seeing that training of deep SNNs was sensitive to the magnitude of SGs [24], we speculated that the robustness to weight initialization we observed in three-layer CSNNs could be attributed to the use of our optimizer with a per-parameter learning rate during training (see methods). To test this idea, we trained three-layer CSNNs initialized with different σ_U either with a smart optimizer [25, 26] or with stochastic gradient descent (SGD) without an optimizer. We found that networks trained with SGD were indeed more sensitive to the fluctuation magnitude at initialization (supplementary figure S9). This effect was especially prominent in recurrent CSNNs.

Homeostatic plasticity increases robustness to initialization in deep SNNs. Because quiescent hidden layers are closely linked to vanishing SGs and thus to preventing training in deep SNNs, a homeostatically maintained firing rate, as observed in biological neural networks [27–29], could rescue activity propagation and therefore enable training. To test this hypothesis, we implemented homeostatic plasticity as an additional regularization term in the loss function that sets a lower bound on the firing rate of each individual neuron [23], which penalizes quiescent neurons (figure 5(a); see methods). We trained three-layer recurrent CSNNs on the SHD dataset, either with or without the additional homeostatic plasticity term. Indeed, homeostatic plasticity rescued training performance for networks initialized with $\sigma_U \ll 1$ (figure 5(b)).

Next, we investigated whether homeostatic plasticity was necessary throughout the whole training period, or whether rescuing activity propagation before supervised training would be sufficient to enable learning.

Table 1. Test accuracy in percent after training networks with different numbers of hidden layers and different initializations (Kaiming, Kaiming with homeostatic plasticity and fluctuation-driven initialization with $\sigma_U = 1$) on the SHD, CIFAR-10, and DVS-Gesture datasets. Errors correspond to the standard deviation.

	SHD		CIFAR-10		DVS-Gesture	
	$n_H = 3$	$n_H = 7$	$n_H = 2$	$n_H = 4$	$n_H = 6$	$n_H = 8$
Kaiming	**83.1 ± 1.2**	4.5 ± 0.0	59.5 ± 0.8	10.0 ± 0.0	54.6 ± 37.1	9.1 ± 0.0
Kaiming & Hom.	—	77.0 ± 2.9	—	**70.3 ± 0.9**	—	82.3 ± 5.3
Fluct.-driven	82.7 ± 1.1	**83.5±1.3**	**62.4 ± 0.3**	65.6 ± 1.3	**86.7 ± 1.2**	**86.4 ± 1.7**

To this end, we developed a form of dynamic initialization for SNNs involving a homeostatic priming period before training. During the initial priming period, initialized networks were optimized solely on the homeostatic objective to nudge the spiking activity into a regime conducive to learning. After priming, we removed the homeostatic objective and started the supervised training period as usual. Like ongoing homeostatic plasticity, the homeostatic priming period was capable of rescuing learning for initializations with $\sigma_U \ll 1$ (figure 5(c)). However, in rare cases, the network did not train after successful priming and the restored spiking activity vanished during training on the supervised loss function.

We wondered whether homeostatic plasticity affected the network's generalization performance and thus compared the test error of networks trained with the proposed homeostatic mechanisms. Neither ongoing homeostatic plasticity nor homeostatic priming had a systematic effect on the test error (figure 5(d)). Therefore, we concluded that both biologically inspired homeostatic plasticity and homeostatic priming are effective strategies to increase the robustness toward initialization in deep SNNs without impairing their performance.

Deep SNNs with skip connections are more robust to initialization. In deep ANNs, skip connections are standard practice to facilitate optimization and improve training performance [13, 30, 31]. For instance, residual networks [31] use residual connections, a specific type of identity skip connections whereby the inputs are added directly to the output of a layer or block. We argued that residual connections are ill-defined in SNNs as the spiking non-linearity would only allow adding spikes to the input spike train. Instead, we considered classic skip connections and asked whether they rescue spike propagation in deep CSNNs. We tested this idea in CSNNs with three hidden layers by implementing trainable skip connections between each hidden layer and the readout layer (supplementary figure S10a; methods). Skip connections indeed increased robustness to initialization, with respect to both large $\sigma_U > 10$ and small $\sigma_U \ll 1$ (supplementary figure S10(b)). However, generalization performance after training did not increase as a result of added skip connections (supplementary figure S10(d)). Notably, for initializations with small $\sigma_U \ll 1$, optimized networks only propagated activity through the skip connection between the first hidden layer and the readout layer, effectively reducing the network to a single hidden layer. As skip connections did not prevent all layers from being quiescent in deep SNNs, we wondered whether homeostatic plasticity and skip connections complement each other and further increase performance for initializations with $\sigma_U \ll 1$. Thus, we trained three-layer CSNNs with skip connections and ongoing homeostatic plasticity. Networks with combined skip connections and homeostatic plasticity also exhibited enhanced robustness to initialization but did not show a significantly better generalization performance (supplementary figures S10(c) and (d)). We concluded that skip connections are a viable approach to increase the robustness toward initializations with large $\sigma_U > 10$ in deep CSNNs, but are not able to compensate for vanishing gradients in deep layers when $\sigma_U \ll 1$.

Fluctuation-driven initialization performs robustly across datasets. Together, our results suggest that traditional Kaiming initialization used for ANNs is sufficient for training three-layer CSNNs, but breaks down when training seven-layer or deeper CSNNs on the SHD dataset. In contrast, our proposed initialization strategy with the target fluctuation parameter set to $\sigma_U = 1$ yields close-to-optimal training performance in all three-, seven-, and even ten-layer networks. To directly compare fluctuation-driven and Kaiming initialization, we measured generalization performance in terms of test accuracy after training. As expected, we found only small differences in test accuracy for three-layer networks (figure 6(a); table 1).

Specifically, Kaiming initialized three-layer networks achieved an average test accuracy of $83.1\% \pm 1.2$ (validation accuracy: $95.9\% \pm 1.6$), while the same networks initialized with our proposed strategy reached an average test accuracy of $82.7\% \pm 1.1$ (validation accuracy: $94.1\% \pm 1.7$). Seven-layer networks initialized with Kaiming initialization performed close to chance level after training (test accuracy: $4.5\% \pm 0.0$; validation accuracy: $4.7\% \pm 0.5$), while networks initialized with $\sigma_U = 1$ reached $83.5\% \pm 1.3$ accuracy on the test set (figure 6(b); table 1; validation accuracy: $94.9\% \pm 1.0$). As homeostatic plasticity was able to compensate for suboptimal initializations by rescuing activity propagation in three-layer CSNNs, we wondered whether these results extend to seven-layer networks. To this end, we trained Kaiming-initialized seven-layer CSNNs with

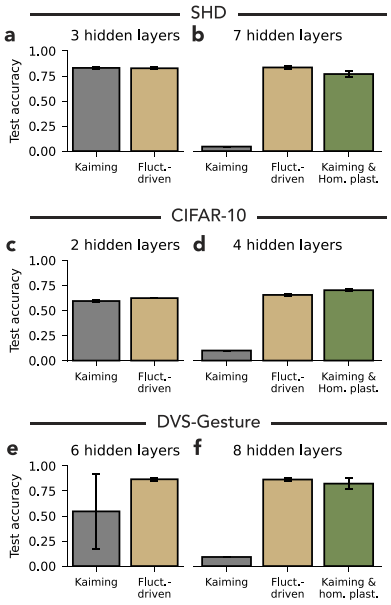

Figure 6. Fluctuation-driven initialization enables training of deep SNNs across multiple datasets. (a) Test accuracy of three-layer CSNNs trained on the SHD dataset. Networks were initialized either with standard Kaiming initialization (Kaiming) or fluctuation-driven initialization with $\sigma_U = 1$. All error bars indicate standard deviation across five runs. (b) Test accuracy of seven-layer CSNNs trained on the SHD dataset. Networks with Kaiming initialization were additionally trained with ongoing homeostatic plasticity (Kaiming & Hom. plast.). (c) Same as panel (a), but for two-layer feed-forward CSNNs trained on the CIFAR-10 dataset. (d) Same as panel (b), but for four-layer feed-forward CSNNs trained on the CIFAR-10 dataset. (e) Same as panel (a), but for six-layer feed-forward CSNNs trained on the DVS-Gesture dataset. (f) Same as panel (b), but for eight-layer feed-forward CSNNs trained on the DVS-Gesture dataset.

ongoing homeostatic plasticity. Indeed, homeostatic plasticity rescued training, but test accuracy after training (test accuracy: $77.0\% \pm 3.0$; validation accuracy: $95.0\% \pm 1.8$) was worse compared to networks initialized with $\sigma_U = 1$ that were trained without homeostatic plasticity (figure 6(b); table 1).

So far, we have limited our investigation to initialization-dependence to deep CSNNs trained on the SHD dataset, which is relatively small and may thus be prone to overfitting. To test whether our findings would generalize to other tasks, we trained deep feed-forward CSNNs on two additional datasets from different input modalities. First, we considered CIFAR-10, a dataset consisting of static images. To translate static image input into spiking, we augmented the networks with an additional layer of simulated sensory neurons into which we injected the individual image pixel values as currents. Both bias currents and current gain were optimized end-to-end with all other network parameters (see methods). We then constructed deep CSNNs with increasing numbers of hidden layers (see table 6; methods). As before, networks were either initialized with traditional Kaiming initialization or with a target membrane potential fluctuation magnitude of $\sigma_U = 1$. We observed that networks with up to two hidden layers showed good training performance with both initializations (figure 6(c); table 1). When we increased the number of hidden layers to four, networks initialized with $\sigma_U = 1$ continued to show good training performance, while networks initialized with Kaiming initialization failed to train (figure 6(d); table 1). Training on CIFAR-10 with ongoing homeostatic plasticity was able to rescue learning in Kaiming initialized SNNs with four hidden layers.

Since CIFAR-10 is a still image dataset, which lacks temporal dynamics, it is less well suited for assessing SNNs performance. To check whether our results generalize to other commonly used SNN datasets, we considered the DVS-Gesture dataset [32], which consists of short videos that depict humans performing different hand gestures. These videos were recorded using an event camera, yielding event-based outputs that can be used to train SNNs on the classification of the performed gestures. As before, we initialized deep CSNNs with an increasing number of hidden layers using either Kaiming initialization or a target $\sigma_U = 1$ and compared their test accuracy after training (see table 6; methods). We found that networks with up to six hidden layers could be successfully trained using either Kaiming or our proposed initialization (figure 6(e); table 1). However, in six-layer networks, initialization with $\sigma_U = 1$ yielded more reliable training performance and higher accuracy than Kaiming initialization. When we increased the number of hidden layers to eight, networks initialized with Kaiming initialization did not train successfully, while networks initialized with a target $\sigma_U = 1$ continued to show good learning performance (figure 6(f); table 1). As already observed on the SHD and CIFAR-10 datasets, training of Kaiming initialized deep networks could be rescued by adding homeostatic plasticity during training.

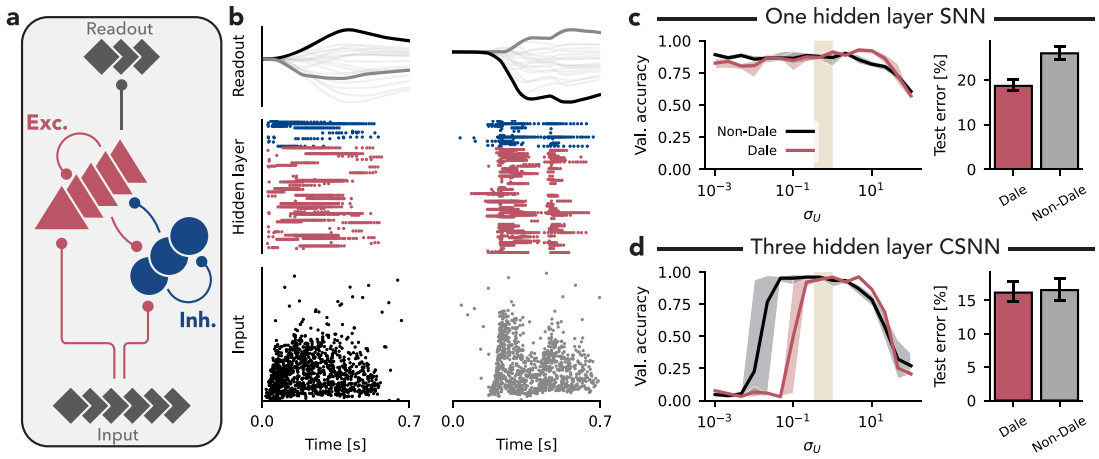

Figure 7. Initialization of Dalian SNNs in the fluctuation-driven regime. (a) Schematic of a shallow SNN obeying Dale's law. Excitatory (red) and inhibitory (blue) populations are recurrently connected, but separate. (b) Snapshot of network activity over time after training a shallow SNN obeying Dale's law on the SHD dataset. Bottom: spike raster of input layer activity from two samples corresponding to two different classes. Middle: spike raster of excitatory (red) and inhibitory (blue) activity in the hidden layer. Top: membrane potential of readout units. The readout units corresponding to the two input classes are highlighted in different shades. (c) Performance comparison of Dalian and non-Dalian shallow SNNs. Left: validation accuracy after training on the SHD dataset as a function of initialization target σ_U. The shaded region around the lines indicates the range of values across five random seeds. The sand-colored shaded region corresponds to our suggested target fluctuation magnitude $1 \leqslant \xi \leqslant 3$. Right: test error of the five best-performing models in terms of validation accuracy, for Dalian and non-Dalian SNNs. Error bars mark $\pm$ one standard deviation. (d) As panel (c), for Dalian and non-Dalian three-layer CSNNs.

Taken together, these findings paint a clear pattern of initialization dependencies across datasets: up to a certain number of hidden layers, which is dataset dependent, Kaiming initialization yields good training performance in SNNs. However, when networks become too deep, vanishing SGs prevent training in networks with Kaiming initialization. In contrast, our proposed initialization strategy enables learning at high performance for deeper networks when the target fluctuation magnitude is set to $\sigma_U = 1$. As a complementary data-dependent strategy, homeostatic plasticity can be used to prevent vanishing gradients and rescue learning in deep networks that were initialized in a suboptimal regime.

2.3. Initializing SNNs that obey Dale's law

Neurons in biological SNNs are separated into excitatory and inhibitory populations, a constraint commonly known as Dale's law [33]. With added biological constraints, functional SNNs constitute an important *in silico* model system for computational neuroscience. To advance the development of biologically constrained SNNs, we extended our initialization theory to SNNs obeying Dale's law (see methods), i.e., in which each hidden layer consists of recurrently connected but separate excitatory and inhibitory populations (figure 7(a)). At initialization, we require a balance between excitatory and inhibitory currents ($\mu_U = 0$), as is commonly observed in biology [34, 35]. To accomplish such balance, we assume that excitatory and inhibitory synaptic weights are drawn from independent exponential distributions, whose mean values are set according to our theory to ensure the desired membrane potential dynamics (supplementary figure S11). This strategy allowed us to initialize Dalian networks with the same target σ_U as non-Dalian networks.

To test whether Dalian networks in the fluctuation-driven regime could be trained to high accuracy like their non-Dalian counterparts, we first considered fully connected recurrent Dalian SNNs with one hidden layer trained on the SHD dataset (see methods). Dalian networks initialized with $\sigma_U = 1$ accurately solved the SHD task after training for 200 epochs ($99.8\% \pm 0.0$ train & $82.2\% \pm 1.2$ test accuracy; figure 7(b)). Next, to test the robustness to initialization in Dalian networks, we initialized Dalian SNNs with different targets σ_U and trained them on the SHD dataset. For direct comparison between Dalian SNNs and non-Dalian SNNs, we constructed SNNs with a total of $n_\mathrm{h} = 160$ hidden layer neurons, which were further split into $n_\mathrm{exc} = 128$ and $n_\mathrm{inh} = 32$ neurons for the Dalian case (see table 4; methods). After training, the Dalian networks exhibited similar robustness to initialization as non-Dalian networks (figure 7(c)). While we did not observe a large difference between Dalian and non-Dalian networks in validation accuracy, Dalian networks exhibited higher accuracy on the SHD test dataset. This result suggests that the separation into excitatory and inhibitory populations could provide a functionally beneficial constraint for SNNs with one recurrently connected hidden layer trained on the SHD dataset.

We wondered whether the better generalization performance of shallow Dalian SNNs would extend to deeper CSNN network architectures. To address this question, we constructed Dalian CSNNs with three hidden

layers (see methods). Again, networks were initialized with different targets σ_U and trained on the SHD dataset. We found that Dalian CSNNs with three hidden layers were more sensitive to initialization than their non-Dalian counterparts (figure 7(d)). However, when successfully trained, Dalian and non-Dalian CSNNs resulted in similar test accuracies.

In summary, our initialization strategy extends to Dalian SNNs with different network architectures and enables robust training on the SHD dataset. Unexpectedly, constraining networks with Dale's law increased generalization accuracy by 7.1% in shallow networks. However, this effect did not generalize to deep CSNNs. Thus initializing Dalian networks in the fluctuation-driven regime is beneficial for their training and it will be interesting future work to study whether and how these findings generalize to larger datasets.

3. Discussion

We have introduced a general and easy-to-implement initialization strategy for SNNs and shown that it yields close-to-optimal training speed and classification performance across different SNN architectures and datasets. To that end, we developed a simple and general theory based on the notion of fluctuation-driven firing and tested it empirically in numerical simulations. We found that shallow SNN architectures are surprisingly robust to initialization with small synaptic weight magnitudes, whereas deep CSNNs require carefully chosen initial weight distributions that our theory accurately predicts. Further, our analysis showed that suboptimal initial weight choices result in vanishing or exploding SGs, similar to ANNs. Importantly, for all network architectures, including deep convolutional, recurrent, and Dalian SNNs, and the different datasets we considered, we found that fluctuation-driven initialization with given target membrane fluctuations of $\sigma_U = 1$, resulted in stable activity propagation and close-to-optimal learning performance. Based on our results, we recommend initializing SNNs in the fluctuation-driven regime using a target $\sigma_U = 1$ for all practical purposes. If activity propagation remains limited after training, a problem we observed in deeper network architectures, we recommend the addition of firing rate homeostasis to the training loss either for the entire training process or transiently during an initial priming period.

Functional SNNs are most commonly obtained by converting a previously trained ANN [36–40] or through direct training using timing-based methods [41–45] or SGs [5, 6, 46]. While both approaches can result in well-performing networks, direct training typically leads to sparser activity levels while also leveraging spike timing which can be beneficial for energy efficiency [47]. The initialization strategy developed in this article mainly applies to direct training approaches and specifically for SNNs trained with SGs.

Most previous SNN studies relied on weight initialization strategies that were established for ANNs in which they aim at keeping the variance of gradients constant through time or layers. For example, Xavier (Glorot) initialization [10] achieves stable variance in the backward pass by appropriately scaling the initial weight distribution. While the Xavier initialization was originally developed for linear networks, the Kaiming (He) initialization [11] extends this approach by explicitly taking into account the ReLU non-linearity, thereby enabling the training of deeper ReLU networks. While both Xavier and Kaiming initialization posit a scaling of weights by the number of input neurons as $\sim 1/n_{\text{in}}$, their profound effects on learning performance largely result from the different choice of the absolute weight scale, which differs by a factor of two, a direct consequence of neuronal non-linearity.

Alternatively, the weight scale in the case of SNNs is often determined empirically, however, there are some proposed initialization strategies, although they often lack a sound theoretical foundation. For example, Lee *et al* [14] proposed to normalize the magnitude of back-propagated errors across layers by initializing synaptic weights from a uniform distribution $W_{(l)} \sim \mathcal{U}[-\sqrt{3/n_l}, \sqrt{3/n_l}]$, where n_l is the number of incoming synapses. However, this approach was only validated in networks with two hidden layers trained on an event-based version of the MNIST dataset [48] and requires manual tuning of a per-layer weight scale to define the spiking threshold. Bellec *et al* [49] in turn initialized weights for spiking LSTM models from a normal distribution as $W_{(l)} \sim \mathcal{N}\left(0, 1/n_{l-1}\right)$, whereas Zenke and Vogels [5] used a uniform distribution $W_{(l)} \sim \mathcal{U}[-\sqrt{1/n_{l-1}}, \sqrt{1/n_{l-1}}]$. A more intricate approach was developed by Herranz-Celotti and Rouat [50], who suggested several conditions on the initial weights that aim, e.g., to balance the variance of the gradients across time and layers. Based on those conditions, the authors derived a way to determine the weight scale for a uniform distribution. While initialization with an ad hoc chosen weight scale can support successful training in shallow networks, none of these studies applied their initialization strategies to network architectures with more than two hidden layers. However, as shown in this article, shallow network architectures are intrinsically robust to initialization as long as the weights are small enough while the need for SNN-specific initialization mainly arises when training deep SNNs. It thus remains an open question whether these results generalize to deep SNNs.

Recently, Ding *et al* [51] proposed an initialization strategy that generalized to deep SNNs architectures. The authors related the magnitude of backpropagating gradients in feed-forward SNNs to the synaptic weight distribution and proposed a weight scale for normally distributed synaptic weights that takes into account some parameters of neuronal dynamics, but does not consider dataset-dependent input parameters. While similar to the approach outlined here, this initialization strategy is limited to centered weight distributions and feed-forward networks. In addition, this particular study limited the forward pass to 20 time steps and delta synapses, compared to 100–500 time steps and current-based synapses in our simulations. Using delta synapses and a smaller number of time steps can increase the performance of SNNs but does so at the cost of biologically realistic membrane potential dynamics. How well these results generalize to recurrently connected SNNs or more biologically plausible membrane dynamics is unclear.

Our fluctuation-driven initialization strategy follows a similar approach to Glorot and Bengio [10] and He *et al* [11] by setting a target variance for neuronal activity. However, due to the non-continuous nature of the spiking non-linearity, we formulated the goal in terms of the membrane potential variance σ_U instead of the post-non-linearity activation. Our theory results in weight scaling that not only accounts for the number of hidden layer neurons but also data- and architecture-dependent parameters.

In contrast to the above approaches, Mishkin and Matas [12] proposed an iterative initialization strategy to achieve unit variance of neuronal activations at each layer during a pre-training period. The implementation of a pre-training period is similar to the homeostatic priming period we applied here. However, instead of setting an explicit target for the population variance, our homeostatic regularizer tuned per-neuron spiking activity to enable activity propagation.

Our work has several limitations. First, our theory is limited to LIF neurons with current-based synapses. Although the current-based LIF is by far the most commonly used neuron model in SNNs, its synaptic dynamics can allow for biologically implausible and undesirable membrane potential values. Indeed, we found that some neurons exhibit exceptionally small ($u(t) \ll 0$) or large ($u(t) \gg \theta$) membrane potential values after training, which were not intended when designing the SGs. Future work could explore the possibility of using conductance-based synapses or additional regularization losses to constrain membrane potentials to a biologically plausible range while still allowing for large simulation time steps and thus rapid training.

Second, we performed numerical simulations with a relatively large time step of $\Delta t = 2$ ms. Choosing the simulation time step marks a trade-off between computational efficiency on one side and sensitivity of the membrane potential to quickly changing inputs on the other. Indeed, better performing deep SNNs have been trained using a time step on the order of the membrane potential time constant [51, 52]. Our choice of simulation time step reflects a compromise between minimizing computation time and allowing for sufficiently realistic membrane dynamics.

Third, our initialization theory for recurrent SNNs and SNNs following Dale's law, rests on the assumption of balanced input currents, i.e., $\mu_U = 0$, similar to what is observed in neurobiology [19, 20]. Whether and how this balanced state contributes to initial learning phases in the brain remains an open question for experimental and theoretical neuroscience. However, in our numerical simulations, sweeps across the parameters of initial weight distribution in shallow SNNs (figure 3) suggest that a slight dominance of inhibition over excitation may represent a similarly favorable or even more advantageous initial state for learning. Therefore, it equally remains to be clarified whether unbalanced currents, for example by a slight dominance of inhibition at initialization, could further support learning in functional SNNs models.

Fourth, our fluctuation-driven initialization theory makes several assumptions that could be violated in some use cases using real-world data. Our theory assumed that all input neurons are independent of each other and fire according to a homogeneous Poisson process with a common firing rate ν. Although we have shown that the systematic bias from violating this assumption in the Randman and SHD datasets is not too large (figure 2), other datasets with a different spatiotemporal structure could lead to destructive deviations from the theory. As a result, the current initialization strategy could be improved by taking into account more complex firing statistics of the input data. Additionally, our derivations neglected the spike reset of LIFs neurons. While mathematically more complex, it would be possible to consider the reset dynamics in our derivations using a Fokker–Plank approach [21]. However, given that the deviations from the theory due to spatiotemporal structure in the data likely outweigh the contribution of the spike reset, it is questionable whether this extension would confer an advantage.

Finally, we assumed equal firing rates $\nu = \nu_{\text{dataset}}$ for all neurons in a layer in deep SNNs, and for both excitatory and inhibitory populations in Dalian SNNs. Despite being violated for most initialization targets (cf figure 4), this simplification allowed for effective initialization with a common target $\sigma_U = 1$ across multiple datasets with vastly different average firing rates (see methods). Interestingly, for initialization with target $\sigma_U = 1$ on the SHD dataset, we indeed observed relatively constant firing rates across layers. A consistent method to estimate the firing rate distribution in deep layers at the time of initialization could improve the

Table 2. Dataset specifications after pre-processing.

	Randman	SHD	CIFAR-10	DVS-Gesture
Duration T_{dataset} (ms)	200	700	100	1000
Input dimensions	1	1	2	3
Input neurons n_{dataset}	20	700	32×32	$2 \times 32 \times 32$
Firing rate ν_{dataset} (Hz)	5	15.8	14.3	9.2
Classes	10	20	10	11
Total training samples	8000	7340	54 000	1077

performance of other initialization targets and could potentially enable training of deeper SNNs. As an alternative approach, dynamic initialization during a pre-training priming period could be extended to adjust weights by regularizing the output firing rate to a target value in an iterative fashion. Similar to approaches that have been proposed for ANNs [12], such an iterative and dynamic initialization strategy could enable activity propagation and learning in even deeper SNNs. However, increasing the number of layers in recurrently connected SNNs did not lead to significant performance improvements in our study. Given the success of deep ANNs, this suggests either that the datasets used to evaluate SNNs are too simple, or that deep SNN architectures and learning algorithms are still in their infancy and could be significantly improved.

In conclusion, the fluctuation-driven initialization proposed in this article facilitates training of diverse SNN architectures in neuromorphic engineering and computational neuroscience by striking a balance between seamless applicability and learning performance. Our work also adds further support to the idea that the fluctuation-driven firing regime, which is widely observed in the brain, may serve as an optimal initial state for future learning, and specifically for scenarios in which learning can be seen as an end-to-end optimization problem [53–55]. While our work only provides the first step toward more effective SNN initialization, it opens up several future exciting directions such as initialization in the presence of sparse connectivity or neuronal cell-type diversity, and suggests that we should take a deeper look at the role of homeostatic plasticity in dynamically preparing networks for optimal learning performance.

4. Methods

4.1. Learning tasks

We trained SNNs on several synthetic and real-world classification problems with increasing computational complexity and from different input modalities (auditory, static images, video) to test our initialization strategy. We chose one synthetic dataset and three real-world datasets covering different input modalities to generalize our results across different datasets. In the following, we briefly describe each dataset. The exact specifications of each dataset after preprocessing are summarized in table 2.

Synthetic random manifolds (Randman). We used a versatile synthetic classification dataset based on precise input spike timings drawn from smooth random manifolds as previously described ([5]; https://github.com/fzenke/randman). The approach allows for flexible dataset generation with different degrees of complexity by varying the number of classes, the intrinsic manifold dimension D, the smoothness parameter α, and the embedding space dimension M.

Here, we chose parameters to ensure the problem could not be solved by an ANN without a hidden layer. Specifically, we set the embedding space dimension $M = n_{\text{randman}} = 20$, $D = \alpha = 1$ and generated spike trains of 100 ms duration with 10 different classes for all our simulation experiments. To account for delays in activity propagation through the network, we appended 100 ms of no spiking activity to the generated inputs, resulting in a total duration of $T_{\text{randman}} = 200$ ms and hence an average input firing rate of $\nu_{\text{randman}} = 5$ Hz. Further, we used the same random seed to generate the dataset for all experiments in which we compare different initializations to avoid variability due to differences in the dataset. Specifically, we generated a 10-way classification dataset with 1000 samples for each class, 800 of which served as training data and two sets of 100 samples each served as validation and testing data, respectively.

SHD. The SHD dataset [23] is a real-world auditory dataset containing recordings of spoken digits (0–9) in both German and English from different speakers. It is freely available for download at https://ieee-dataport.org/open-access/heidelberg-spiking-datasets. To obtain input spikes, the raw audio data was pre-processed by a biologically inspired cochlear model [23] and mapped into an $n_{\text{SHD}} = 700$ dimensional input space. As individual input samples are of different duration, we considered only the first $T_{\text{SHD}} = 700$ ms of each sample, which corresponds to a fraction >98% of all input spikes. Spliced inputs were binned into $\frac{T_{\text{SHD}}}{\Delta t}$ time steps and fed directly into the SNNs. We used a random subset corresponding to 10% of the training data as a validation set. To evaluate generalization performance, we finally used the standard SHD test dataset which contains data from separate speakers that were not included in the training dataset.

CIFAR-10. The CIFAR-10 dataset consists of $3 \times 32 \times 32$ pixel images belonging to 10 different classes (6000 images for each class) and is commonly used as a visual classification dataset for neural networks [56]. The first dimension of the input data corresponds to the three RGB color channels. As an image dataset, it does not have an intrinsic time dimension in the input. To translate static images into temporal spiking input, we designed an additional sensory neuron encoding layer, placed in between the input layer and the first hidden layer, that converts static images into spike trains. First, each input pixel data was repeated with a fan-out factor of five along the channel dimension, thereby creating an effective input dimension of $15 \times 32 \times 32$ for each image. Second, each pixel value was multiplied by a gain factor to which a bias term was added before using the result as a current input to an encoding layer consisting of $15 \times 32 \times 32$ LIF units. The encoding weights for height and width dimensions were tied across all encoding units, leading to an encoding weight matrix of shape $15 \times 1 \times 1$. Thus, each encoding neuron receives the weighted pixel value of a single color channel as a constant synaptic current and transduces this value into an output spike train. Synaptic weights of the encoding layer were not subject to our initialization strategy, but randomly drawn from a normal distribution with mean 0 and standard deviation $\frac{\Delta t}{\tau_{\text{syn}}\sqrt{n_{l-1}}}$. Biases of encoding units were randomly drawn from a normal distribution with mean 0 and standard deviation $\frac{1}{\sqrt{n_{\text{enc}}}}$. Both encoding weights and biases were optimized end-to-end during training. For training, CIFAR-10 images were transformed to a normalized range $[-1, 1]$ and presented as input to the encoding layer for a duration of $T_{\text{CIFAR}-10} = 100$ ms. To obtain an estimate of the firing rate required for the initialization of hidden layers, we measured the average population firing rate of the encoding layer in response to the CIFAR-10 training dataset at the time of initialization, resulting in $\nu_{\text{CIFAR}-10} = 14.3$ Hz.

DVS128 Gesture dataset. The DVS-Gesture dataset [32] is a standard benchmark for event-based processing. It consists of 1342 videos of 11 different hand and/or arm gestures that were recorded with a biologically inspired dynamic vision sensor (DVS), yielding sparse and asynchronous input spike trains. The data from 23 recorded subjects serve as training data, while the data from six separate subjects serve as test data. Before training, we applied data augmentation and down-sampling, more specifically (1) random omission of events, (2) down-sampling of the original recordings, and (3) random temporal crop. First, recorded (binary) events were dropped with a probability of $p = 0.5$. Second, the original $2 \times 128 \times 128$ pixels recordings were down-sampled to $2 \times 32 \times 32$ pixels. Third, a random 1-second fragment was extracted from each sample. These 1-second long segments were then binned into $\frac{1000 \text{ ms}}{\Delta t}$ time steps and used as input to the SNN for $T_{\text{DVS-Gesture}} = 1000$ ms.

4.2. Network models

All SNN models were trained with SGs using PyTorch [57]. To this end, we used custom software written in Python 3.6.9, which is available on https://github.com/fmi-basel/stork. It includes the fluctuation-driven initialization methods discussed in this paper and example notebooks to replicate all main findings. For numerical simulations, all models were implemented in discrete time with a time step $\Delta t = 2$ ms. This time step was a compromise between numerical integration accuracy and computational and memory efficiency during training.

Neuron model. All units were implemented as simple LIF neurons with exponential current-based synapses [22]. In discrete time, the membrane potential of neuron i in layer l is characterized by the update equation

$$U_i^{(l)}[n+1] = \left(\lambda_{\text{mem}} U_i^{(l)}[n] + (1-\lambda_{\text{mem}}) I_i^{(l)}[n]\right)\left(1 - S_i^{(l)}[n]\right), \tag{10}$$

where $U_i^{(l)}[n]$ is this neuron's membrane potential at time step n and $S_i^{(l)}[n]$ is the associated binary (spiking) output of this neuron defined as $S_i^{(l)}[n] = \Theta\left(U_i^{(l)}[n] - \theta\right)$ with spike threshold θ, where Θ is the Heaviside step function. For simplicity, we set $\theta = 1$, so that the resting membrane potential is zero and the firing threshold is equal to one. The membrane decay variable λ_{mem} is determined by the membrane time constant τ_{mem} through $\lambda_{\text{mem}} \equiv \exp\left(-\frac{\Delta t}{\tau_{\text{mem}}}\right)$. Lastly, $I_i^{(l)}[n]$ denotes the incoming synaptic current to neuron i at time step n and is defined as

$$I_i^{(l)}[n+1] = \lambda_{\text{syn}} I_i^{(l)}[n] + \sum_j w_{ij}^{(l)} S_j^{(l-1)}[n] + \sum_j v_{ij}^{(l)} S_j^{(l)}[n] \tag{11}$$

with the feed-forward weight matrix W and optional recurrent weight matrix V. The synaptic decay variable λ_{syn} is related to the synaptic time constant through $\lambda_{\text{syn}} \equiv \exp\left(-\frac{\Delta t}{\tau_{\text{syn}}}\right)$. The neuronal parameters used throughout our simulations can be found in table 3. At the beginning of each mini-batch, all neurons were reset to their resting membrane potential of $U_i^{(l)}[0] = 0$ and a non-spiking state $S_i^{(l)}[0] = 0$.

Table 3. Neuronal parameters τ_{mem} and τ_{syn} used in the numerical simulations of SNNs.

	Non-Dalian SNNs	Dalian SNNs (exc./inh.)
τ_{mem} (ms)	20	20/20
τ_{syn} (ms)	10	10/20

Readout units. The units in the readout layer are identical to the above neuron model but were not allowed to spike. Additionally, the membrane time constant of readout units τ_{out} could be different from the hidden layer units. Unless otherwise mentioned, we set $\tau_{\text{out}} = T_{\text{data}}$ for all simulations to allow readout units to integrate inputs over the entire stimulus duration.

Dale's law. In SNNs obeying Dale's law (cf figure 7), each hidden layer consists of independent excitatory (E) and inhibitory (I) populations of LIF neurons with membrane time constants τ^E_{mem} and τ^I_{mem}, respectively. In discrete time, the membrane potential of each excitatory or inhibitory neuron i in layer l is identical to equation (10), where λ_{mem} is replaced with the population-specific decay variables λ^E_{mem} and λ^I_{mem}, respectively (cf table 3). Like in the non-Dalian case, the decay variables are related to the membrane time constants as $\lambda^E_{\text{mem}} \equiv \exp\left(-\frac{\Delta t}{\tau^E_{\text{mem}}}\right)$ and $\lambda^I_{\text{mem}} \equiv \exp\left(-\frac{\Delta t}{\tau^I_{\text{mem}}}\right)$. Contrary to the non-Dalian case, the input currents in Dalian SNNs consist of separate excitatory and inhibitory components originating from distinct presynaptic populations. For both excitatory and inhibitory populations, the input current can therefore be decomposed into feed-forward excitatory (F), recurrent excitatory (R), and recurrent inhibitory (I) components, such that

$$I_i^{(l),E}[n] = I_i^{(l),FE}[n] + I_i^{(l),RE}[n] - I_i^{(l),IE}[n] \tag{12}$$

$$I_i^{(l),I}[n] = I_i^{(l),FI}[n] + I_i^{(l),RI}[n] - I_i^{(l),II}[n]. \tag{13}$$

Thus, both the excitatory and inhibitory populations receive two sources of excitatory and one source of inhibitory input. In discrete time, the incoming synaptic currents to the excitatory neuron i of layer l are given as

$$I_i^{(l),FE}[n+1] = \lambda^E_{\text{syn}} I_i^{(l),FE}[n] + \sum_j w_{ij}^{(l),FE} S_j^{(l-1),E}[n] \tag{14}$$

$$I_i^{(l),RE}[n+1] = \lambda^E_{\text{syn}} I_i^{(l),RE}[n] + \sum_j w_{ij}^{(l),RE} S_j^{(l),E}[n] \tag{15}$$

$$I_i^{(l),IE}[n+1] = \lambda^I_{\text{syn}} I_i^{(l),IE}[n] + \sum_j w_{ij}^{(l),IE} S_j^{(l),I}[n], \tag{16}$$

where λ^E_{syn} and λ^I_{syn} are the decay variables of excitatory and inhibitory currents, which are related to their respective synaptic time constants τ^E_{syn} and τ^I_{syn} (cf table 3) as described before. Similarly, the synaptic currents into inhibitory neuron i of layer l are defined as

$$I_i^{(l),FI}[n+1] = \lambda^E_{\text{syn}} I_i^{(l),FI}[n] + \sum_j w_{ij}^{(l),FI} S_j^{(l-1),E}[n] \tag{17}$$

$$I_i^{(l),RI}[n+1] = \lambda^E_{\text{syn}} I_i^{(l),RI}[n] + \sum_j w_{ij}^{(l),RI} S_j^{(l),E}[n] \tag{18}$$

$$I_i^{(l),II}[n+1] = \lambda^I_{\text{syn}} I_i^{(l),II}[n] + \sum_j w_{ij}^{(l),II} S_j^{(l),I}[n]. \tag{19}$$

Together, the dynamics of each Dalian hidden layer are therefore determined by two feed-forward weight matrices W^{FE} and W^{FI} and four recurrent weight matrices W^{RE}, W^{RI}, W^{IE}, and W^{II}.

Connectivity. Feed-forward and recurrent networks were all-to-all connected without bias terms unless mentioned otherwise.

We used two types of convolutional networks with one-dimensional and two-dimensional convolutional kernels (cf tables 5 and 6). All CSNNs have recurrently connected layers unless mentioned otherwise. Recurrent connections in CSNNs were implemented as convolutions with filter kernels of size five and a stride of one. For weight initialization of CSNNs, we set $n = \text{fan}_{\text{in}}$, the number of inputs to each filter.

In SNNs and CSNNs obeying Dale's law, weights between consecutive layers and recurrent weights within hidden layers were constrained to be positive both at initialization and continuously during training, except for the readout weights, which were not sign constrained. Both excitatory and inhibitory populations in hidden layers received feed-forward inputs from the excitatory population of the previous layer. All networks obeying

Dale's law were fully recurrent, featuring recurrent connections within and between excitatory and inhibitory populations in each layer ($E \to E$, $E \to I$, $I \to I$, and $I \to E$).

Skip connections. We implemented skip connections as additional all-to-all connections with trainable weights between each except the last hidden layer and the readout layer, such that the readout units receive a separate input from every hidden layer (cf supplementary figure S10).

Supervised loss function. All networks were trained by minimizing a standard cross-entropy loss

$$\mathcal{L}_{\text{sup}} = -\frac{1}{K}\sum_{k=1}^{K}\sum_{c=1}^{C} y_c^k \log\left(p_c^k\right), \tag{20}$$

where the one-hot encoded target for input k is denoted by y_c^k, K is the number of input samples and C is the number of classes. The associated output probabilities p_c^k are given by the Softmax function

$$p_c^k = \frac{\exp(a_c^k)}{\sum_{i=1}^{C}\exp(a_i^k)}. \tag{21}$$

The scores a_c^k for each input k are dependent on the membrane potential of the associated readout units $U_c^{(\text{out})}$ and can take different forms. For all simulations in this paper, we defined the score as the maximum value over all time steps $a_c^k = \max_n\left(U_c^{(\text{out})}[n]\right)$.

Activity regularization. Unless otherwise mentioned, all networks were subject to activity regularization to constrain spiking activity. To that end, we added loss terms corresponding to a soft upper bound on the population-level spiking activity for each layer l as

$$g_{\text{upper}}^{(l),k} = \left(\left[\frac{1}{M^{(l)}}\sum_{i}^{M^{(l)}}\zeta_i^{(l),k} - v_{\text{upper}}\right]_+\right)^2, \tag{22}$$

where $\zeta_i^{(l),k} = \left(\sum_n^N S_i^{(l),k}[n]\right)$ is the spike count of neuron i in layer l given input sample k and $M^{(l)}$ is the number of neurons in hidden layer l. In one-dimensional CSNNs receiving auditory inputs, we set $M^{(l)} = n_{\text{features}}^{(l)} \times n_{\text{neurons}}^{(l)}$ and, similarly, in two-dimensional CSNNs receiving visual inputs, we set $M^{(l)} = n_{\text{features}}^{(l)} \times n_x^{(l)} \times n_y^{(l)}$ with $n_x^{(l)}$ and $n_y^{(l)}$ denoting the number of x and y coordinates in the layer, respectively. This activity regularization effectively prevents the population-level activity from exceeding the threshold spike count v_{upper}, which we set to $v_{\text{upper}} = \frac{T_{\text{data}}}{100}$ to achieve an upper bound average population firing rate of 10 Hz per layer. The regularization loss in case of population-level upper bound for spiking activity $\mathcal{L}_{\text{UB}}$ would thus be

$$\mathcal{L}_{\text{UB}} = -\lambda_{\text{upper}}\sum_{l}^{L} g_{\text{upper}}^{(l),k}, \tag{23}$$

where λ_{upper} denotes the strength of the regularization.

Homeostatic plasticity. In networks with homeostatic plasticity (cf figures 5 and 6) we added an additional term to the total loss acting as a per-neuron lower bound on the spiking activity. This per-neuron lower bound loss on spiking activity $\mathcal{L}_{\text{HP}}$ was defined as

$$g_{\text{lower}}^{(l),k} = \frac{1}{M^{(l)}}\sum_{i}^{M^{(l)}}\left(\left[-\left(\zeta_i^{(l),k} - v_{\text{lower}}\right)\right]_+\right)^2 \tag{24}$$

$$\mathcal{L}_{\text{HP}} = -\lambda_{\text{lower}}\sum_{l}^{L} g_{\text{lower}}^{(l),k}, \tag{25}$$

where the first equation describes the per-neuron loss term for each layer l. $\zeta_i^{(l),k}$ corresponds to the spike count of neuron i in layer l, $M^{(l)}$ is the number of neurons in layer l, v_{lower} denotes a lower bound on the spike count and λ_{lower} is the regularizer strength.

With $v_{\text{lower}} = 1$, this additional regularization term penalizes neurons that do not spike and thus ensures spiking activity in each neuron. By setting v_{lower} to other positive values one may achieve a desired lower bound on the per-neuron firing rate.

Surrogate Gradient (SG) descent. To minimize the loss $\mathcal{L}$, we adjusted network parameters in the direction of the negative SG. We computed SGs for the parameter updates using BPTT and the automatic differentiation capabilities of PyTorch [57]. Because the spiking non-linearity of the spiking neuron model is not

Table 4. Network and training parameters used for simulations of fully connected SNNs with a single hidden layer on the Randman and SHD datasets.

	Randman	SHD
No. input neurons	20	700
No. output neurons	10	20
No. hidden neurons	128	128 or 160
Dalian SNNs: No. hidden neurons	—	128 exc./32 inh.
Mini-batch size	400	400
No. training epochs	200	200

differentiable, we approximate its derivative

$$S'\left(U_i^{(l)}[n]\right) = \Theta'\left(U_i^{(l)}[n] - \theta\right) \tag{26}$$

with the surrogate

$$\tilde{S}'\left(U_i^{(l)}[n]\right) = h\left(U_i^{(l)}[n] - \theta\right). \tag{27}$$

Throughout this study, we use the SuperSpike surrogate non-linearity [46]

$$h(x) = \frac{1}{\left(\beta|x| + 1\right)^2} \tag{28}$$

with steepness parameter $\beta = 20$. For the simulations of deep CSNNs with a rescaled SG non-linearity reported in supplementary figure S8, we used the re-scaled surrogate derivative

$$\tilde{S}'\left(U_i^{(l)}[n]\right) = \frac{h\left(U_i^{(l)}[n] - \theta\right)}{h(\theta)}. \tag{29}$$

In this case, the surrogate derivative at rest is equal to one, i.e., $\tilde{S}'(0) = 1$, where 'at rest' refers to the absence of input to the corresponding neuron, causing its membrane potential to remain at zero. Thus, using this rescaled non-linearity, and in the absence of any membrane potential fluctuations, gradient magnitudes do not decay during backpropagation over the inactive layers.

Optimizer. We used the SMORMS3 optimizer [25] unless mentioned otherwise. Given a parameter θ, SMORMS3 performs the following update step after every mini-batch:

$$g_1^{(\theta)} := \left(1 - r^{(\theta)}\right) g_1^{(\theta)} + r^{(\theta)} \left(\frac{\partial \mathcal{L}}{\partial \theta}\right) \tag{30}$$

$$g_2^{(\theta)} := \left(1 - r^{(\theta)}\right) g_2^{(\theta)} + r^{(\theta)} \left(\frac{\partial \mathcal{L}}{\partial \theta}\right)^2 \tag{31}$$

$$m^{(\theta)} := 1 + m^{(\theta)} \left(1 - \frac{\left(g_1^{(\theta)}\right)^2}{g_2^{(\theta)} + \epsilon}\right), \tag{32}$$

where $r^{(\theta)} = \frac{1}{m^{(\theta)}+1}$ and $\epsilon = 1 \times 10^{-16}$ is a small positive value to avoid division by zero. Before the first training epoch, the optimizer state variables are initialized as $g_1^{(\theta)} = g_2^{(\theta)} = 0$ and $m^{(\theta)} = 1$. The parameter update after each mini-batch is then performed as

$$\Delta\theta = -\left(\frac{\partial \mathcal{L}}{\partial \theta}\right) \min\left(\eta, \frac{\left(g_1^{(\theta)}\right)^2}{g_2^{(\theta)} + \epsilon}\right) \frac{1}{\sqrt{g_2^{(\theta)}} + \epsilon}, \tag{33}$$

where η is the base learning rate.

Overview of network architectures. The diverse network architectures considered in this article were chosen as a compromise between memory requirements, performance, and training time. In the following, we provide a brief overview, while giving the precise parameter values in tables 4–6.

Fully connected shallow SNNs. In figure 3 and supplementary figures S2, S5 we implemented fully connected feed-forward SNNs with a single hidden layer. These networks were trained either on the Randman or the SHD dataset. For both tasks, we used 128 neurons in the hidden layer and adjusted the sizes of the input and readout layers to match the dataset requirements. For the fully connected recurrent SNNs used in figure 7 and

Table 5. Network and training parameters used for simulations of deep convolutional SNNs on the SHD dataset.

Dataset	SHD			
No. input neurons	700			
No. output neurons	20			
No. training epochs	200			
No. hidden layers	1	3	7	10
Mini-batch size	400	400	100	100
No. hidden neurons	16	16–32–64	16–32–64–...–64	16–32–64–...–64
Kernel size (ff)	21	21–7–7	7–5–...–5	5
Stride (ff)	10	10–3–3	3–2–...–2	2
Padding (ff)	2	2	2	2
No. parameters (ff)	24 858	24 656	99 952	157 520
Kernel size (rec)	5	5	5	5
Stride (rec)	1	1	1	1
No. parameters (rec)	52 734	51 536	208 752	327 760

Table 6. Network and training parameters used for simulations of deep convolutional SNNs on the CIFAR-10 and DVS-Gesture datasets.

Dataset	CIFAR-10		DVS-Gesture	
No. input neurons	32×32		$2 \times 32 \times 32$	
No. output neurons	10		11	
No. training epochs	50		20	
No. hidden layers	2	4	6	8
Mini-batch size	128	128	16	8
No. hidden neurons	32–32	32–32–64–64	32–32–64–64–128–128	32–32–64–64–128–...–128
Kernel size	3×3	3×3	3×3	3×3
Stride	1	1	1	1
Padding	2	2	2	2
No. parameters	95 456	109 792	308 800	586 816

supplementary figure S5, we added fully connected recurrent weights to the hidden layer. The recurrent SNNs in figure 7 had a wider hidden layer with 160 neurons, to match the number of hidden neurons in the Dalian network. The exact network specifications are summarized in table 4.

Fully connected shallow SNNs following Dale's law. Shallow SNNs following Dale's law (cf figure 7) had one hidden layer with 160 neurons, 128 of which were excitatory and 32 inhibitory, following a four-to-one ratio between excitatory and inhibitory neurons commonly observed in the mammalian cortex. Dalian networks were fully recurrently connected within the hidden layer through $I \rightarrow E$, $I \rightarrow I$, $E \rightarrow I$, and $E \rightarrow E$ connections. Feed-forward connections from the input to the hidden layer populations were constrained to be excitatory. Readout units received inputs solely from the excitatory population of the hidden layer, but readout weights were not subject to a sign constraint.

Deep feed-forward convolutional SNNs. Deep feed-forward CSNNs (cf figures 4, 6, supplementary figures S6 and S8), were implemented with up to ten consecutive hidden layers and trained on the SHD, CIFAR-10, or the DVS-Gesture datasets. CSNNs trained on the CIFAR-10 or DVS-Gesture datasets additionally implemented a max pooling operation with a kernel size of 2×2 after every second hidden layer. Network sizes and parameters of the convolutional operations are summarized in tables 5 and 6. These architectures were chosen to create networks of different depths with similar widths while ensuring that deeper layers still contained a reasonable number of neurons.

Deep recurrent convolutional SNNs. Unless mentioned otherwise, all CSNNs trained on the SHD dataset additionally implemented recurrent connections in each hidden layer (cf figures 4–7; supplementary figures S4, S7–S10). Recurrent connections in CSNNs were implemented as convolutional operations with a kernel size of five and a stride of one. The exact parameters of recurrent CSNNs trained on the SHD dataset can be found in table 5.

Deep recurrent CSNNs following Dale's law. CSNNs following Dale's law (cf figure 7) were implemented with separate excitatory and inhibitory populations in each hidden layer. Except for readout connections, which were not sign constrained, all feed-forward connections were constrained to be excitatory. As in the case of shallow SNNs following Dale's law, each hidden layer consisted of separate excitatory and inhibitory populations. Dalian CSNNs followed the same architecture as non-Dalian CSNNs (table 5) for excitatory neurons,

Table 7. Values of the PSP-kernel integrals $\bar{\epsilon}$ and $\hat{\epsilon}$ used for weight initialization in the numerical simulations, rounded to four decimal places. Due to the large simulation time step of $\Delta t = 2$ ms, $\bar{\epsilon}$ and $\hat{\epsilon}$ were obtained numerically. The analytical expressions for $\bar{\epsilon}$ and $\hat{\epsilon}$ can be found in supplementary table S1.

	Non-Dalian SNNs	Dalian SNNs (exc./inh.)
$\bar{\epsilon}$	0.0110	0.0110/0.0061
$\hat{\epsilon}$	0.0020	0.0020/0.0012

and each hidden layer was augmented with an additional population of inhibitory neurons of size $N_{\text{exc}}/4$. Excitatory $E \to I$ and $E \to E$ recurrent connections were implemented as convolutional operations with a kernel size of 5 and a stride of 1. Inhibitory $I \to I$ and $I \to E$ recurrent connections were implemented as convolutional operations with a kernel size of 3 and a stride of 1.

4.3. Weight initialization

For fluctuation-driven initialization of synaptic weight parameters, the PSP-kernel parameters $\bar{\epsilon}$ and $\hat{\epsilon}$ introduced in equations (2) and (3) can be computed analytically or numerically (see supplementary material S1). Because we used a relatively large time step of $\Delta t = 2$ ms for which there are non-negligible differences between the two, we used the numerical integration values for all simulations as they are closer to the actual simulation (table 7).

For strictly feed-forward networks, the fluctuation-driven initialization strategy was already covered in the main text. In the following, we derive the extensions to deep convolutional SNNs, recurrent SNNs, and SNNs obeying Dale's law.

Fluctuation-driven initialization of recurrent networks. For the initialization of recurrent layers, we introduce the additional parameter $0 < \alpha < 1$, that determines the proportion of membrane potential fluctuations caused by *feed-forward* connections in contrast to *recurrent* connections:

$$\alpha = \frac{\text{part of } \sigma_U^2 \text{ caused by feed-forward connections}}{\text{total } \sigma_U^2}. \tag{34}$$

To this end, we consider a postsynaptic LIF neuron receiving feed-forward input from n_F neurons with firing rate ν_F and recurrent input from n_R hidden layer neurons with average firing rate ν_R. Feed-forward weights are initialized as $W \sim \mathcal{N}(\mu_W, \sigma_W^2)$ and recurrent weights are initialized as $V \sim \mathcal{N}(\mu_V, \sigma_V^2)$ The mean μ_U and variance σ_U^2 of the membrane potential are then given by

$$\mu_U = n_F \mu_W \nu_F \bar{\epsilon} + n_R \mu_V \nu_R \bar{\epsilon} \tag{35}$$

$$\sigma_U^2 = n_F(\sigma_W^2 + \mu_W^2)\nu_F \hat{\epsilon} + n_R(\sigma_V^2 + \mu_V^2)\nu_R \hat{\epsilon}. \tag{36}$$

In practice the firing rate ν_R of the hidden layer is difficult to predict due to finite-size effects. Hence, we make the simplifying assumption $\nu = \nu_F = \nu_R = \nu_{\text{dataset}}$. In other words, we assume that the average firing rate of the hidden layers is equal to the input firing rate.

Since we want α to control the membrane potential *fluctuations* only, which are determined by σ_W^2 and σ_V^2, we can initialize recurrent and feed-forward weights with a common mean, i.e., $\mu_{WV} = \mu_W = \mu_V$ defined as

$$\mu_{WV} = \frac{\mu_U}{(n_F + n_R)\nu_{\text{dataset}}\bar{\epsilon}} \tag{37}$$

and subsequently solve for σ_W^2 and σ_V^2 independently:

$$\sigma_W^2 = \frac{\alpha}{n_F \nu \hat{\epsilon}} \left(\frac{\theta - \mu_U}{\xi} \right)^2 - \mu_{WV}^2 \tag{38}$$

$$\sigma_V^2 = \frac{1 - \alpha}{n_R \nu \hat{\epsilon}} \left(\frac{\theta - \mu_U}{\xi} \right)^2 - \mu_{WV}^2. \tag{39}$$

In this article, we used $\alpha = 0.9$ for all simulations with recurrently connected hidden layers unless stated otherwise, so that the majority of membrane potential fluctuations originate from feed-forward input.

Fluctuation-driven initialization of Dalian networks. Networks following Dale's law consist of separate excitatory and inhibitory populations whose output weights are sign constrained. To initialize the sign constrained connections, we relied on exponential or log–normal weight distributions instead of normally distributed weights, where the choice of a log–normal distribution is inspired by findings from neurobiology [58].

Parameterizing the excitatory and inhibitory weight distributions with λ for the exponential and μ for the log-normal distribution, respectively, allows us to obtain explicit expressions for the initial weight distributions

leading to the target membrane potential fluctuations with mean μ_U and variance σ_U^2. Unless stated otherwise, the weights in Dalian SNNs were initialized using the exponential distribution throughout the numerical simulations. While we provide here the expression for weight initialization using the exponential distribution, a derivation for log–normally distributed initial weights can be found in the supplementary material S3.

We start by observing that, regardless of the weight distribution from which synaptic weights are sampled, the mean and the variance of the membrane potential of a neuron i in a Dalian network are defined as

$$\mu_U^{(i)} = \sum_j^{n_E} w_{ij}^E \nu_E \bar{\epsilon}_E - \sum_k^{n_I} w_{ik}^I \nu_I \bar{\epsilon}_I \tag{40}$$

$$\left(\sigma_U^{(i)}\right)^2 = \sum_j^{n_E} (w_{ij}^E)^2 \nu_E \hat{\epsilon}_E + \sum_k^{n_I} (w_{ik}^I)^2 \nu_I \hat{\epsilon}_I, \tag{41}$$

where we assume equal firing rates ν_E and ν_I for all excitatory and inhibitory neurons in our experiments, respectively.

For weights drawn from exponential distributions, i.e., $w^E \sim \exp(\lambda_E)$ and $w^I \sim \exp(\lambda_I)$ with mean $\frac{1}{\lambda}$ and variance $\frac{1}{\lambda^2}$, we can rewrite the mean and the variance of the membrane potential for each neuron as

$$\mu_U = \frac{n_E \nu_E \bar{\epsilon}_E}{\lambda_E} - \frac{n_I \nu_I \bar{\epsilon}_I}{\lambda_I} \tag{42}$$

$$\begin{aligned}\sigma_U^2 &= n_E \left(\frac{1}{\lambda_E^2} + \left(\frac{1}{\lambda_E}\right)^2 \right) \nu_E \hat{\epsilon}_E + n_I \left(\frac{1}{\lambda_I^2} + \left(\frac{1}{\lambda_I}\right)^2 \right) \nu_I \hat{\epsilon}_I \\ &= \frac{2 n_E \nu_E \hat{\epsilon}_E}{\lambda_E^2} + \frac{2 n_I \nu_I \hat{\epsilon}_I}{\lambda_I^2}.\end{aligned} \tag{43}$$

We further assume that the target $\mu_U = 0$, as would be expected in balanced networks. From the definition of μ_U, we obtain an explicit relationship between λ_I and λ_E

$$\lambda_I = \lambda_E \frac{n_I \nu_I \bar{\epsilon}_I}{n_E \nu_E \bar{\epsilon}_E}, \tag{44}$$

which we use to define a combined E/I ratio based on network parameters

$$\Delta_{EI} = \frac{n_I \nu_I \bar{\epsilon}_I}{n_E \nu_E \bar{\epsilon}_E}. \tag{45}$$

Substitution of this relationship into equation (43) gives us

$$\sigma_U^2 = \frac{2 n_E \nu_E \hat{\epsilon}_E}{\lambda_E^2} + \frac{2 n_I \nu_I \hat{\epsilon}_I}{(\lambda_E \Delta_{EI})^2}. \tag{46}$$

Finally, we can solve the above for λ_E

$$\lambda_E = \frac{\sqrt{2(\Delta_{EI}^2 n_E \nu_E \hat{\epsilon}_E + n_I \nu_I \hat{\epsilon}_I)}}{\sigma_U \Delta_{EI}}. \tag{47}$$

Together, equations (44) and (47) allow us to parameterize excitatory and inhibitory weights as a function of σ_U, taking into account data- and network-dependent parameters, which is summarized in table 8. Note that this initialization relies on a target membrane potential mean $\mu_U = 0$.

Fluctuation-driven initialization of Dalian networks with excitatory recurrence. Dalian layers are always recurrently connected, as they require a connection between the separate excitatory and inhibitory populations in each layer. For this reason, the Dalian network from the above paragraph has recurrent inhibitory connections ($I \to E$ and $I \to I$). Here, we consider the case of additional excitatory recurrence, i.e. $E \to I$ and $E \to E$ connections.

Again, we require inhibitory currents to balance excitatory currents on average to achieve a mean membrane potential $\mu_U = 0$. Additionally, similar to non-Dalian SNNs with recurrent connections, the parameter α describes the proportion of excitatory membrane potential fluctuations that are caused by feed-forward excitatory connections, whereas the proportion of recurrent excitation is given by $(1 - \alpha)$. For the derivation,

Table 8. Summary of strategies for fluctuation-driven initialization of SNNs.

Network architecture	Weight distribution	Weight parameters	Good regime for initialization
Feed-forward	Centered: $W \sim \mathcal{N}(0, \sigma_W^2)$	$\sigma_W^2 = \frac{\sigma_U^2}{n\nu\hat{\epsilon}}$	$\frac{1}{3} \leqslant \sigma_U \leqslant 1$
	Non-centered: $W \sim \mathcal{N}(\mu_W, \sigma_W^2)$	$\mu_W = \frac{\mu_U}{n\nu\bar{\epsilon}}$	$\mu_U < \theta$
		$\sigma_W^2 = \frac{1}{n\nu\hat{\epsilon}}\left(\frac{\theta-\mu_U}{\xi}\right)^2 - \mu_W^2$	$1 \leqslant \xi \leqslant 3$
Recurrent	Centered: $W \sim \mathcal{N}(0, \sigma_W^2)$, $V \sim \mathcal{N}(0, \sigma_V^2)$	$\sigma_W^2 = \alpha\frac{\sigma_U^2}{n_F\nu\hat{\epsilon}}$	$\frac{1}{3} \leqslant \sigma_U \leqslant 1$
		$\sigma_V^2 = (1-\alpha)\frac{\sigma_U^2}{n_R\nu\hat{\epsilon}}$	$0 < \alpha < 1$
	Non-centered: $W \sim \mathcal{N}(\mu_{WV}, \sigma_W^2)$, $V \sim \mathcal{N}(\mu_{WV}, \sigma_V^2)$	$\mu_{WV} = \frac{\mu_U}{(n_F+n_R)\nu\bar{\epsilon}}$	$\mu_U < \theta$
		$\sigma_W^2 = \frac{\alpha}{n_F\nu\hat{\epsilon}}\left(\frac{\theta-\mu_U}{\xi}\right)^2 - \mu_{WV}^2$	$1 \leqslant \xi \leqslant 3$
		$\sigma_V^2 = \frac{1-\alpha}{n_R\nu\hat{\epsilon}}\left(\frac{\theta-\mu_U}{\xi}\right)^2 - \mu_{WV}^2$	$0 < \alpha < 1$

we consider a single neuron in a Dalian layer, receiving one recurrent inhibitory (I), one feed-forward excitatory (F), and one recurrent excitatory (R) input connection. In this setting, mean μ_U and variance σ_U^2 of the membrane potential of that neuron are given by

$$\mu_U = \frac{N_F\nu_F\bar{\epsilon}_E}{\lambda_F} + \frac{N_R\nu_R\bar{\epsilon}_E}{\lambda_R} - \frac{N_I\nu_I\bar{\epsilon}_I}{\lambda_I} \tag{48}$$

$$\sigma_U^2 = \frac{2N_F\nu_F\hat{\epsilon}_E}{\lambda_F^2} + \frac{2N_R\nu_R\hat{\epsilon}_E}{\lambda_R^2} + \frac{2N_I\nu_I\hat{\epsilon}_I}{\lambda_I^2}. \tag{49}$$

For the sake of simpler notation, we assume $\nu = \nu_F = \nu_R = \nu_I$ in this derivation. We also made this assumption of equal firing rates in the application of this initialization strategy in our numerical simulations. Since it is not possible to estimate the firing rates of excitatory and inhibitory hidden neuron populations in advance, we chose $\nu = \nu_{\text{dataset}}$.

The ratio of membrane potential fluctuations caused by the excitatory feed-forward connections compared to the total excitatory input, which we defined as α, can explicitly be written as

$$\alpha = \frac{\text{part of } \sigma_U^2 \text{ caused by excitatory feed-forward connections}}{\text{part of } \sigma_U^2 \text{ caused by all excitatory connections}} = \frac{\frac{2N_F\nu\hat{\epsilon}_E}{\lambda_F^2}}{\frac{2N_F\nu\hat{\epsilon}_E}{\lambda_F^2} + \frac{2N_R\nu\hat{\epsilon}_E}{\lambda_R^2}}, \tag{50}$$

which we can solve for λ_R to obtain

$$\lambda_R = \lambda_F\sqrt{\frac{\alpha N_R}{N_F - \alpha N_F}} = \lambda_F\Delta_R, \tag{51}$$

where we introduced the scalar Δ_R to make subsequent notation easier. We can then insert equation (51) into equation (48)

$$\mu_U = \frac{N_F\nu\bar{\epsilon}_E}{\lambda_F} + \frac{N_R\nu\bar{\epsilon}_E}{\lambda_F\Delta_R} - \frac{N_I\nu\bar{\epsilon}_I}{\lambda_I} \tag{52}$$

to receive an expression for λ_I

$$\lambda_I = \lambda_F\frac{\Delta_R\bar{\epsilon}_I N_I}{\Delta_R\bar{\epsilon}_E N_F + \bar{\epsilon}_E N_R} = \lambda_F\Delta_{EI}^R. \tag{53}$$

Table 9. Summary of strategies for fluctuation-driven initialization of Dalian SNNs.

Network architecture	Weight distribution	Weight parameters		Good regime for initialization
Feed-forward	$W_E \sim \exp(\lambda_E)$ $W_I \sim \exp(\lambda_I)$	$\lambda_E = \frac{\sqrt{2(\Delta_{EI}^2 n_E \nu_E \hat{\epsilon}_E + n_I \nu_I \hat{\epsilon}_I)}}{\sigma_U \Delta_{EI}}$ $\lambda_I = \Delta_{EI} \lambda_E$	$\Delta_{EI} = \frac{n_I \nu_I \bar{\epsilon}_I}{n_E \nu_E \bar{\epsilon}_E}$	$\mu_U = 0$ $\frac{1}{3} \leqslant \sigma_U \leqslant 1$
Recurrent	$W_F \sim \exp(\lambda_F)$ $W_R \sim \exp(\lambda_R)$ $W_I \sim \exp(\lambda_I)$	$\lambda_F = \frac{\sqrt{2\nu\left((\Delta_{EI}^R)^2 \hat{\epsilon}_E N_R + \Delta_R^2 \left(\Delta_{EI}^R N_F \hat{\epsilon}_E + N_I \hat{\epsilon}_I\right)\right)}}{\sigma_U \Delta_R \Delta_{EI}^R}$ $\lambda_R = \lambda_F \Delta_R$ $\lambda_I = \lambda_F \Delta_{EI}^R$	$\Delta_R = \sqrt{\frac{\alpha N_R}{N_F - \alpha N_F}}$ $\Delta_{EI}^R = \frac{\Delta_R \bar{\epsilon}_I N_I}{\Delta_R \bar{\epsilon}_E N_F + \bar{\epsilon}_E N_R}$	$\mu_U = 0$ $\frac{1}{3} \leqslant \sigma_U \leqslant 1$ $0 < \alpha < 1$

We introduce here again a network-parameter dependent scalar Δ_{EI}^R. Using the scalars Δ_R and Δ_{EI}^R, we can now substitute both λ_I and λ_R in equation (49) to obtain

$$\sigma_U^2 = \frac{2N_F \nu \hat{\epsilon}_E}{\lambda_F^2} + \frac{2N_R \nu \hat{\epsilon}_E}{(\lambda_F \Delta_R)^2} + \frac{2N_I \nu \hat{\epsilon}_I}{\left(\lambda_F \Delta_{EI}^R\right)^2}, \tag{54}$$

which can be solved for λ_F

$$\lambda_F = \frac{\sqrt{2\nu\left((\Delta_{EI}^R)^2 \hat{\epsilon}_E N_R + \Delta_R^2 \left(\Delta_{EI}^R N_F \hat{\epsilon}_E + N_I \hat{\epsilon}_I\right)\right)}}{\sigma_U \Delta_R \Delta_{EI}^R}. \tag{55}$$

Equations (51), (53) and (55) let us parameterize the initial feed-forward excitatory (F), recurrent excitatory (R), and recurrent inhibitory (I) weight distributions as a function of the target membrane potential fluctuations σ_U with a mean membrane potential $\mu_U = 0$. The suggested weight distributions including their parameters and a range of values for a good initialization are summarized in table 9.

Kaiming (He) initialization. We implemented Kaiming (He) initialization as described by He *et al* [11]. This commonly used strategy was originally derived for ANNs with ReLU nonlinearities and purports drawing the initial weights from a centered normal distribution

$$W \sim \mathcal{N}\left(0, \frac{2}{n}\right), \tag{56}$$

where n is the number of neurons and the weights have mean zero and variance $\sigma_W^2 = \frac{2}{n}$.

Acknowledgments

This work was supported by the Novartis Research Foundation and the Swiss National Science Foundation (Grant Number PCEFP3_202981).

Author contributions

FZ conceived the study. JR, JG, and FZ wrote simulation code. JR and JG performed simulations and analyses. JR, JG, and FZ wrote the manuscript.

Conflict of interest

The authors declare no competing interests.

Data availability statement

The simulation code that supports the findings of this study is openly available at the following URL: https://github.com/fmi-basel/stork.

ORCID iDs

Julian Rossbroich https://orcid.org/0000-0002-1927-8198
Julia Gygax https://orcid.org/0000-0001-7880-6694
Friedemann Zenke https://orcid.org/0000-0003-1883-644X

References

[1] Sterling P and Laughlin S 2017 *Principles of Neural Design* (Cambridge, MA: MIT Press)
[2] Indiveri G *et al* 2011 Neuromorphic silicon neural circuits *Front. Neurosci.* **5** 73
[3] Poole B, Lahiri S, Raghu M, Sohl-Dickstein J and Ganguli S 2016 Exponential expressivity in deep neural networks through transient chaos *Advances in Neural Information Processing Systems* vol 29 ed D Lee, M Sugiyama, U Luxburg, I Guyon and R Garnett (Red Hook, NY: Curran Associates)
[4] Hunsberger E and Eliasmith C 2015 Spiking deep networks with LIF neurons (arXiv:1510.08829 [cs.LG])
[5] Zenke F and Vogels T P 2021 The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks *Neural Comput.* **33** 899–925
[6] Neftci E O, Mostafa H and Zenke F 2019 Surrogate gradient learning in spiking neural networks: bringing the power of gradient-based optimization to spiking neural networks *IEEE Signal Process. Mag.* **36** 51–63
[7] Hochreiter S 1991 Untersuchungen zu dynamischen neuronalen Netzen *MA Thesis* Technische Universität, München
[8] Hochreiter S and Schmidhuber J 1997 Long short-term memory *Neural Comput.* **9** 1735–80
[9] Pascanu R, Mikolov T and Bengio Y 2013 On the difficulty of training recurrent neural networks *ICML*
[10] Glorot X and Bengio Y 2010 Understanding the difficulty of training deep feedforward neural networks *Proc. of the 13th Int. Conf. on Artificial Intelligence and Statistics Proceedings of Machine Learning Research, PMLR vol 9* ed Y W Teh and M Titterington pp 249–56
[11] He K, Zhang X, Ren S and Sun J 2015 Delving deep into rectifiers: surpassing humanlevel performance on ImageNet classification *2015 IEEE Int. Conf. on Computer Vision (ICCV)* (Santiago, Chile: IEEE) pp 1026–34
[12] Mishkin D and Matas J 2015 All you need is a good init (arXiv:1511.06422 [cs.LG])
[13] Srivastava R K, Greff K and Schmidhuber J 2015 Training very deep networks (arXiv:1507.06228 [cs.LG])
[14] Lee J H, Delbruck T and Pfeiffer M 2016 Training deep spiking neural networks using backpropagation *Front. Neurosci.* **10** 508
[15] Ledinauskas E, Ruseckas J, Juršėnas A and Buračas G 2020 Training deep spiking neural networks (arXiv:2006.04436 [cs.CV])
[16] Tiesinga P H, José J V and Sejnowski T J 2000 Comparison of current-driven and conductance-driven neocortical model neurons with Hodgkin–Huxley voltage-gated channels *Phys. Rev.* E **62** 8413–9
[17] Kuhn A, Aertsen A and Rotter S 2004 Neuronal integration of synaptic input in the fluctuation-driven regime *J. Neurosci.* **24** 2345–56
[18] Petersen P C and Berg R W 2016 Lognormal firing rate distribution reveals prominent fluctuation-driven regime in spinal motor networks *eLife* **5** e18805
[19] Vogels T P, Rajan K and Abbott L F 2005 Neural network dynamics *Annu. Rev. Neurosci.* **28** 357–76
[20] Brunel N 2000 Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons *J. Comput. Neurosci.* **8** 183–208
[21] Amit D J and Brunel N 1997 Model of global spontaneous activity and local structured activity during delay periods in the cerebral cortex *Cerebral Cortex* **7** 237–52
[22] Gerstner W, Kistler W M, Naud R and Paninski L 2014 *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition* (Cambridge: Cambridge University Press)
[23] Cramer B, Stradmann Y, Schemmel J and Zenke F 2022 The Heidelberg spiking data sets for the systematic evaluation of spiking neural networks *IEEE Trans. Neural Netw. Learn. Syst.* **33** 2744–57
[24] Yin B, Corradi F and Bohte S M 2021 Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks (arXiv:2103.12593)
[25] Funk S 2022 RMSprop loses to SMORMS3—beware the epsilon! https://sifter.org/simon/journal/20150420.html (Accessed 20 April 2015)
[26] Kingma D P and Ba J 2017 ADAM: a method for stochastic optimization (arXiv:1412.6980 [cs])
[27] Turrigiano G G and Nelson S B 2004 Homeostatic plasticity in the developing nervous system *Nat. Rev. Neurosci.* **5** 97–107
[28] Gjorgjieva J, Evers J F and Eglen S J 2016 Homeostatic activity-dependent tuning of recurrent networks for robust propagation of activity *J. Neurosci.* **36** 3722–34
[29] Zenke F and Gerstner W 2017 Hebbian plasticity requires compensatory processes on multiple timescales *Phil. Trans. R. Soc.* B **372** 20160259
[30] Srivastava R K, Greff K and Schmidhuber J 2015 Highway networks (arXiv:1505. 00387)
[31] He K, Zhang X, Ren S and Sun J 2016 Deep residual learning for image recognition *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition* pp 770–8
[32] Amir A *et al* 2017 A low power, fully event-based gesture recognition system *2017 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (IEEE) pp 7388–97
[33] Eccles J C, Fatt P and Koketsu K 1954 Cholinergic and inhibitory synapses in a pathway from motor-axon collaterals to motoneurones *J. Physiol.* **126** 524–62
[34] Rupprecht P and Friedrich R W 2018 Precise synaptic balance in the zebrafish homolog of olfactory cortex *Neuron* **100** 669–83
[35] Spiegel I, Mardinly A R, Gabel H W, Bazinet J E, Couch C H, Tzeng C P, Harmin D A and Greenberg M E 2014 Npas4 regulates excitatory–inhibitory balance within neural circuits through cell-type-specific gene programs *Cell* **157** 1216–29
[36] Esser S K, Appuswamy R, Merolla P, Arthur J V and Modha D S 2015 Backpropagation for energy-efficient neuromorphic computing *Advances in Neural Information Processing Systems* vol 28 Cortes C, Lawrence N, Lee D, Sugiyama M and Garnett R (Red Hook, NY: Curran Associates)
[37] Hunsberger E and Eliasmith C 2016 Training spiking deep networks for neuromorphic hardware (arXiv:1611.05141 [cs.LG])
[38] Cao Y, Chen Y and Khosla D 2015 Spiking deep convolutional neural networks for energy-efficient object recognition *Int. J. Comput. Vis.* **113** 54–66

[39] O'Connor P, Neil D, Liu S-C, Delbruck T and Pfeiffer M 2013 Real-time classification and sensor fusion with a spiking deep belief network *Front. Neurosci.* **7** 178
[40] Bu T, Ding J, Yu Z and Huang T 2022 Optimized potential initialization for low-latency spiking neural networks (arXiv:2202.01440 [cs.NE])
[41] Bohte S M, Kok J N and La Poutré H 2002 Error-backpropagation in temporally encoded networks of spiking neurons *Neurocomputing* **48** 17–37
[42] Booij O and tat Nguyen H 2005 A gradient descent rule for spiking neurons emitting multiple spikes *Inf. Process. Lett.* **95** 552–8
[43] Mostafa H 2018 Supervised learning based on temporal coding in spiking neural networks *IEEE Trans. Neural Netw. Learn. Syst.* **29** 3227–35
[44] Kheradpisheh S R and Masquelier T 2020 Temporal backpropagation for spiking neural networks with one spike per neuron *Int. J. Neural Syst.* **30** 2050027
[45] Comsa I M, Potempa K, Versari L, Fischbacher T, Gesmundo A and Alakuijala J 2020 Temporal coding in spiking neural networks with alpha synaptic function *2020 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE) pp 8529–33
[46] Zenke F and Ganguli S 2018 SuperSpike: supervised learning in multilayer spiking neural networks *Neural Comput.* **30** 1514–41
[47] Davidson S and Furber S B 2021 Comparison of artificial and spiking neural networks on digital hardware *Front. Neurosci.* **15** 651141
[48] Orchard G, Jayawant A, Cohen G K and Thakor N 2015 Converting static image datasets to spiking neuromorphic datasets using saccades *Front. Neurosci.* **9** 437
[49] Bellec G, Salaj D, Subramoney A, Legenstein R and Maass W 2018 Long short-term memory and learning-to-learn in networks of spiking neurons *Proc. of the 32nd Int. Conf. on Neural Information Processing Systems* (Red Hook, NY: Curran Associates) pp 795–805
[50] Herranz-Celotti L and Rouat J 2022 Surrogate gradients design (arXiv:2202.00282 [cs.AI])
[51] Ding J, Zhang J, Yu Z and Huang T 2022 Accelerating training of deep spiking neural networks with parameter initialization https://openreview.net/forum? id=T8BnDXDTcFZ
[52] Na B, Mok J, Park S, Lee D, Choe H and Yoon S 2022 AutoSNN: towards energy efficient spiking neural networks (arXiv:2201.12738 [cs.NE])
[53] Marblestone A H, Wayne G and Kording K P 2016 Toward an integration of deep learning and neuroscience *Front. Comput. Neurosci.* **10** 94
[54] Richards B A *et al* 2019 A deep learning framework for neuroscience *Nat. Neurosci.* **22** 1761–70
[55] Lillicrap T P, Santoro A, Marris L, Akerman C J and Hinton G 2020 Backpropagation and the brain *Nat. Rev. Neurosci.* **21** 335–46
[56] Krizhevsky A 2009 Learning multiple layers of features from tiny images *Tech. Report*
[57] Paszke A *et al* 2019 PyTorch: an imperative style, high-performance deep learning library *Advances in Neural Information Processing Systems* vol 32 ed H Wallach, H Larochelle, A Beygelzimer, F d'Alché-Buc, E Fox and , R Garnett (Red Hook, NY: Curran Associates) pp 8024–35
[58] Buzsáki G and Mizuseki K 2014 The log-dynamic brain: how skewed distributions affect network operations *Nat. Rev. Neurosci.* **15** 264–78