Causal pieces: analysing and improving spiking neural networks piece by piece

Dominik Dold, Philipp Christian Petersen
Faculty of Mathematics and Research Network Data Science @ Uni Vienna
University of Vienna
Kolingasse 14-16, 1090 Vienna, Austria
dominik.dold@univie.ac.at

Abstract

We introduce *causal pieces*, a novel concept for analysing spiking neural networks (SNNs), inspired by *linear pieces* used to study expressivity and trainability in artificial neural networks (ANNs). Causal pieces partition the input and parameter space of a feedforward SNN with single-spike coding into distinct regions where the same subnetwork causes the output spikes. For networks of current-based leaky integrate-and-fire (LIF) neurons with large membrane time constants, we show that within each causal piece, output spike times are locally Lipschitz continuous with respect to inputs and network parameters. We further prove a lower bound on the approximation error that depends on the number of causal pieces. Thus, the number of causal pieces is a measure of the approximation capabilities of SNNs, which is valid despite spike-time discontinuities and applies to networks with both excitatory and inhibitory synapses. Empirically, we find that parameter initialisations yielding more causal pieces on the training set strongly correlate with SNN training success across multiple benchmarks, including Yin-Yang, Fashion-MNIST, and EuroSAT. Moreover, simulations with standard single-spike LIF neurons indicate that our findings extend beyond the theoretically analysed regime. These results establish causal pieces as a powerful and principled tool for analysing and improving the computational capabilities of SNNs.

1 Introduction

Spiking neural networks (SNNs) have recently received increased attention due to their ability to facilitate low-power hardware solutions for deep learning methods, particularly for edge applications, e.g., in outer space onboard spacecraft [1–3]. In large parts, this is caused by the development of methods and software tools that allow the use of error backpropagation to train SNNs [4–8], as well as emerging spike-based hardware systems [9] such as Intel's digital Loihi [10, 11] and the analog BrainScaleS-2 [12, 13] chip. Despite these developments, the computational capabilities of SNNs remain poorly understood [14, 15], and it is still an open question whether spike-based models provide advantages compared to non-spiking artificial neural networks (ANNs) [3, 16–19].

Existing theoretical analyses of SNNs are often limited in scope. Early theoretical results based on VC dimension are limited to binary classification tasks and abstract spiking neuron models [20–22]. More recent approaches introduce additional assumptions to increase analytical tractability and avoid spike-time discontinuities, such as single-spike coding, positive weights, feedforward architectures, and simplified membrane and synapse dynamics (e.g. linearly rising postsynaptic potentials) [14, 15, 23]. Generally, the fact that output spike times can change discontinuously or vanish due to continuous changes in inputs or parameters has so far prevented a rigorous analysis of SNN expressivity.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

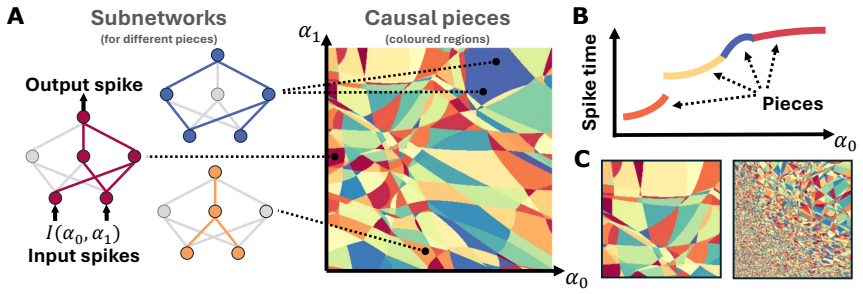

Figure 1: Causal pieces. **(A)** Every coloured region is a causal piece, fully partitioning the input space (here: 2D inputs). Within each piece, the same subnetwork causes the SNN output (left, coloured like the corresponding piece). Here we scan a 2D hyperplane I in the 40-dimensional input space, parametrised by $(\alpha_0, \alpha_1) \in [0, 1]^2$. **(B)** On each piece (coloured), the output spike time of the SNN is a continuous function. **(C)** Causal pieces for an SNN with no (left) and one (right) hidden layer.

To approach this challenge, we introduce a framework for characterising SNNs inspired by *linear pieces*, a concept used to analyse the expressivity of feedforward ReLU-based ANNs [24–27]. Our main contribution is the idea of *causal pieces*: a structured way to partition the input and parameter space of a feedforward SNN with single-spike coding into regions where output spikes are caused by the same subnetwork. Causal pieces naturally account for discontinuities in spike timing, can be evaluated for commonly used spiking neuron models, and do not require restrictions on synapse types. While piecewise decompositions in discrete-time spiking models [28] have recently been introduced, these differ fundamentally from our approach: their regions correspond to constant function outputs, and their number does not increase with neural network depth.

Fig. 1 illustrates our concept. Each causal piece corresponds to a region of the inputs and network parameters in which the same subset of neurons and synapses caused the output spikes (Fig. 1A). Within a piece, the output spike times vary smoothly, while transitions between pieces can induce discontinuities (Fig. 1B). Although the number of pieces is determined by the network parameters, it can be vastly increased by adding more layers (Fig. 1C, shown architectures: $[40, 1]$, $[40, 40, 1]$).

This work establishes causal pieces as a tool for analysing feedforward SNNs both theoretically and empirically. We show that the number of causal pieces provides a measure of expressivity by deriving a lower bound on the approximation error for single-spike, current-based leaky integrate-and-fire (LIF) neurons with large membrane time constant (Theorem 1). Furthermore, we demonstrate that the number of causal pieces the training data fall into at network initialisation correlates with training success (Figs. 3 and 4), providing a grounded metric to guide SNN initialisation currently missing in the literature [29]. Notably, even SNNs with only positive weights can realise a large number of causal pieces and achieve competitive performance compared to fully connected ANNs. While our theoretical results are derived for a specific class of LIF neurons, simulations with standard single-spike LIF neurons indicate that key results may extend to more complex neuron models. Moreover, our derived bounds generalise to multi-spike neurons via a recently introduced transference principle [30], which also provides a way of extending the definition of causal pieces to the multi-spike setting.

In the following, we briefly introduce the preliminaries required to follow this study before providing theoretical and experimental results. Mathematical proofs, simulation details, and algorithms can be found in the Appendix. Code is publicly available on GitHub [31].

2 Methods

In this work, we focus on the widely used current-based leaky integrate-and-fire neuron model in the limit of very large membrane time constants τ_{m} (see Definition 1), which is also known as the nLIF model [5, 6]. While this type of neuron model features no leakage, input spikes are still integrated with a finite synaptic time constant τ_{s}. In addition, we assume a widely used purely time-dependent encoding scheme in which each neuron spikes at most once [6–8, 23, 32, 33], which is also known as phase, latency, or time-to-first-spike coding. This setup is motivated by three considerations: **(i)** It sets the focus on purely temporal spike coding, which has increasingly been used for machine learning applications [3, 6, 23, 32]. **(ii)** The nLIF model is analytically tractable, enabling provable

theoretical results. **(iii)** As a special case of the widely used LIF model (see Section A.2.2), it provides a principled starting point for extending our results in future work. We demonstrate this by providing simulations for single-spike LIF neurons with finite membrane time constant ($\tau_{\mathrm{m}} = 2\tau_{\mathrm{s}}$). In general, causal pieces are computed similarly for standard LIF and other more complex neuron models.

The definition of a feedforward nLIF neural network is given in Definition 1.

Definition 1 (nLIF) *Let $L \in \mathbb{N}$, $\ell \in [1, L]$, $N_\ell \in \mathbb{N}$ be the number of neurons per layer ℓ, $\tau_{\mathrm{s}} \in \mathbb{R}^+$ be the synaptic time constant, $\vartheta \in \mathbb{R}$ be the threshold, and $t^{(0)} \in \mathbb{R}^{N_0}$, $N_0 \in \mathbb{N}$, be the inputs to the neural network. For $i \in [1, N_\ell]$, $j \in [1, N_{\ell-1}]$, let $W_{ij}^{(\ell)} \in \mathbb{R}$ be the synaptic weights from layer $\ell - 1$ to ℓ. Then the membrane potential $u_i^{(\ell)} \in \mathbb{R}$ of a neuron i in layer ℓ at time $t \in \mathbb{R}$ is given by:*

$$u_i^{(\ell)}(t) = \sum_{t_j^{(\ell-1)} \leq t} W_{ij}^{(\ell)} \left[1 - \exp\left(-\frac{t - t_j^{(\ell-1)}}{\tau_{\mathrm{s}}} \right) \right] . \tag{1}$$

The spike time $t_i^{(\ell)}$ of a neuron i in layer ℓ is defined as $t_i^{(\ell)} = \inf\{t : u_i^{(\ell)}(t) = \vartheta\}$.

The output spike time of an nLIF neuron (and LIF with $\tau_{\mathrm{m}} = 2\tau_{\mathrm{s}}$) can be calculated analytically by finding, given a set of input spike times and weights, the corresponding *causal set* (see Section A.2.1) [6]. The causal set contains the indices of all pre-synaptic neurons that cause the output spike time, i.e., it is the sorted set of neurons whose input spikes occur before the output spike (Definition 2).

Definition 2 (Causal set) *Let $t_i^{(\ell)} \in \mathbb{R} \cup \{\infty\}$ be the spike time of a neuron receiving $N_{\ell-1} \in \mathbb{N}$ input spikes at times $t_j^{(\ell-1)}$ for $j \in [1, N_{\ell-1}]$. Then the corresponding causal set is given by $\mathcal{C}_i^{(\ell)}(t_1^{(\ell-1)}, ..., t_{N_{\ell-1}}^{(\ell-1)}) = (j_1, ..., j_k)$ with $\{j_1, ..., j_k\} = \{j : t_j^{(\ell-1)} < t_i^{(\ell)}\}$ and $j_1 \leq ... \leq j_k$ if $t_i^{(\ell)} < \infty$ and $\mathcal{C}_i^{(\ell)}(t_1^{(\ell-1)}, ..., t_{N_{\ell-1}}^{(\ell-1)}) = ()$ otherwise.*

Causal sets implicitly depend on the neural network weights through $t_i^{(\ell)}$. For deep feedforward SNNs, the concept is generalised by introducing *causal subnetworks* (Definition 3).

Definition 3 (Causal subnetwork) *Let $L \in \mathbb{N}$, $\ell \in [1, L]$, $N_\ell \in \mathbb{N}$, $N_0 \in \mathbb{N}$. Further, let $\mathcal{C}_i^{(m)}$ be the causal set of neuron $i \in [1, N_m]$ in layer $m \in [1, L]$. Then for a subset $I \subseteq [1, N_\ell]$ of neurons in layer ℓ, the causal subnetwork $\mathcal{P}_I^{(\ell)}(t^{(0)})$ given inputs $t^{(0)} \in \mathbb{R}^{N_0}$ is defined recursively:*

$$\mathcal{P}_{I,n-1}^{(\ell)} = \left(\mathcal{C}_j^{(n-1)} : j \in \mathcal{C} \text{ for } \mathcal{C} \in \mathcal{P}_{I,n}^{(\ell)} \right) , \tag{2}$$

with $\mathcal{P}_{I,\ell}^{(\ell)} = \left(\mathcal{C}_i^{(\ell)} : i \in I \right)$ and $n \in [1, \ell]$.

As depicted in Fig. 1A, a causal subnetwork refers to the subset of neurons and connections that influenced the output spike times of neurons $i \in I$ of layer ℓ, given inputs $t^{(0)}$. In this work, we represent it as a list of lists: for each layer, we include a list containing the causal sets of neurons in this layer that contributed to the spike times of neurons in I. It can be calculated from the observed spike times and connectome alone (see the algorithm in Section A.5.9), but depends implicitly on the weights. An illustration of causal sets and causal subnetworks can be found in Section A.1.1.

Using these definitions, we can now introduce the concept of causal pieces, which we later demonstrate to be a useful tool for analysing the computational properties of SNNs. For a subset I of neurons in layer ℓ of a feedforward SNN, the causal piece is a region in the joint input and parameter space for which the causal subnetwork remains fixed, meaning that the spike times of neurons $i \in I$ depend on the same subnetwork (Fig. 1A) within this region. Formally, using Definitions 2 and 3 we define a causal piece as follows (Definition 4):

Definition 4 (Causal piece) *Let $L \in \mathbb{N}$, $\ell \in [1, L]$, $N_\ell \in \mathbb{N}$, $t_0 \in \mathbb{R}^{N_0}$ be the input spike times to the SNN with $N_0 \in \mathbb{N}$, and weights $W \in \mathbb{W} = \mathbb{R}^{N_0 \cdot N_1} \times ... \times \mathbb{R}^{N_{L-1} \cdot N_L}$. Then for a subset $I \subseteq [1, N_\ell]$ of neurons from layer $\ell \in [1, L]$, we call $\mathbb{P}[\mathcal{P}_I^{(\ell)}]$ the causal piece associated to $\mathcal{P}_I^{(\ell)}$:*

$$\mathbb{P}[\mathcal{P}_I^{(\ell)}] = \{(t_0, W) \in \mathbb{R}^{N_0} \times \mathbb{W} : \text{ given } t_0 \text{ \& } W, \text{ the neurons } i \in I \text{ have causal subnetwork } \mathcal{P}_I^{(\ell)}\}$$

Throughout this paper, we often consider causal pieces for networks with fixed weights. In such cases, the causal piece is only defined by the inputs and reduces to $\mathbb{P}[\mathcal{P}_I^{(\ell)}] \subseteq \mathbb{R}^{N_0}$.

Under the perspective of Definition 4, the output spike times of an SNN are piecewise continuous, non-linear functions (Fig. 1B) of the inputs, with each region corresponding to a distinct causal piece. For nLIF networks, within each piece, the output spike times are Lipschitz continuous with respect to both the input spike times and weights – a useful property for gradient-based training. When transitioning between causal pieces – for example, by varying the input to an nLIF neuron – the output spike time may change continuously, discontinuously, or become undefined (i.e., the neuron ceases to spike), depending on how the causal sets of neurons change (see Section A.3.1 for details).

3 Results

In the following, we first prove that the number of causal pieces provides a lower bound for the approximation error of a feedforward nLIF SNN. We then continue by demonstrating how to count them. The theoretical results are complemented by simulations, showing, in particular, that a high number of causal pieces on the training samples at initialisation correlates with training success (Fig. 3).

3.1 The number of causal pieces is a measure of expressivity

The approximation error of an nLIF network, i.e., how well a given function can be approximated, is lower bounded by an expression depending inversely on the number of causal pieces – meaning that more causal pieces result in potentially more expressive SNNs (Theorem 1; for a proof, see Section A.4.3). Theorem 1 proves a lower bound on the approximation error on an interval $[a, b]$. For a similar lower bound on a high-dimensional domain, it is enough to exhibit one subset of the domain on which the error is already large, since for every line $s \subset [a, b]^d$ it holds that $\|\Phi - g\|_{L^\infty([a,b]^d)} \geq \|\Phi - g\|_{L^\infty(s)}$. Most importantly, this result is valid even if Φ has any discontinuous behaviour.

Theorem 1 (Approximation bound) *Let $-\infty < a < b < \infty$, $g \in C^3([a, b])$ so that $\exp(g/\tau_\mathrm{s})$ is not affine. Then there exists a constant $c > 0$ that only depends on $\tau_\mathrm{s} \int_a^b \sqrt{|\frac{\mathrm{d}^2}{\mathrm{d}x^2} e^{g(x)/\,\tau_\mathrm{s}}|}\mathrm{d}x$ and a constant $\zeta > 0$ only depending on the maximum of $\max_x \left(e^{\Phi(x)/\,\tau_\mathrm{s}}\right)$ and $\max_x \left(e^{g(x)/\,\tau_\mathrm{s}}\right)$ so that*

$$\|\Phi - g\|_{L^\infty([a,b])} > \frac{c}{\zeta} p^{-2} \tag{3}$$

for all feedforward nLIF neural networks Φ (Definition 1) with p causal pieces and synaptic time constant τ_s.

This bound is also valid for nLIF neurons that spike multiple times (with a simple reset mechanism) due to a recently proven transference principle [30]. Through this principle, an implied notion of causal pieces can be defined for multi-spike SNNs by constructing equivalent single-spike SNNs.

3.2 Estimating the total number of causal pieces

Since the number of causal pieces is a measure of the expressivity of nLIF neural networks, it is of substantial interest to estimate this number. As every causal piece is characterised by a unique causal subnetwork, one way is to calculate the total number of causal subnetworks that can be formed. For a single nLIF neuron with N total inputs, a naive upper bound for the number of causal pieces is therefore $2^N - 1$ (the number of non-empty subsets that can be formed from a set of N elements).

However, not all of these subsets will be valid causal sets. In case of nLIF neurons, this means that the sum of the respective input weights might not exceed the threshold. We obtain an improved upper bound by calculating the probability that, given weights sampled from a static random distribution q, the sum of k weights exceeds the threshold, denoted by p_k^q. This is equivalent to the probability of a discrete random walk with continuous random step sizes (i.e. the weights) being above the threshold at step k (Fig. 2A). This criterion is sufficient, as we can freely choose the inputs: all inputs of neurons in the causal set spike at the same time, while neurons not part of the set spike after the output neuron (Section A.4.4). The number of causal pieces η^q is then upper bounded by $\eta^q \leq \sum_{k=1}^N \binom{N}{k} p_k^q$.

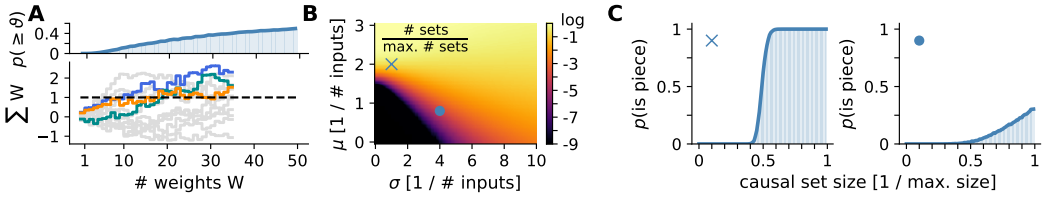

Figure 2: Estimating the number of causal pieces. **(A)** The probabilities p_k^q are obtained by counting how many trajectories (cumulative sum of weights) are above the threshold at step k. The top panel shows the corresponding values of p_k^q, where k is the number of weights. **(B)** Estimated number of pieces for weights sampled from normal distribution with different mean (y-axis) and standard deviation (x-axis). Colours are shown in log-scale. **(C)** p_k^q for two points in (B), denoted by markers.

We show the improved upper bound of the number of sets as a fraction of $2^N - 1$ in Fig. 2B for weights randomly initialised from Gaussian distributions with different mean and variance (using a Monte Carlo approach, see Algorithm 1 in the appendix). For illustration purposes, p_k^q is shown for two different q in Fig. 2C. The obtained results highlight two points: **(i)** the highest number of causal pieces is reached only for distributions with non-zero mean – which is quite remarkable given that initialisation schemes in the literature, often borrowed from traditional deep learning, sample the weights from distributions with zero mean [29, 33–37]. However, for extreme distributions, e.g., with very high mean, the estimated number of pieces is only achieved for data distributions that are very different from those found in practice, which will be discussed more thoroughly in the next section. **(ii)** With increasing variance, results tend to improve even if the mean is set non-optimally. In fact, one can show that in the limit of large variance, the number of pieces is lower bounded by an expression proportional to $N^{-3/2}$ (Theorem 2):

Theorem 2 (Number of pieces in limit) *Let q be a symmetric and continuous probability distribution with mean $\mu < \infty$ and variance σ^2, and $W_j \sim q$ for $0 \leq j < N$. In the limit $\frac{\mu}{\sigma} \to 0$ and $\frac{\vartheta}{\sigma} \to 0$, the number of causal pieces is lower bounded by*

$$\eta^q \geq \frac{2^N - 1}{2N\sqrt{\pi \cdot (N - \frac{2}{3})}}\,, \tag{4}$$

which is, quite remarkably, valid for all such probability distributions. This is a direct consequence of the Sparre Andersen theorem for random walks [38, 39], see Section A.4.5. While this result is only valid when all weights follow the same distribution, this can be guaranteed at initialisation, explaining the observed growth in the number of causal pieces with increased variance (Fig. 2B).

For deep SNNs, the number of causal pieces corresponds to the number of distinct routes through which spikes can propagate from the inputs to the outputs (Definition 3). For nLIF networks with $\{N_1, ..., N_\ell, 1\}$ neurons per layer, we find in Section A.4.6 that a naive upper bound for the number of pieces of the output neuron is $\eta^q \leq 2^{\prod_{i=1}^{\ell} N_i} \leq 2^{N^\ell}$, where $N = \max\{N_1, ..., N_\ell, 1\}$. This differs from ReLU ANNs, for which linear pieces are determined by neuron activation patterns rather than routes, with known upper bounds for the number of pieces scaling exponentially in depth [25, 26] (or the total number of neurons [27]). However, it remains to be seen whether SNNs with such a large number of pieces can be constructed, although Fig. 1C suggests quite dramatic increases in the number of causal pieces by adding even a single hidden layer. Since for general LIF neurons, weights also have to sum above threshold, similar estimates for the expected number of pieces apply.

3.3 The practically relevant number of causal pieces

In practice, even for single neurons we expect the number of pieces to be below the improved bound we found, as most of these pieces will not be traversed when given realistic input data (i.e., not all inputs being identical). Moreover, the total number of pieces may be irrelevant for the learning problem at hand if a large fraction of the pieces occupy parts of the domain that are not populated by data. For example, Fig. 1C shows that the density of pieces can change dramatically throughout the domain. Thus, we propose an alternative approach to counting causal pieces which is more aligned

to practical scenarios and less resource demanding: given a dataset, we count only the number of pieces that contain at least one data point. In the following, we only use this approach to count the number of causal pieces. An algorithm for counting causal pieces is provided in Section A.5.9. Using this approach, the complexity of calculating causal pieces scales linearly with the number of neural network parameters and training samples (see Section A.1.2 for simulation results on compute time for a single data sample) – although this can be optimised by parallelising the calculation per layer.

3.4 The number of practically relevant causal pieces correlates with training success

The initialisation scheme of parameters is crucial for training both ANNs and SNNs. Although for SNNs, schemes derived experimentally or adopted from ANNs have been successfully applied, a recent study highlighted the lack of a principled approach for identifying initialisation schemes that facilitate the training of SNNs [29]. As a first application, we demonstrate that the number of causal pieces at initialisation, evaluated only using training samples, is a strong predictor of training success. This is motivated by Theorem 1, which provides a lower bound of the approximation error, meaning that no nLIF SNN with p pieces can reach an error lower than this bound. Thus, when an SNN has only few pieces at initialisation, this bound is high, and lower errors are initially not reachable. In contrast, if the number of pieces is high at initialisation, this particular approximation-theoretic obstruction is reduced; whether optimization actually reaches a low-error solution remains dependent on the training dynamics.

We trained 200 shallow nLIF networks with $[4, 30, 3]$ neurons, where the last layer is a linear readout layer ("affine decoder" [15]) that ignores neurons that do not spike, on the Yin-Yang dataset [40] (with 5000 training samples). Training was performed using exact error backpropagation on analytically calculated spike times (via standard autograd in pyTorch), as introduced in [5, 6] – although we do not use any weight regularisation like "weight-bumping" [5, 6, 32]. Inputs were presented to the network as single spike times. To guarantee networks with a large variety of causal pieces after initialisation, we sampled weights from a normal distribution with randomly sampled mean and variance (see Section A.5). As shown in Fig. 3A,B, both the number of causal pieces of the last layer before and after training (evaluated using only training samples) strongly correlate with the

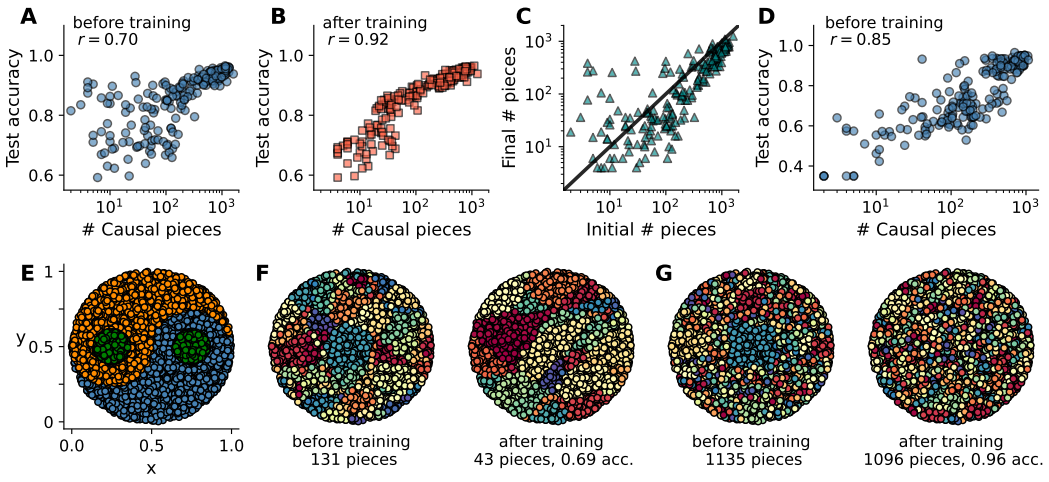

Figure 3: Network initialisation strongly affects training success. **(A)** The logarithm of the number of pieces (here: of the output layer) at network initialisation strongly correlates with performance after training ($r = 0.70$, log-linear). Shown for $\tau_s = 0.5$. **(B)** Same as (A), but with the number of pieces after training. **(C)** Number of pieces before and after training. The diagonal indicates no change in pieces. **(D)** Same as (A), but for nLIF neurons with time constant $\tau_s = 0.1$. **(E)** Illustration of the Yin-Yang dataset with three classes: the two halves and the dots. **(F)** Causal pieces (each piece is indicated by a different colour) of the hidden layer for a poor initialisation, evaluated using only training samples. **(G)** Same as (F), but for one of the best initialisations.

final accuracy achieved on the test split. Before training, all SNNs perform around chance level independent of the initial number of causal pieces.

Interestingly, we find that it seems hard to recover from a poor initialisation with a low number of pieces through training (Fig. 3C). Although some SNNs with an initially small piece count achieve a higher – but not the best – test accuracy, this is only possible because the number of pieces was significantly increased during training, in agreement with Theorem 1. However, for most SNNs, the piece count decreases during training. A similar result is obtained for smaller time constants (Fig. 3D), where SNNs with an initial low piece count perform even worse on average. Due to the decreased time constant, inputs can drive the neuron's membrane potential faster above the threshold, meaning that weights have to be carefully tuned to allow integration of late input spikes – thus complicating the creation of new causal pieces during training. Examples of the causal piece structure on the training data of the Yin-Yang dataset are shown for the hidden layer of an SNN achieving low (Fig. 3F) and high performance (Fig. 3G) – clearly highlighting the difference in the number of causal pieces before and after training (separate clusters with similar colours are different causal pieces). We also find that initially, training samples frequently switch between different causal pieces – a reorganisation that slows down and stabilises in later epochs. For SNNs with a low initial piece count, reorganisation eventually dies out, while for SNNs with high initial count, it never stops (see Section A.1.3).

A similar trend is observed for single-spike LIF neurons with finite membrane and synapse time constant (Fig. 4A-D). In this case, recovery from an initial low piece count is even rarer, most likely because creating causal pieces is harder than in the nLIF case: a valid causal set not only requires that the weights of input spikes exceed the threshold, but that the input spikes also co-occur sufficiently close together in time. This illustrates that, while Theorem 1 is restricted to nLIF neurons, the applicability of causal pieces extends to more complex neuron models with leaky dynamics.

These results are not restricted to the Yin-Yang dataset. In Fig. 4E-H, we observe the same trend for shallow SNNs with 200 hidden neurons (and linear readout) trained via exact gradient descent on the Fashion-MNIST dataset [41] and the EuroSAT RGB benchmark [42], a land cover classification task using real images from the European Space Agency's (ESA) Sentinel-2 satellite. Consequently, we hypothesise that the number of causal pieces is a promising metric for identifying good initialisation

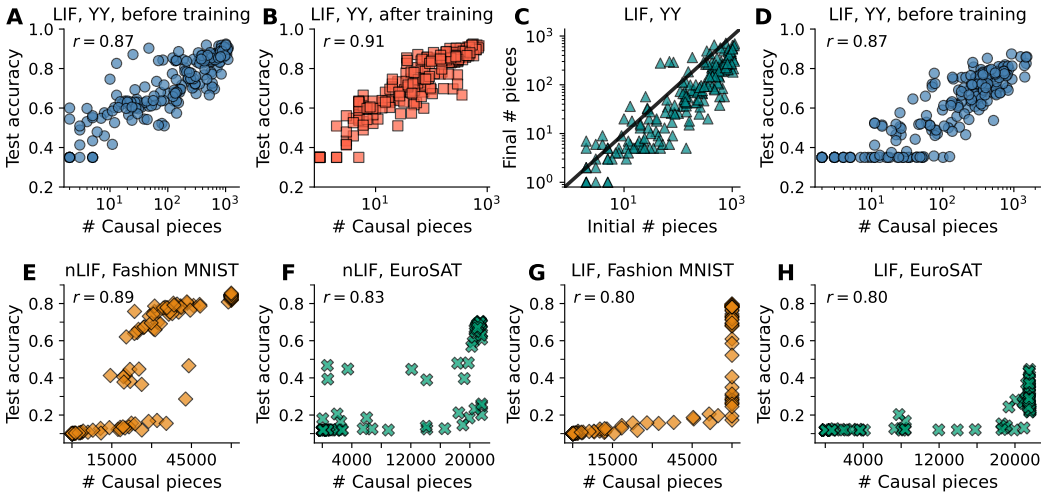

Figure 4: Experimental validation using more complex neuron models and datasets. **(A-C)** Same as Fig. 3A-C, but for LIF neurons with $\tau_{\mathrm{m}} = 0.5$ and $\tau_{\mathrm{s}} = 0.25$. **(D)** Same as Fig. 3D, but for LIF neurons with $\tau_{\mathrm{m}} = 0.1$ and $\tau_{\mathrm{s}} = 0.05$. **(E-F)** The number of causal pieces at initialisation correlates with final test accuracy for nLIF SNNs trained on Fashion MNIST and EuroSAT. **(G-H)** Same as E-F, but for LIF SNNs with $\tau_{\mathrm{m}} = 0.5$ and $\tau_{\mathrm{s}} = 0.25$. E-H are plotted on linear axes for easier readability. Reported correlations are also calculated using the untransformed number of causal pieces. Log-linear correlations are (from E to H): 0.74, 0.69, 0.62, and 0.62.

schemes for SNNs – although it is certainly not the sole predictor of training success. For image data, pixel values were translated into spike time latencies before feeding them into the network (see Section A.5).

3.5 Increasing the number of pieces by going wide or deep

As seen in the previous subsection, a large initial number of pieces is strongly associated with successful training in the settings studied here. Therefore, it is a natural question to ask how this number can be increased. From the previous results, an obvious option is to optimise the weight initialisation to yield networks with many pieces. We investigate this for networks ($[4, 100, 3]$ neurons) with weights initialised randomly from either a Gaussian or a uniform distribution, using a Yin-Yang dataset obtained from a 400×400 grid on the data domain. We chose a larger dataset here to properly probe the number of causal pieces. In case of a Gaussian distribution, the weights projecting into layer $\ell \in \mathbb{N}$, $W^{(\ell)} \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$, are initialised by sampling from $\mathcal{N}(\alpha_0 \cdot n_{\ell-1}^{-\alpha_1}, \left[\alpha_2 \cdot n_{\ell-1}^{-\alpha_3}\right]^2)$ where n_ℓ is the number of neurons in layer ℓ. Similarly, in case of a uniform distribution, weights are sampled from $\mathcal{U}(-v_0 + v_1, v_0 + v_1)$ with $v_0 = \beta_0 \cdot n_{\ell-1}^{-\beta_1}$ and $v_1 = \beta_2 \cdot n_{\ell-1}^{-\beta_3}$. The parameters α_i and β_i ($i \in [0, 3]$) are found using a simple evolutionary algorithm that maximises the number of causal pieces (Section A.5.1). For this specific setup, we found $\alpha_0 = 1.69$, $\alpha_1 = 0.79$, $\alpha_2 = 1.13$, $\alpha_3 = 0.49$ and $\beta_0 = 1.85$, $\beta_1 = 0.39$, $\beta_2 = 1.02$, $\beta_3 = 0.54$. The corresponding probabilities p_k^q of these weight initialisations are shown in Fig. 5A. As for the single neuron case, the weight distributions feature non-zero means. We visualise the causal pieces for a single output neuron in Section A.1.5.

Another option to adjust the number of pieces is to change the width and depth of the SNN, as shown in Fig. 5B,C. We present three scenarios: (line) a shallow network where the width is steadily increased by increments of 20 neurons, (dashed) a deep network, where in each increment an additional hidden layer with 20 neurons is added, and (dotted) the same as for dashed, but with 40 neurons per hidden layer. Results are shown for the two distributions found using evolutionary optimisation. For the shallow network, the number of pieces grows consistently with increased network width, although slower than for deep networks and with saturation setting in for very wide networks. In case of deep networks, the number of pieces grows rapidly initially, but then stagnates to a constant number of causal pieces. The effect is more pronounced if the hidden layers are wider, with a much stronger increase and final number of causal pieces for the network with 40 neurons per layer. Different from the expected exponential increase, we rather see a logistic growth. In fact, fitting logistic curves of the form $\gamma_0/(\gamma_1 + e^{-\gamma_2 N})$ with $\gamma_i \in \mathbb{R}$ and N the number of neurons, we get a median relative error of $4 \cdot 10^{-2}$ (shallow), $2 \cdot 10^{-2}$ (deep 20), and $2 \cdot 10^{-2}$ (deep 40) for the Gaussian initialisation, and $9 \cdot 10^{-2}$ (shallow), $2 \cdot 10^{-2}$ (deep 20), and $5 \cdot 10^{-3}$ (deep 40) for the uniform one. The saturation for (deep 20) most likely occurs due to a diminishing effect of pieces being split by consecutive layers, similar to how for ReLU ANNs, an exponential growth in linear pieces with depth is often not observed in practice [27]. For all other cases, saturation occurs as we reach the maximum number of causal pieces that can be counted using the data samples.

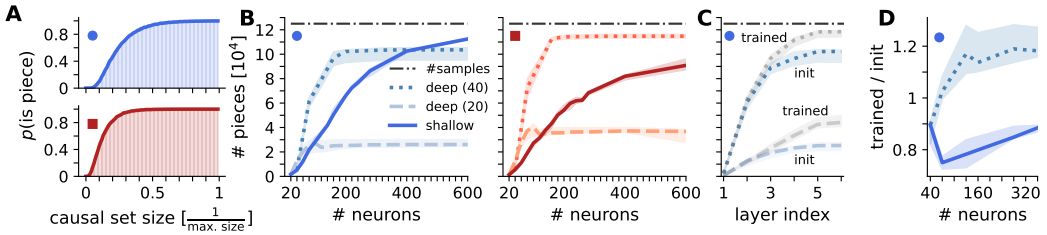

Figure 5: Width-and depth dependence of causal pieces. **(A)** p_k^q of the optimised (top, dot) normal, and (bottom, square) uniform initialisation. **(B)** Number of pieces for shallow and deep networks. The maximum number, which is the number of input samples used to evaluate the number of causal pieces, is shown as a dash-dotted line. **(C)** Number of pieces per layer in a single network, before and after training. **(D)** Increase in the total number of pieces for deep and shallow networks. Markers denote results that belong together. We show medians (lines) and quartiles (shaded areas).

In Fig. 5C, we show the number of pieces per layer for a network with 5 hidden layers. Similarly to how initially adding hidden layers increased the number of pieces drastically in Fig. 5B, the highest increase is seen in the first few layers, with diminishing returns in deeper layers. In contrast, if we compare the number of pieces per layer before and after training, we find a slight increase in the number of causal pieces for deep layers. If we just focus on the total number of pieces of the whole network, we find that shallow networks end up with less pieces than at initialisation, while deep networks end up with more (Fig. 5D). Most likely, this is because in a deep network, the number of pieces can be optimised by improving the misalignment of pieces between consecutive layers.

3.6 Large number of causal pieces in spiking neural networks with only positive weights

Inspired by [15], we now study the case of SNNs with only excitatory neurons. In the mammalian neocortex, around 80% [43] of neurons are excitatory, i.e., their synapses only excite other neurons, which is equivalent to neurons having only positive outgoing weights in our nLIF neural networks. Although having only positive weights seems limiting at first, it comes with a significant advantage: controlling for continuity between causal pieces becomes much easier. In fact, the network is globally Lipschitz continuous as long as for each neuron, the input weights have a sum larger than the threshold – which can be easily enforced during training, e.g., through a regularisation term. The global Lipschitz constant of a neural network can be used to derive its covering number, which provides an upper bound for the network's generalisation error [15, 44] (see Theorem 3 in Section A.3.1).

We again optimise the parameters of two initialisation distributions, this time a lognormal and a uniform distribution – which both lead to networks with a similar number of pieces than for distributions with both positive and negative values. Their respective p_k^q probabilities are shown in Fig. 6A. Using these initialisation schemes, we train deep networks (two hidden layers with 200 and 100 neurons) composed of an SNN with positive weights and a single linear readout layer (with positive and negative weights, see Fig. 6C) on three different benchmarks: Yin-Yang, MNIST, and EuroSAT RGB – achieving performance levels similar to fully-connected ANNs, and far outcompeting linear models (Fig. 6D). An illustration of the causal pieces is shown in Section A.1.

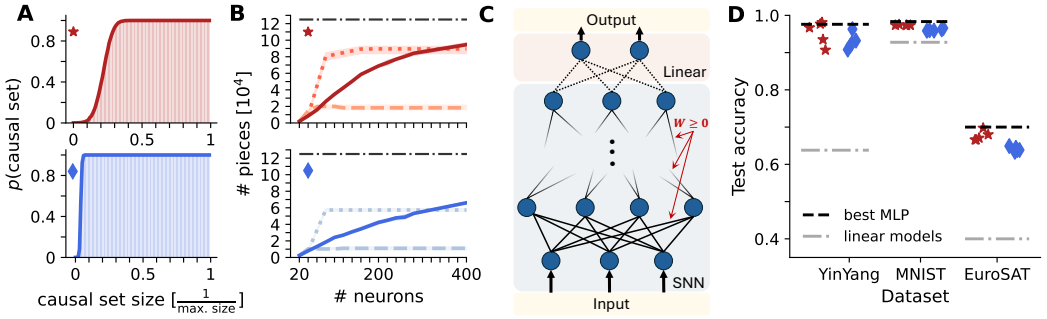

Figure 6: SNNs with only positive weights. **(A)** p_k^q for (top, star) lognormal, and (bottom, diamond) uniform initialisation. **(B)** Number of pieces for shallow and deep networks. Labels as in Fig. 5B. **(C)** Used network architecture. **(D)** Performance on benchmarks. Each training run was repeated 5 times for different random seeds. Markers denote results that belong together.

4 Discussion

The presented results demonstrate that causal pieces are not only a powerful tool for increasing our understanding of SNNs, but also for guiding the design of improved network architectures and training methods. The causal piece framework naturally fits the discontinuous, event-based nature of SNNs. It enables a mathematically rigorous analysis of SNNs without requiring restrictive assumptions such as positive weights and highly abstract neuron models. Across all reported experiments, we found that initialising weights from distributions with non-zero mean yields SNNs with the highest number of causal pieces – a strategy also used in [6] for single-spike LIF neurons.

While previous work has briefly explored linear pieces in simplified spike-response models with linearly rising postsynaptic potentials [14], our work lays the foundation for elevating this concept

to more realistic neuron models. The decomposition of the input and parameter space into causal pieces should, in principle, be applicable to any spiking neuron model commonly used in practice, especially since only the connectome and spike times of an SNN are required to obtain causal pieces. Although our theory and experiments focused on single-spike coding, we note that key results – such as the approximation bound in Theorem 1 – extend to the multi-spike setting through a transference principle [30]. Moreover, this transference principle provides a promising path forward to define causal pieces for multi-spike SNNs and derive approximation bounds such as Theorem 1 directly. For spiking neurons with leak, the definition of causal sets, and hence causal subnetworks and pieces, remains unchanged (cf. [6, 7]). Accordingly, causal pieces can be readily evaluated for general LIF neurons with single-spike coding, which we demonstrated in simulations (Fig. 4). Similar to affine pieces, causal pieces are limited to feedforward neural networks (including convolutional networks). The scope of our contribution is summarised in Table 1. Extending mathematical results to more

Table 1: Summary of our work's scope.

Defined	Causal sets, subnetworks, and pieces for feedforward, single-spike SNNs, calculated from connectivity and observed spike times. The definitions are not specific to nLIF dynamics and apply unchanged to standard LIF neurons.
Proved	Theorem 1, local Lipschitz continuity within pieces (Theorem 3), and counting results for feedforward nLIF networks. The limited multi-spike extension follows through the cited transference principle.
Empirically supported	Correlation results for single-spike nLIF and finite-time-constant LIF networks trained with exact spike-time gradients on the three reported static classification benchmarks, including signed- and positive-weight neural networks.
Not established here	Arbitrary multi-spike dynamics, surrogate-gradient training, and recurrent neural networks.

complex neuron models will require non-trivial adaptations of the proof strategy. Generally, as long as the output spike time of a neuron can be rewritten as a linear function of its inputs (e.g., via a change of variables), the same proof strategy as presented in this work remains applicable – which is true for other simple neuron models, such as the simplified spike-response model used in [45]. Models with analytical solutions for spike times, such as LIF with $\tau_{\mathrm{m}} = \tau_{\mathrm{s}}$ or $\tau_{\mathrm{m}} = 2\tau_{\mathrm{s}}$ [6], provide a promising angle for generalising the proven theorems to neurons with leakage. In any case, our experimental results with standard LIF neurons indicate that theorems such as Theorem 1 may partially extend to more complex neuron models.

Another important property of neural networks is their ability to generalise to previously unseen data. Causal pieces, like their counterparts linear pieces used for ANNs, are primarily a tool for assessing the approximation ability of neural networks though. Typical approaches for bounding the generalisation error of neural networks use covering numbers [15], which can only be calculated for neural networks that are globally Lipschitz continuous – a property that does not hold for SNNs in general. However, novel measures based on causal pieces, e.g., comparing the number of pieces the training and validation data fall into, might provide novel insights into the generalisation capabilities of SNNs and ANNs alike. As a first step, we already report test errors in simulations in this work.

To conclude, we are confident that this approach is a promising direction that has the potential to be applicable to a wide range of neuron models and enables principled comparisons across spiking neuron models as well as with ReLU-based ANNs. Moreover, we believe that the usefulness of causal pieces extends beyond technical applications and domains, potentially providing novel ways to study biological neurons by analysing their causal piece structure derived from experimental data.

Acknowledgments and Disclosure of Funding

We would like to thank Julian Göltz for helpful feedback on the manuscript. D.D. was funded by the Horizon Europe's Marie Skłodowska-Curie Actions (MSCA) Project 101103062 (BASE). Simulations were performed on the Austrian Scientific Computing (ASC) cluster.

References

[1] D. Izzo et al. "Neuromorphic computing and sensing in space". In: *Artificial Intelligence for Space: AI4SPACE*. CRC Press, 2022, pp. 107–159.

[2] J. Schumann. *Radiation Tolerance and Mitigation for Neuromorphic Processors*. Tech. rep. NTRS Author Affiliations: KBR (United States) NTRS Document ID: 20220013182 NTRS Research Center: Ames Research Center (ARC). Jan. 2022. URL: https://ntrs.nasa.gov/citations/20220013182 (visited on 05/28/2024).

[3] P. Lunghi et al. "Investigation of Low-Energy Spiking Neural Networks Based on Temporal Coding for Scene Classification". In: *75th International Astronautical Congress (IAC 2024)*. 2024, pp. 1–13.

[4] E. O. Neftci, H. Mostafa, and F. Zenke. "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks". In: *IEEE Signal Processing Magazine* 36.6 (2019), pp. 51–63.

[5] H. Mostafa. "Supervised learning based on temporal coding in spiking neural networks". In: *IEEE transactions on neural networks and learning systems* 29.7 (2017), pp. 3227–3235.

[6] J. Göltz et al. "Fast and energy-efficient neuromorphic deep learning with first-spike times". In: *Nature machine intelligence* 3.9 (2021), pp. 823–835.

[7] I. M. Comsa et al. "Temporal coding in spiking neural networks with alpha synaptic function". In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 8529–8533.

[8] C. Klos and R.-M. Memmesheimer. "Smooth exact gradient descent learning in spiking neural networks". In: *Physical Review Letters* 134.2 (2025), p. 027301.

[9] C. Frenkel, D. Bol, and G. Indiveri. "Bottom-up and top-down approaches for the design of neuromorphic processing systems: Tradeoffs and synergies between natural and artificial intelligence". In: *Proceedings of the IEEE* 111.6 (2023), pp. 623–652.

[10] M. Davies et al. "Loihi: A Neuromorphic Manycore Processor with On-Chip Learning". In: *IEEE Micro* 38.1 (Jan. 2018), pp. 82–99. ISSN: 0272-1732, 1937-4143. DOI: 10.1109/MM.2018.112130359. URL: https://ieeexplore.ieee.org/document/8259423/ (visited on 05/23/2024).

[11] G. Orchard et al. "Efficient neuromorphic signal processing with loihi 2". In: *2021 IEEE Workshop on Signal Processing Systems (SiPS)*. IEEE. 2021, pp. 254–259.

[12] B. Cramer et al. "Surrogate gradients for analog neuromorphic computing". In: *Proceedings of the National Academy of Sciences* 119.4 (2022), e2109194119.

[13] P. Spilger et al. "hxtorch. snn: Machine-learning-inspired spiking neural network modeling on BrainScaleS-2". In: *Proceedings of the 2023 Annual Neuro-Inspired Computational Elements Conference*. 2023, pp. 57–62.

[14] M. Singh, A. Fono, and G. Kutyniok. "Expressivity of Spiking Neural Networks through the Spike Response Model". In: *UniReps: the First Workshop on Unifying Representations in Neural Models*. 2023.

[15] A. M. Neuman, D. Dold, and P. C. Petersen. "Stable learning using spiking neural networks equipped with affine encoders and decoders". In: *Journal of Machine Learning Research* (2025). URL: https://jmlr.org/papers/v26/24-0658.html.

[16] S. Davidson and S. B. Furber. "Comparison of artificial and spiking neural networks on digital hardware". In: *Frontiers in Neuroscience* 15 (2021), p. 651141.

[17] B. Yin, F. Corradi, and S. M. Bohté. "Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks". In: *Nature Machine Intelligence* 3.10 (2021), pp. 905–913.

[18] A. S. Kucik and G. Meoni. "Investigating spiking neural networks for energy-efficient on-board ai applications. a case study in land cover and land use classification". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 2020–2030.

[19] M. Dampfhoffer et al. "Are SNNs really more energy-efficient than ANNs? An in-depth hardware-aware study". In: *IEEE Transactions on Emerging Topics in Computational Intelligence* 7.3 (2022), pp. 731–741. DOI: 10.1109/TETCI.2022.3214509.

[20] W. Maass. "Networks of spiking neurons: the third generation of neural network models". In: *Neural networks* 10.9 (1997), pp. 1659–1671.

[21] W. Maass and M. Schmitt. “On the complexity of learning for spiking neurons with temporal coding”. In: *Information and Computation* 153.1 (1999), pp. 26–46. DOI: 10.1006/inco.1999.2806.

[22] A. M. Zador and B. A. Pearlmutter. “VC dimension of an integrate-and-fire neuron model”. In: *Proceedings of the ninth annual conference on Computational learning theory*. 1996, pp. 10–18. DOI: 10.1145/238061.238064.

[23] A. Stanojevic et al. “An exact mapping from ReLU networks to spiking neural networks”. In: *Neural Networks* 168 (2023), pp. 74–88.

[24] C. L. Frenzen, T. Sasao, and J. T. Butler. “On the number of segments needed in a piecewise linear approximation”. In: *Journal of Computational and Applied mathematics* 234.2 (2010), pp. 437–446.

[25] G. F. Montufar et al. “On the number of linear regions of deep neural networks”. In: *Advances in neural information processing systems* 27 (2014).

[26] M. Telgarsky. “Benefits of depth in neural networks”. In: *Conference on learning theory*. PMLR. 2016, pp. 1517–1539. DOI: 10.48550/arXiv.1602.04485.

[27] B. Hanin and D. Rolnick. “Complexity of linear regions in deep networks”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 2596–2604.

[28] D. A. Nguyen et al. “Time to Spike? Understanding the Representational Power of Spiking Neural Networks in Discrete Time”. In: *International Conference on Machine Learning*. PMLR. 2025, pp. 45954–45987.

[29] J. Rossbroich, J. Gygax, and F. Zenke. “Fluctuation-driven initialization for spiking neural network training”. In: *Neuromorphic Computing and Engineering* 2.4 (2022), p. 044016.

[30] D. Dold and P. C. Petersen. “Equivalence of approximation by networks of single-and multi-spike neurons”. In: *Artificial Neural Networks and Machine Learning – ICANN 2026: Proceedings, Part 1. Springer* (2026). DOI: 10.48550/arXiv.2603.13478.

[31] D. Dold. *snnpiece*. Version 2c57054. 2026. URL: https://github.com/dodo47/snnpiece.

[32] J. Göltz et al. “DelGrad: exact event-based gradients for training delays and weights on spiking neuromorphic hardware”. In: *Nat Commun., 16(1):8245* (2025).

[33] K. Che et al. “ETTFS: An Efficient Training Framework for Time-to-First-Spike Neuron”. In: *arXiv preprint arXiv:2410.23619* (2024).

[34] G. Bellec et al. “Long short-term memory and learning-to-learn in networks of spiking neurons”. In: *Advances in neural information processing systems* 31 (2018).

[35] F. Zenke and T. P. Vogels. “The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks”. In: *Neural computation* 33.4 (2021), pp. 899–925.

[36] J. H. Lee, T. Delbruck, and M. Pfeiffer. “Training deep spiking neural networks using backpropagation”. In: *Frontiers in neuroscience* 10 (2016), p. 508.

[37] J. Ding et al. “Accelerating training of deep spiking neural networks with parameter initialization”. In: (2022). URL: https://openreview.net/forum?id=T8BnDXDTcFZ.

[38] E. S. Andersen. “On the fluctuations of sums of random variables II”. In: *Mathematica Scandinavica* (1954), pp. 195–223.

[39] S. N. Majumdar. “Universal first-passage properties of discrete-time random walks and Lévy flights on a line: Statistics of the global maximum and records”. In: *Physica A: Statistical Mechanics and its Applications* 389.20 (2010), pp. 4299–4316.

[40] L. Kriener, J. Göltz, and M. A. Petrovici. “The yin-yang dataset”. In: *Proceedings of the 2022 Annual Neuro-Inspired Computational Elements Conference*. 2022, pp. 107–111.

[41] H. Xiao, K. Rasul, and R. Vollgraf. “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms”. In: *arXiv preprint arXiv:1708.07747* (2017).

[42] P. Helber et al. “Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification”. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 12.7 (2019), pp. 2217–2226.

[43] R. Nieuwenhuys. “The neocortex: an overview of its evolutionary development, structural organization and synaptology”. In: *Anatomy and embryology* 190.4 (1994), pp. 307–337.

[44] P. Petersen and J. Zech. “Mathematical theory of deep learning”. In: *arXiv preprint arXiv:2407.18384* (2024).

[45] A. Stanojevic et al. “High-performance deep spiking neural networks with 0.3 spikes per neuron”. In: *Nature Communications* 15.1 (2024), p. 6793.

[46] R. Johnson. *Elementary central binomial coefficient estimates*. Mathematics Stack Exchange (version: 2023-10-23). URL: https://math.stackexchange.com/q/932509.

[47] A. Paszke. “Pytorch: An imperative style, high-performance deep learning library”. In: *arXiv preprint arXiv:1912.01703* (2019).

[48] B. J. Kim et al. “On the ideal number of groups for isometric gradient propagation”. In: *Neurocomputing* 573 (2024), p. 127217.

[49] W. Senn et al. “A neuronal least-action principle for real-time learning in cortical circuits”. In: *ELife* 12 (2024), RP89674.

A Technical appendices and supplementary material

A.1 Additional figures and simulation results

A.1.1 Illustration: causal sets and causal subnetworks

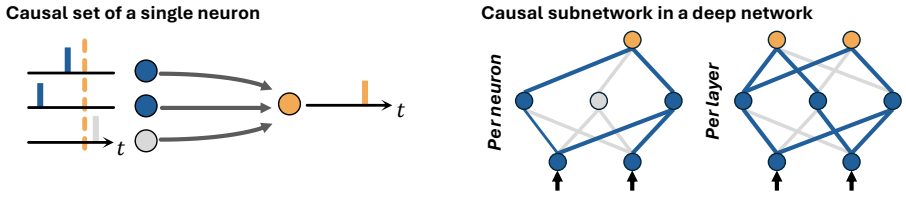

Figure 7: Left: A single output neuron (orange) receiving input from three neurons. Only those input neurons (blue) that spike before the output neuron (i.e., before the dotted line) are part of the causal set. Right: In deep networks, this corresponds to subnetworks (blue), here shown for a single output neuron (left), or the whole output layer (right).

A.1.2 Time complexity

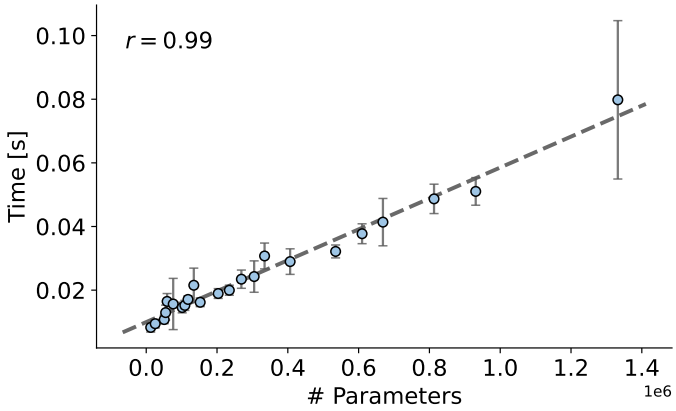

Figure 8: Wall-clock time required to calculate the causal piece for a single data sample, using MNIST. Network architectures range from a single to three hidden layers. For each network architecture, the average of 40 different runs is shown. Error bars show the standard deviation, and the dashed line is a linear fit.

A.1.3 Causal piece reorganisation during training

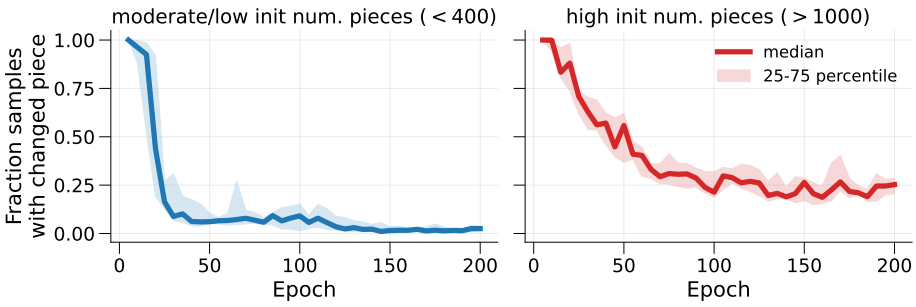

Figure 9: Fraction of data samples that switched causal piece compared to five epochs before. For 10 nLIF SNNs trained on Yin-Yang with low/intermediate initial causal piece count, and 10 nLIF SNNs with high initial count. The neurons use $\tau_s = 0.5$.

A.1.4 Ablation result: fraction of active neurons

We test whether our observation can be explained via the average number of neurons that are active after initialisation, averaged over the training data samples.

Both for nLIF and LIF, the logarithm of the number of causal pieces at initialisation and the number of active neurons have a correlation coefficient of around 0.7-0.8, which is expected. However, when training a linear model for predicting the chance-corrected test accuracy, we find that the logarithm of the number of causal pieces is more predictive than the fraction of active neurons (Table 2).

Table 2: Predictive power of feature sets for explaining final test accuracy. Lower Sum of Squared Residuals (SSR) is better.

Model / Dataset	Feature set	SSR
nLIF on Yin-Yang	Active fraction only	5.17
nLIF on Yin-Yang	Log initial piece count only	2.95
nLIF on Yin-Yang	Active fraction + log initial piece count	**2.63**
LIF on Yin-Yang	Active fraction only	3.90
LIF on Yin-Yang	Log initial piece count only	1.17
LIF on Yin-Yang	Active fraction + log initial piece count	**1.10**

A.1.5 Causal piece structures

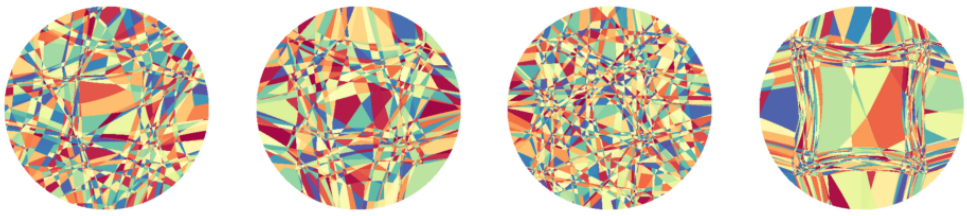

Figure 10: Causal pieces (coloured regions) of one of the output neurons for an nLIF neural network with $[4, 30, 3]$ neurons, using the initialisations obtained through evolutionary optimisation (Fig. 5 and Fig. 6). Causal pieces are evaluated using a 400×400 grid on the data domain.

A.2 Methods

A.2.1 Analytical solution of output spike time

Given the causal set $\mathcal{C}_i^{(\ell)}$, the corresponding output spike time $t_i^{(\ell)}$ of an nLIF neuron is [5]

$$t_i^{(\ell)} = \begin{cases} \tau_\mathrm{s} \ln\left(\frac{\sum_{j\in\mathcal{C}_i^{(\ell)}} W_{ij}^{(\ell)} e^{t_j^{(\ell-1)}/\,\tau_\mathrm{s}}}{\sum_{j\in\mathcal{C}_i^{(\ell)}} W_{ij}^{(\ell)} - \vartheta}\right) & \text{if } \mathcal{C}_i^{(\ell)} \neq \emptyset\,, \\ \infty, & \text{else}\,, \end{cases} \tag{5}$$

where the spike time is set to infinity if the input does not cause the neuron to spike. To find the causal set in simulations, we use the iterative approach described in Section A.2.3.

A.2.2 Relationship between nLIF and LIF neuron models

The current-based LIF neuron model with exponential synaptic kernel is given by

$$\frac{\mathrm{d}}{\mathrm{d}t} u_i^{(\ell)}(t) = \frac{1}{\tau_\mathrm{m}}(u_\mathrm{rest} - u_i^{(\ell)}(t)) + \frac{1}{\tau_\mathrm{s}} \sum_j W_{ij}^{(\ell)}\, \Theta\left(t - t_j^{(\ell-1)}\right) \exp\left(-\frac{t - t_j^{(\ell-1)}}{\tau_\mathrm{s}}\right), \tag{6}$$

where $u_i^{(\ell)}(t) \in \mathbb{R}$ is the membrane potential of neuron i in layer ℓ at time $t \in \mathbb{R}$, $W_{ij}^{(\ell)} \in \mathbb{R}$ is the synaptic weight connecting neuron j of layer $\ell-1$ to neuron i of layer ℓ, $t_j^{(\ell-1)}$ is the spike time of neuron j in layer $\ell-1$, $\tau_{\mathrm{m}} \in \mathbb{R}^+$ and $\tau_{\mathrm{s}} \in \mathbb{R}^+$ are the membrane and synaptic integration time constants, $\Theta(\cdot)$ is the Heaviside function, and $u_{\mathrm{rest}} \in \mathbb{R}$ is the rest value of the membrane potential.

In the special case $\tau_{\mathrm{m}} \gg \tau_{\mathrm{s}}$, this simplifies to

$$\frac{\mathrm{d}}{\mathrm{d}t} u_i^{(\ell)}(t) = \frac{1}{\tau_{\mathrm{s}}} \sum_j W_{ij}^{(\ell)} \, \Theta\left(t - t_j^{(\ell-1)}\right) \exp\left(-\frac{t - t_j^{(\ell-1)}}{\tau_{\mathrm{s}}}\right), \tag{7}$$

which can be solved for $u_i^{(\ell)}(t)$ by integration:

$$u_i^{(\ell)}(t) = \int_{-\infty}^{t} \frac{\mathrm{d}}{\mathrm{d}t'} u_i^{(\ell)}(t') \, \mathrm{d}t' = \sum_{t_j^{(\ell-1)} \le t} W_{ij}^{(\ell)} \left[1 - \exp\left(-\frac{t - t_j^{(\ell-1)}}{\tau_{\mathrm{s}}}\right)\right]. \tag{8}$$

A.2.3 Calculating causal sets for nLIF

To find the causal set, we use the following approach: In case of an nLIF neuron that has $N_{\ell-1}$ input spike times $t_j^{(\ell-1)}$ with weights $W_{ij}^{(\ell)}$, we first define $\mathcal{K} = \{j_1, j_2, ..., j_{N_{\ell-1}}\}$ with $t_{j_1} \le t_{j_2} \le ... \le t_{j_{N_{\ell-1}}}$. Furthermore, we set $\mathcal{K}_k = \{j_1, ..., j_k\}$ for $k > 0$. The causal set is then given by the subset $\mathcal{K}_m$ with the smallest index m satisfying

$$\textbf{1.} \sum_{j \in \mathcal{K}_m} W_{ij}^{(\ell)} \ge \vartheta \quad \textbf{and 2.} \quad \mathcal{K}_m = \{j : t_j^{(\ell-1)} \le t_i^{(\ell)}\}, t_i^{(\ell)} = \tau_{\mathrm{s}} \ln\left(\frac{\sum_{j \in \mathcal{K}_m} W_{ij}^{(\ell)} e^{t_j^{(\ell-1)} / \tau_{\mathrm{s}}}}{\sum_{j \in \mathcal{K}_m} W_{ij}^{(\ell)} - \vartheta}\right).$$

These two conditions are summarised as follows: (1) the inputs have to be strong enough to drive the membrane potential across the threshold, and (2) all inputs that did not cause the spike at time $t_i^{(\ell)}$ occur after it. The criterion of selecting the set with minimal m ensures that we find the earliest possible output spike time. If no such set is found, the causal set is defined as the empty set, reflecting the fact that none of the inputs caused the neuron to spike. In simulations, we set the output spike time to a sufficiently large value such that it does not affect any other neuron in the network, emulating spiking at infinity.

A.3 Additional theorems

A.3.1 Lipschitz continuity

To ease the notation, we drop the nested list notation of causal subnetworks in the following. We first state the result for a single nLIF neuron (Theorem 3).

Theorem 3 (Lipschitz continuous) *Let $N_0 \in \mathbb{N}$, $j \in [1, N_0]$, and $\mathcal{C}_1^{(1)} \subset [1, \ldots, N_0]$. Moreover, let $a, b \in \mathbb{P}[\mathcal{C}_1^{(1)}]$ be the input to a single nLIF neuron with N_0 input times. Then the output spike time (Eq. (5)) is Lipschitz continuous with respect to input times and weights $W_{1j}^1 \in \mathbb{R}$, $j \in [1, N_0]$:*

$$\left\| t_1^{(1)}(a) - t_1^{(1)}(b) \right\|_{L^\infty(\mathbb{P}[\mathcal{C}_1^{(1)}])} \le 2|\mathcal{C}_1^{(1)}| \max\left(\frac{\bar{W}}{\delta}, \frac{\tau_{\mathrm{s}}}{\delta}\right) \|a - b\|_{L^\infty(\mathbb{P}[\mathcal{C}_1^{(1)}])}, \tag{9}$$

where $|\mathcal{C}|$ denotes the cardinality of $\mathcal{C}$, $\|W_{1j}^{(1)}\| \le \bar{W}$, $\delta < \sum_{j \in \mathcal{C}_1^{(1)}} W_{1j}^{(1)} - \vartheta$.

The proof is given in Sections A.4.1 and A.4.2. In addition, the output spike time may change continuously, discontinuously, or become undefined when transitioning between causal pieces. Which of these occurs can be determined by inspecting the causal set: if all added or removed input neurons have identical spike times, the output spike time changes continuously; otherwise, it changes discontinuously. If the causal set would reach maximum size, but all inputs together do not reach the threshold, the output spike disappears. The corresponding result for entire networks follows from the fact that the composition of Lipschitz-continuous functions is itself Lipschitz continuous.

The Lipschitz constant can be reduced by choosing parameters that produce sparsely populated causal sets (small $|\mathcal{C}|$) that strongly overstep the threshold (large δ). However, the contribution of the size of

the causal sets in the Lipschitz constant is counter-balanced by the maximum weight $\bar{W}$, which has to be increased with decreasing set sizes to ensure that the sum of the weights exceeds ϑ.

A.4 Mathematical proofs

A.4.1 Proof of continuity and differentiability

To improve readability, we drop the layer and output neuron indices in the following. First note that within a causal piece, the output spike time Eq. (5) is a composition of continuous and differentiable functions, and hence itself continuous and differentiable with respect to input spike times and weights.

In the following, we prove under which conditions the output spike time is a continuous function of input spike times and weights when crossing between neighbouring causal pieces. First, let $\mathcal{C}$ be the causal set of an nLIF neuron with input spike times $[t_0, ..., t_{N-1}]$, weights $[W_0, ..., W_{N-1}]$, and output spike time

$$t = \tau_\mathrm{s} \ln\left(T\right) = \tau_\mathrm{s} \ln\left(\frac{\sum_{j\in\mathcal{C}} W_j e^{t_j/\ \tau_\mathrm{s}}}{\sum_{j\in\mathcal{C}} W_j - \vartheta}\right). \tag{10}$$

Let $\mathcal{C}'$ be the causal set of a neighbouring causal piece, with spike times $[\tilde{t}_0, ..., \tilde{t}_{N-1}, \tilde{t}_N]$, weights $[\tilde{W}_0, ..., \tilde{W}_{N-1}, \tilde{W}_N]$, and output spike time $\tilde{t}$:

$$\tilde{t} = \tau_\mathrm{s} \ln\left(\tilde{T}\right) = \tau_\mathrm{s} \ln\left(\frac{\sum_{j\in\mathcal{C}} \tilde{W}_j e^{\tilde{t}_j/\ \tau_\mathrm{s}} + \tilde{W}_N e^{\tilde{t}_N/\ \tau_\mathrm{s}}}{\sum_{j\in\mathcal{C}} \tilde{W}_j + \tilde{W}_N - \vartheta}\right). \tag{11}$$

We assume that the output spike time of $\mathcal{C}$ is along the border between the two causal pieces, meaning that $t = t_N$. Since output spike times can be shifted by Δ by shifting all input spike times by Δ, without loss of generality, we assume that $\forall x \in \{t, \tilde{t}, t_0, ..., t_N, \tilde{t}_0, ..., \tilde{t_N}\}$, $x \geq 0$. All spike times are finite, thus $\exists t_\mathrm{max}$ with $0 < t_\mathrm{max} < \infty$ such that $\forall x \in \{t, \tilde{t}, t_0, ..., t_N, \tilde{t}_0, ..., \tilde{t_N}\}$, $x \leq t_\mathrm{max}$. Similarly, $\exists \bar{W} > 0$ such that $\forall \omega \in \{W_0, ..., W_N, \tilde{W}_0, ..., \tilde{W}_N\}$, $\|\omega\| \leq \bar{W}$. Furthermore, $\exists \epsilon_\vartheta$ with $0 < \epsilon_\vartheta < \infty$ such that $\epsilon_\vartheta < \sum_{j\in\mathcal{C}} \tilde{W}_j + \tilde{W}_N - \vartheta$. Lastly, we highlight the following identity:

$$\begin{aligned}
T &= T \cdot \frac{\sum_{j\in\mathcal{C}} W_j + M - \vartheta}{\sum_{j\in\mathcal{C}} W_j + M - \vartheta} && (12)\\
&= T \cdot \frac{\sum_{j\in\mathcal{C}} W_j - \vartheta}{\sum_{j\in\mathcal{C}} W_j + M - \vartheta} + \frac{M \cdot T}{\sum_{j\in\mathcal{C}} W_j + M - \vartheta} && (13)\\
&= \frac{\sum_{j\in\mathcal{C}} W_j e^{t_j/\ \tau_\mathrm{s}}}{\sum_{j\in\mathcal{C}} W_j - \vartheta} \cdot \frac{\sum_{j\in\mathcal{C}} W_j - \vartheta}{\sum_{j\in\mathcal{C}} W_j + M - \vartheta} + \frac{M \cdot T}{\sum_{j\in\mathcal{C}} W_j + M - \vartheta} && (14)\\
&= \frac{\sum_{j\in\mathcal{C}} W_j e^{t_j/\ \tau_\mathrm{s}} + M \cdot e^{t_N/\ \tau_\mathrm{s}}}{\sum_{j\in\mathcal{C}} W_j + M - \vartheta} && (15)
\end{aligned}$$

for all $M \in \mathbb{R}$ with $\sum_{j\in\mathcal{C}} W_j + M - \vartheta > 0$.

We first prove continuity for the argument of the logarithm by showing that $\forall \epsilon > 0$, $\exists \delta > 0$ such that $\|t_j - \tilde{t}_j\| < \delta$ with $j \in [0, N]$, $\|W_j - \tilde{W}_j\| < \delta$ with $j \in [0, N-1]$[1], and $\|T - \tilde{T}\| < \epsilon$. Using

[1]Note that W_N and $\tilde{W}_N$ cannot cause a switch between the two causal sets.

Eq. (15), we have:

$$\|T - \tilde{T}\| \tag{16}$$

$$= \left\| \frac{\sum_{j\in\mathcal{C}} W_j e^{t_j/\,\tau_\mathrm{s}} + \sum_{j\in\mathcal{C}} \tilde{W}_j e^{t_N/\,\tau_\mathrm{s}} + \tilde{W}_N e^{t_N/\,\tau_\mathrm{s}} - \sum_{j\in\mathcal{C}} W_j e^{t_N/\,\tau_\mathrm{s}}}{\sum_{j\in\mathcal{C}} \tilde{W}_j + \tilde{W}_N - \vartheta} \right.$$

$$\left. \frac{-\sum_{j\in\mathcal{C}} \tilde{W}_j e^{\tilde{t}_j/\,\tau_\mathrm{s}} - \tilde{W}_N e^{\tilde{t}_N/\,\tau_\mathrm{s}}}{\sum_{j\in\mathcal{C}} \tilde{W}_j + \tilde{W}_N - \vartheta} \right\| \tag{17}$$

$$\leq \frac{1}{\epsilon_\vartheta} \Bigg(\|\tilde{W}_N\| \cdot \|e^{\tilde{t}_N/\,\tau_\mathrm{s}} - e^{t_N/\,\tau_\mathrm{s}}\| + \sum_{j\in\mathcal{C}} \|W_j\| \cdot \|e^{t_j/\,\tau_\mathrm{s}} - e^{\tilde{t}_j/\,\tau_\mathrm{s}}\|$$

$$+ \|W_j - \tilde{W}_j\| \cdot \|e^{\tilde{t}_j/\,\tau_\mathrm{s}} - e^{t_N/\,\tau_\mathrm{s}}\| \Bigg) . \tag{18}$$

In the first step, we used Eq. (15) with $M = \sum_{j\in\mathcal{C}}(\tilde{W}_j - W_j) + \tilde{W}_N$, which leads to both T and $\tilde{T}$ having the same denominator. Furthermore, we added the term $\sum_{j\in\mathcal{C}} W_j e^{\tilde{t}_j/\,\tau_\mathrm{s}} - \sum_{j\in\mathcal{C}} W_j e^{\tilde{t}_j/\,\tau_\mathrm{s}}$ in the numerator. In the next step, we used $\frac{1}{\epsilon_\vartheta} \geq \frac{1}{\sum_{j\in\mathcal{C}} \tilde{W}_j + \tilde{W}_N - \vartheta}$, and applied the triangle inequality several times. Using $\|\tilde{W}_j\| \leq \bar{W}\ \forall j \in [0, N]$, $\|e^{\tilde{t}_j/\,\tau_\mathrm{s}} - e^{t_N/\,\tau_\mathrm{s}}\| \leq \|1 - C\|$ with $C = e^{t_\mathrm{max}/\,\tau_\mathrm{s}}$, and the mean value theorem for the exponential function, we then obtain:

$$\|T - \tilde{T}\| \leq \frac{C}{\epsilon_\vartheta \tau_\mathrm{s}} \left(\sum_{j\in\mathcal{C}'} \bar{W} \|\tilde{t}_j - t_j\| + \sum_{j\in\mathcal{C}} \frac{\tau_\mathrm{s}\|1 - C\|}{C} \|\tilde{W}_j - W_j\| \right) . \tag{19}$$

Choosing $\|\tilde{W}_j - W_j\| < \delta_W$ with $\delta_W = \frac{\epsilon_\vartheta}{2N\|1-C\|} \cdot \epsilon$ and $\|\tilde{t}_j - t_j\| < \delta_t$ with $\delta_t = \frac{\epsilon_\vartheta \tau_\mathrm{s}}{C \cdot \bar{W} \cdot 2(N+1)} \cdot \epsilon$, we arrive at

$$\|T - \tilde{T}\| < \epsilon\,. \tag{20}$$

The proof concludes by setting $\delta = \min(\delta_W, \delta_t)$. Continuity of the spike times then follows from the fact that the concatenation of continuous functions is again a continuous function.

Here we assumed that the neighbouring causal set $\mathcal{C}'$ has the property $\sum_{j\in\mathcal{C}'} \tilde{W}_j - \vartheta > 0$. If this is not the case, then at least one more input neuron with spike time $t^* = \min_x\{t_x \mid x \in \mathcal{K} \setminus \mathcal{C}'\}$ (with $t^* > t$) has to be added to the causal set until the condition holds again. Since the new output spike time has to be larger than t^*, its value jumps and is therefore not continuous when passing between causal pieces.

A.4.2 Lipschitz constants

To improve readability, we drop the layer and output neuron indices in the following. Within a causal piece $\mathcal{C}$, the causal set does not change and the output spike time t^* (Eq. (5)) is a composition of continuous and differentiable functions, and is therefore also continuous and differentiable. Hence, we estimate the Lipschitz constant by bounding the first derivative of the output spike time t^*.

Let $\mathcal{C}$ be a causal set with corresponding input spike times $t_0, ..., t_{N-1}$ for $N \in \mathbb{N}$, weights $W_0, ..., W_{N-1}$, and output spike time t^*. As in the previous subsection, we assume an upper bound for the absolute value of the weights, i.e., $\exists \bar{W} > 0$ such that $\forall \omega \in \{W_0, ..., W_{N-1}\}$, $\|\omega\| \leq \bar{W}$. Moreover, we assume that all spike times are larger or equal to 0, and we choose a $\delta > 0$ such that $\delta \leq \sum_j W_j - \vartheta$.

We first calculate the Lipschitz constant with respect to input spike times:

$$\left\| \frac{\partial t^*}{\partial t_k} \right\| = \left\| \frac{\partial}{\partial t_k} \tau_\mathrm{s} \ln \left(\frac{\sum_{j\in\mathcal{C}} W_j e^{t_j/\,\tau_\mathrm{s}}}{\sum_{j\in\mathcal{C}} W_j - \vartheta} \right) \right\| \tag{21}$$

$$= e^{-t^*/\,\tau_\mathrm{s}} \left\| \frac{W_k e^{t_k/\,\tau_\mathrm{s}}}{\sum_j W_j - \vartheta} \right\| \tag{22}$$

$$\leq \frac{\bar{W}}{\delta}\,, \tag{23}$$

where we used that $e^{(t_k-t^*)/\ \tau_\mathrm{s}} \leq 1$ since $t^* \geq t_k$ by definition.

For weights, we get:

$$\left\| \frac{\partial t^*}{\partial W_k} \right\| = \left\| \frac{\partial}{\partial W_k} \tau_\mathrm{s} \ln \left(\frac{\sum_{j \in \mathcal{C}} W_j e^{t_j/\ \tau_\mathrm{s}}}{\sum_{j \in \mathcal{C}} W_j - \vartheta} \right) \right\| \tag{24}$$

$$= \tau_\mathrm{s} e^{-t^*/\ \tau_\mathrm{s}} \left\| \frac{e^{t_k/\ \tau_\mathrm{s}}}{\sum_j W_j - \vartheta} - \frac{\sum_{j \in \mathcal{C}} W_j e^{t_j/\ \tau_\mathrm{s}}}{(\sum_j W_j - \vartheta)^2} \right\| \tag{25}$$

$$= \tau_\mathrm{s} e^{-t^*/\ \tau_\mathrm{s}} \left\| \frac{e^{t_k/\ \tau_\mathrm{s}} - e^{t^*/\ \tau_\mathrm{s}}}{\sum_j W_j - \vartheta} \right\| \tag{26}$$

$$= \tau_\mathrm{s} \left\| \frac{e^{(t_k-t^*)/\ \tau_\mathrm{s}} - 1}{\sum_j W_j - \vartheta} \right\| \tag{27}$$

$$\leq \frac{\tau_\mathrm{s}}{\delta}\,, \tag{28}$$

where we used that $0 \leq e^{(t_k-t^*)/\ \tau_\mathrm{s}} \leq 1$ by definition, and hence $\|e^{(t_k-t^*)/\ \tau_\mathrm{s}} - 1\| \leq 1$.

Thus, for a causal piece $\mathbb{P}_\mathcal{C} \subseteq \mathbb{R}^{d \times d}$, where $d \in \mathbb{N}$ is the dimension of the input, and $a, b \in \mathbb{P}_\mathcal{C}$ we have:

$$\|t(a) - t(b)\|_{L^\infty(\mathbb{P}_\mathcal{C})} \leq 2|\mathcal{C}| \max\left(\frac{\bar{W}}{\delta}, \frac{\tau_\mathrm{s}}{\delta} \right) \|a - b\|_{L^\infty(\mathbb{P}_\mathcal{C})} \tag{29}$$

where $L_{\mathbb{P}_\mathcal{C}} = 2|\mathcal{C}| \max\left(\frac{\bar{W}}{\delta}, \frac{\tau_\mathrm{s}}{\delta} \right)$ is the Lipschitz constant of causal piece $\mathbb{P}_\mathcal{C}$ with causal set $\mathcal{C}$, and $|\mathcal{C}|$ is the number of elements in the causal set.

A.4.3 Proof of Theorem 1

To improve readability, we drop the layer indices in the following. First, we recapitulate the following theorem which holds, for example, for ReLU neural networks [24] (Theorem 2)[2]:

Theorem 4 *Let $-\infty < a < b < \infty$, $f \in C^3([a,b])$ and f is not affine. Then there exists a constant $c > 0$ that only depends on $\int_a^b \sqrt{|f''(x)|}\mathrm{d}x$ so that*

$$\|\psi - f\|_{L^\infty([a,b])} > c \cdot p^{-2} \tag{30}$$

for all piecewise linear ψ with $p \in \mathbb{N}$ number of linear pieces.

Eq. (5) can be written as a piecewise linear function by substituting $T_i = e^{t_i/\tau_\mathrm{s}}$ [5], leading to:

$$T_i = \frac{1}{\sum_{j \in \mathcal{C}_i} W_{ij} - \vartheta} \cdot \sum_{k \in \mathcal{C}_i} W_{ik} T_k\,. \tag{31}$$

As input, we provide $t_x = \tau_\mathrm{s} \ln\left(\frac{e^{a/\tau_\mathrm{s}}(b-x) + e^{b/\tau_\mathrm{s}}(x-a)}{b-a} \right)$ with $x \in [a,b]$ and $-\infty < a < b < \infty$. An nLIF neural network $\Psi(x)$ using this substitution is then a composition of piecewise linear functions, and hence also itself a piecewise linear function of x. In this case, Theorem 4 applies to Ψ. The output of an equivalent nLIF network Φ without substitution is given by $\Phi = \tau_\mathrm{s} \ln \Psi$, i.e., we only apply the logarithm to the final output and scale by τ_s. This can be used to derive Theorem 1:

$$\|\Phi - g\|_{L^\infty([a,b])} = \tau_\mathrm{s} \left\| \ln \Psi - \ln\left(e^{g/\ \tau_\mathrm{s}} \right) \right\|_{L^\infty([a,b])}\,, \tag{32}$$

$$\geq \frac{\tau_\mathrm{s}}{\zeta} \|\Psi - e^{g/\ \tau_\mathrm{s}}\|_{L^\infty([a,b])}\,, \quad \text{with} \quad \zeta = \max\left[\max_x(\Psi(x)), \max_x(e^{g(x)/\ \tau_\mathrm{s}}) \right]\,, \tag{33}$$

$$> \frac{c}{\zeta} p^{-2}\,, \quad \text{with} \quad c > 0 \ \text{ depending only on } \ \tau_\mathrm{s} \int_a^b \sqrt{\left| \frac{\mathrm{d}^2}{\mathrm{d}x^2} e^{g(x)/\ \tau_\mathrm{s}} \right|} \mathrm{d}x\,, \tag{34}$$

[2] See also [44], Theorem 6.2

where we applied the mean value theorem to arrive at Eq. (33) (i.e., we apply the mean value theorem to get rid of the logarithms) and Theorem 4 to arrive at Eq. (34). For the latter, we used the fact that $e^{g/\tau_s} \in C^3([a,b])$ is not affine, allowing us to apply Theorem 4 using $f = e^{g/\tau_s}$. Furthermore, we note that Φ and Ψ have the same number of causal pieces.

A.4.4 Random walks

We drop the layer and output neuron index notation used in the main text to clear up the notation. Assume we have a single neuron with N_0 inputs. Let $\mathcal{K} = \{j_1, ..., j_K\} \subseteq [1, N_0]$ with $1 \leq K \leq N_0$, let t_j be the input times and $W_j \in \mathbb{R}$ the corresponding weights, with $j \in [1, N_0]$. We denote by p_k^q the probability that the subset $\mathcal{K}$ is a causal set if weights $W_j \sim q$ are sampled from a distribution q.

For $\mathcal{K}$ to be a causal set, we have to check the two conditions mentioned in Section 2. The first condition is satisfied if

$$\sum_{i \in \mathcal{K}} W_i \geq \vartheta \,. \tag{35}$$

Assuming the weights are sampled from a random distribution, this can be viewed as a random walk with discrete steps and randomly sampled, continuous step sizes. The position of the random walk at step k is given by $S_k = \sum_{i=1}^{k} W_i$. In this framework, the first condition becomes the question of whether the random walk is above or equal to the threshold at step K, i.e., $S_K \geq \vartheta$.

The second condition – only spike times belonging to the causal set appearing before the output spike – can always be achieved by choosing inputs the following way (this does not apply to deep networks):

1. Set $t_j = c$ for $c \in \mathbb{R}$ and $j \in \{j_\ell, ..., j_K\}$.

2. Since condition 1 is satisfied, use Eq. (5) to calculate the output spike time t with $\mathcal{K}$ as the causal set.

3. Set $t_j > t$ for $j \in \{j_\ell, ..., j_K\}$.

This way, any subset that suffices the first condition (sum of weights above threshold) is a valid causal set. Since we can choose inputs arbitrarily for a single nLIF neuron, p_k^q is identical to the probability of the random walker to be above threshold at step k.

The values of p_k^q are lower bounded by the first-passage-time distribution of the random walk. That's because the number of trajectories being above or equal to the threshold at step k is lower-bounded by the number of trajectories that cross the threshold for the first time at step k.

A.4.5 Proof of Theorem 2

Let $N \in \mathbb{N}$ be the number of inputs of a single nLIF neuron. We define $S_n = \sum_{i=1}^{n} W_i$ as the cumulative sum of weights $W_i \in \mathbb{R}$ with $S_0 = 0$ and $0 \leq n \leq N$. For the proof, we first note that $p_n^q \geq p_{\mathrm{FPT}}(n)$, where $p_{\mathrm{FPT}}(n) = p(S_n \geq \vartheta, S_{n-1} < \vartheta, S_{n-2} < \vartheta, ..., S_1 < \vartheta)$ is the first-passage-time distribution (at step n) for a random walk with discrete steps and random continuous step sizes ($W_j \sim q$), see Section A.4.4.

In the assumed limit, the survival probability, i.e., not passing the threshold until step $n+1$, is given by the Sparre Andersen theorem [38, 39]:

$$Q(n) = p(S_n < \vartheta, S_{n-1} < \vartheta, ..., S_1 < \vartheta) = \frac{1}{2^{2n}} \binom{2n}{n} \,. \tag{36}$$

The first-passage-time probability for step $n+1$ is obtained by taking the difference of survival probabilities:

$$p_{\mathrm{FPT}}(n+1) = Q(n) - Q(n+1) \tag{37}$$

$$= \frac{1}{2^{2n}}\binom{2n}{n} - \frac{1}{2^{2n+2}}\binom{2n+2}{n+1} \tag{38}$$

$$= \frac{1}{2^{2n+1}}\binom{2n}{n}\left[2 - \frac{(2n+2)(2n+1)}{2(n+1)(n+1)}\right] \tag{39}$$

$$= \frac{1}{2^{2n+1}}\binom{2n}{n}\left[2 - \frac{(2n+1)}{(n+1)}\right] \tag{40}$$

$$= \frac{1}{2^{2n+1}}\binom{2n}{n}\frac{1}{n+1} \tag{41}$$

$$= \frac{C_n}{2^{2n+1}}\,, \tag{42}$$

with the Catalan number $C_n = \frac{1}{n+1}\binom{2n}{n}$. Using a lower bound for the Catalan number [46], we get:

$$p^q_{n+1} \geq p_{\mathrm{FPT}}(n+1) \geq \frac{1}{2(n+1)\sqrt{\pi \cdot \left(n+\frac{1}{3}\right)}}\,. \tag{43}$$

This expression is monotonically decreasing, hence it reaches its minimum value at $n = N-1$:

$$p^q_{n+1} \geq \frac{1}{2N\sqrt{\pi \cdot \left(N-\frac{2}{3}\right)}}\,. \tag{44}$$

Using this, we can estimate the number of causal pieces:

$$\eta^q = \sum_{k=1}^{N}\binom{N}{k}p^q_k \tag{45}$$

$$\geq \sum_{k=1}^{N}\binom{N}{k}p_{\mathrm{FPT}}(k) \tag{46}$$

$$\geq \frac{1}{2N\sqrt{\pi \cdot \left(N-\frac{2}{3}\right)}} \cdot \sum_{k=1}^{N}\binom{N}{k} \tag{47}$$

$$= \frac{2^N - 1}{2N\sqrt{\pi \cdot \left(N-\frac{2}{3}\right)}}\,. \tag{48}$$

A.4.6 Number of pieces

For a single nLIF neuron, the number of pieces is obtained combinatorically: given N inputs to the neuron, we can create $\binom{N}{k}$ different subsets with k entries from these neurons. We denote by p^q_k the probability that, if weights are sampled from a probability distribution q, a subset of k inputs forms a causal set. The total number of causal pieces is then obtained by summing up the contributions of subsets of different length:

$$\eta = \sum_{k=1}^{N}\binom{N}{k}p^q_k\,. \tag{49}$$

The upper bound is obtained by using $p^q_k \leq 1$ for all k, and therefore $\eta \leq \sum_{k=1}^{N}\binom{N}{k} = 2^N - 1$.

For deep networks, we first look at a 2-layer network with $\{N_1, N_2, 1\}$ neurons, where N_1 is the number of inputs to the network. Starting with the output neuron, we can construct a single causal piece as follows: first, we sample a set of r inputs. From the analysis for single nLIF neurons, we know that $\binom{N_2}{r}p^{q_2}_r$ such sets exist. Next, we have to estimate the number of pieces of the r selected

input neurons, which are all given by $\eta_1 = \sum_{k=1}^{N_1} \binom{N_1}{k} p_k^{q_1}$. However, the causal piece of the output neuron changes if any of its r selected input neurons change their causal set. Thus, the number of pieces is given by $\binom{N_2}{r} p_r^{q_2} \eta_1^r$ – assuming the best case where the pieces of the output neuron are maximally split up by the input neurons. The total number is then given by:

$$\eta_2 = \sum_{r=1}^{N_2} \binom{N_2}{r} p_r^{q_2} \eta_1^r \,. \tag{50}$$

More generally, we have:

$$\eta_n = \sum_{r=1}^{N_n} \binom{N_n}{r} p_r^{q_n} \eta_{n-1}^r \,, \tag{51}$$

for $0 < n \leq \ell$ and $\eta_0 = 1$, where ℓ is the number of layers. Using $p_r^{q_n} \leq 1$ for all n and r and the binomial formula, we get:

$$\eta_n \leq (1 + \eta_{n-1})^{N_n} - 1 \,. \tag{52}$$

Defining $x_n = \eta_n + 1$ leads to $x_n \leq x_{n-1}^{N_n}$. Starting with $n = \ell$ until we arrive at $n = 1$, we then get:

$$x_l \leq 2^{\prod_{i=1}^{\ell} N_i} \tag{53}$$

$$\leq 2^{N^\ell} \,, \tag{54}$$

with $N = \max\{N_1, N_2, ..., N_\ell, 1\}$. Substituting back, this results in $\eta_l \leq 2^{N^\ell} - 1 < 2^{N^\ell}$.

A.5 Simulation details

In all simulations using nLIF neurons, we use $\tau_\mathrm{s} = 0.5$ and $\vartheta = 1$ unless stated otherwise. For LIF neurons, our standard parameters are $\tau_\mathrm{m} = 0.5$ and $\tau_\mathrm{s} = 0.25$. To implement deep learning models, we used pyTorch [47]. If a neuron does not spike, we artificially set its output spike time to a very late time that does not affect other neurons. In this work, we use $t_\text{late} = 5$. Furthermore, linear readout layers ignore any spike times $\geq t_\text{late}$, i.e., outputs are not affected by dead neurons.

Inputs are presented as single spike times to the neural network. In case of image data, we translate pixel values to spike times as follows: first, we rescale the pixel values to $[0, 1]$. Each data sample x_i is then translated into a spike time latency $1 - x_i$. For example, in case of MNIST and Fashion MNIST, this way black pixels are mapped to early spike times, while white pixels are mapped to late spike times.

Simulations were run on the ASC-5 infrastructure. Larger simulations were run on A40 GPUs and AMD Zen3 CPUs (e.g., Fashion-MNIST and EuroSAT), and smaller ones on a NVIDIA Geforce RTX 4060 Laptop GPU (e.g., Yin-Yang). Individual simulations for Yin-Yang are quite short, taking on the order of minutes, while simulations involving Fashion-MNIST or EuroSAT take a few hours.

A.5.1 Optimising initialisations

To find optimised initialisation schemes, we use a simple evolutionary method: Starting with a list with four different sets for the initial parameters, $P \in \mathbb{R}^{4\times4}$, we perturb each set by adding a random value sampled from a normal distribution $\mathcal{N}(0, 0.1^2)$. We then use all eight sets of parameters to initialise nLIF neural networks with weights sampled from our chosen distribution (e.g., normal, lognormal, uniform). For each network, we use the Yin-Yang dataset (or any other method) to estimate the number of pieces. In this case, we sample the input space using a grid ($x \in [0, 1]$,y$\in [0, 1]$, 100 increments per dimension, constrained to the circular area). We then take the parameters that produced the four networks with the highest number of pieces and repeat this process, i.e., with using this new list as P. We stop if the number of pieces does not improve after $n \in \mathbb{N}$ loops.

For positive weights, we initialise weights using a lognormal distribution with mean $\alpha_0 \cdot n_{\ell-1}^{-\alpha_1}$ and standard deviation $\alpha_2 \cdot n_{\ell-1}^{-\alpha_3}$, or a uniform distribution $\mathcal{U}(v_0, v_0 + v_1)$ with $v_0 = \beta_0 \cdot n_{\ell-1}^{-\beta_1}$ and $v_1 = \beta_2 \cdot n_{\ell-1}^{-\beta_3}$. $n_{\ell-1}$ is the number of neuron projecting into layer ℓ. Through the above optimisation loop, we found $\alpha_0 = 1.29$, $\alpha_1 = 0.57$, $\alpha_2 = 0.85$, $\alpha_3 = 0.76$ and $\beta_0 = 0.70$, $\beta_1 = 0.25$, $\beta_2 = 0.80$, $\beta_3 = 0.47$. The final parameters for normal and uniform (with positive and negative values) are provided in the main text.

A.5.2 Details: Fig. 1

To initialise the networks, we use a normal distribution with the parameters found using evolutionary optimisation (see main text and Section A.5.1).

In panel A and C (left), the causal pieces of the output neuron of a network with $[40, 1]$ neurons is shown (both are the same image, but the one in C is a cut-out). We sample three random vectors $d_0 \sim \mathcal{N}(0,1)^{40}$, $d_1 \sim \mathcal{N}(0,1)^{40}$, $o \sim \mathcal{N}(0,1)^{40}$. The inputs I are then obtained by spanning the plane using $I(\alpha_0, \alpha_1) = o + \alpha_0 \cdot (d_0 - o) + \alpha_1 \cdot (d_1 - o)$. We use $\alpha_0 \in [0,1]$ and $\alpha_1 \in [0,1]$ and 400 increments per variable.

For panel C (right), we use the same setup, but for a network with $[40, 40, 1]$ neurons.

A.5.3 Details: Fig. 2

To obtain the results, we used Algorithm 1 (see Section A.5.8) to estimate the number of pieces of a single nLIF neuron with weights sampled from $\mathcal{N}\left(\mu, \sigma^2\right)$. We ran the algorithm for values of μ and σ ranging from 0 to 0.1 with increments 0.001. The maximum number of inputs was set to 100. For each initialisation, we sampled 10^4 weight vectors (per k) to estimate p_k^q.

A.5.4 Details: Fig. 3

The architecture of our shallow network is: 4 spiking input neurons, 30 spiking hidden neurons, and a non-spiking linear layer with 3 output neurons. Causal pieces are calculated for the hidden layer.

We perform two scans: one where we ensure that the initial number of pieces is < 200 and > 1 (scenario A), and where we ensure that the initial number is > 200 (scenario B). This is done to ensure that we have sufficiently many SNNs with low and high piece counts. For both scenarios, we perform 100 random experiments. In scenario A, the normal distributions used to initialise the nLIF neural networks, the mean and standard deviation were both sampled from a uniform distribution $\mathcal{U}(-\alpha, 1-\alpha)$ with $\alpha \in [0.1, 0.2, 0.3, 0.4]$ and $\mathcal{U}(0,1)$, respectively. In scenario B, we use $\mathcal{U}(-\alpha, 1-\alpha)$ with $\alpha \in [0.1, 0.2, 0.3]$ and $\mathcal{U}(0,1)$, respectively. For both scenarios, we continue sampling mean and standard deviation until the initial number of causal pieces is in the required range.

Each reported data point corresponds to one sampled distribution. We calculate the number of causal pieces using only the 5000 training samples. We used the data samples to create the causal piece plots (panels F and G; note that disconnected clusters with similar colours are still different causal pieces). Networks are trained using the Adam optimiser with a learning rate of 10^{-2} (no weight decay), batch size of 200, and 200 epochs. The best test accuracy is reported, calculated using 1000 test samples (default for Yin-Yang). As a loss function, we use a cross-entropy loss.

A.5.5 Details: Fig. 4

For Fashion-MNIST, the trained networks have $[28 \times 28, 200, 10]$ neurons, with the last layer being a standard linear readout layer.

We perform two scans: one where we ensure that the initial number of pieces is $< 4 \cdot 10^4$ and > 1 (scenario A), and where we ensure that the initial number is $> 4 \cdot 10^4$ (scenario B). For both scenarios, we perform 50 random experiments. In scenario A, the normal distributions used to initialise the nLIF neural networks, the mean was sampled from a uniform distribution $\mathcal{U}(-\alpha, 1-\alpha)$ with $\alpha \in [0.4, 0.5, 0.6, 0.7, 0.8]$ randomly selected. The standard deviation was sampled from $\mathcal{U}(0,1)$. In scenario B, we use $\mathcal{U}(-\alpha, 1-\alpha)$ with $\alpha \in [0.3]$ randomly selected. The standard deviation was sampled from $\mathcal{U}(0,1)$. Each reported data point corresponds to one sampled distribution. We calculate the number of causal pieces using only the 60000 training samples. Networks are trained using the Adam optimiser with a learning rate of 10^{-3} (no weight decay), batch size of 100, and 400 epochs. As a loss function, we use the cross-entropy loss. The best test accuracy is reported.

The setup for EuroSAT RGB is the same, but we use networks with $[16 \times 16 \times 3, 200, 10]$ neurons. Moreover, the learning rate is set to 10^{-2}, and scenario A requires that the number of pieces is $< 2 \cdot 10^4$ and > 1, while scenario B requires $> 2 \cdot 10^4$. In scenario B, we use $\mathcal{U}(-\alpha, 1-\alpha)$ with $\alpha \in [0.1, 0.2, 0.3, 0.4]$ randomly selected. The resolution of the EuroSAT images is decreased from $32 \times 32 \times 3$ to $16 \times 16 \times 3$, with no significant decrease in the highest achieved test accuracy.

All experiments for LIF are done similarly as for nLIF (for Yin-Yang, see Section A.5.4). Only for the nLIF experiments using Fashion-MNIST, we further ran 50 more simulations using the same initialisation scheme as in scenario A, filtering for SNNs with an initial piece count $> 1.8 \cdot 10^4$ and $< 5 \cdot 10^4$.

A.5.6 Details: Fig. 5

For each data point, we show results of 10 runs with different random seeds. To calculate the number of causal pieces, we used an enlarged dataset composed of points obtained from a grid within the data domain, i.e., we evaluated the input space $[0, 1]^2$ using a 400×400 grid, leading to 124980 points (only points within the circular area were used). We obtained qualitatively similar results using a 600×600 grid. In panel C, we show the results for a network with $[4, 20, 20, 20, 20, 20, 3]$ and $[4, 40, 40, 40, 40, 40, 3]$ neurons (10 runs with different seeds). In panel D, the number of pieces of the output layer are shown for lognormal initialisation and (line) shallow networks with $[40, 80, 160, 320, 400]$ neurons in the hidden layer, as well as (dotted) deep networks with $[1, 2, 4, 5, 8, 10]$ hidden layers with 40 neurons each. Again, the median over 10 runs with different random seeds is shown. For training, the same setup as described in Section A.5.4 was used.

A.5.7 Details: Fig. 6

Networks are initialised by sampling the weights either from a lognormal or uniform distribution, as described in Section A.5.1. To evaluate p_k^q, we again use the Monte Carlo approach described in Section A.5.8, with a similar setup as in Fig. 2. Panel B is created similarly as panel B in Fig. 5. To keep weights W positive, we apply a ReLU function to them in the forward function, $W \mapsto \max(0, W)$.

For Yin-Yang, we use a network of size $[4, 30, 3]$, with the last layer being a standard linear pyTorch layer. We train the networks using a batch size of 100, learning rate of 10^{-3}, 5000 epochs, and Adam optimiser without weight decay. The reference values (0.638 and 0.976) are taken from [40] (best value also for $[4, 30, 3]$ neurons). They further report an accuracy of 0.855 if only the upper layer is trained, which is also lower than the performance reached by our networks.

For MNIST, we use a network of size $[28 \times 28, 200, 100, 10]$, again with the last layer being a standard linear pyTorch layer. Pixel values are re-scaled to be in the range $[0, 1]$. Images are flattened and no image transformations are used during training. We train the networks using a batch size of 100, learning rate of 10^{-3}, 200 epochs, and Adam optimiser without weight decay. The best performance (0.9833) is taken from [48]. For the performance of a linear layer, we show 0.9277, as, e.g., reported in [49].

For EuroSAT, we use a network of size $[16 \times 16 \times 3, 200, 100, 10]$, again with the last layer being a standard linear pyTorch layer. Images are re-scaled to $16 \times 16 \times 3$ (3 colour channels), with pixel values re-scaled to be in the range $[0, 1]$. Furthermore, we apply random horizontal and vertical flips during training. Images are flattened before they are provided as input to the neural networks. We train the networks using a batch size of 100, learning rate of 10^{-2}, 1000 epochs, and Adam optimiser without weight decay. We found that the best performance of an MLP is similar to the one reached by random forests, which is 0.70. For the performance of linear models, we use the results achieved using logistic regression (0.40). We also reached 0.34 using nearest neighbor and 0.47 using decision trees.

A.5.8 Algorithms: Monte Carlo approach

In simulations, we use Algorithm 1 to calculate p_k^q, from which we calculate the improved upper bound derived in the main paper. A similar algorithm can be used to estimate p_k^q for a static weight vector (with unknown distribution q) by randomly sampling subsets from the vector (e.g., in case of the weights in a trained neural network).

A.5.9 Algorithms: counting pieces

Algorithm 2 is used to count the number of causal pieces for (i) neurons in a deep neural network, and (ii) per layer. To count the pieces, we start from the first layer and index the causal sets. For neurons in the first layer, the causal sets are just composed of the inputs that caused the spike ((Algorithm 3,

Algorithm 1 Monte Carlo estimate for perceptron

Require: Distribution q, number of samples $num_samples$, number of inputs num_inputs, threshold ϑ

$prob_set \leftarrow$ list of length num_inputs filled with 0. ▷ Probability that subset is a causal set.
for $causal_set_length = 1$ to num_inputs **do**
  for $sample_ID = 1$ to $num_samples$ **do**
    $W \leftarrow$ list of length $causal_set_length$ with values sampled from q
    $c \leftarrow causal_set_length - 1$
    $strong_enough \leftarrow \sum_{i=0}^{c} W_i \geq \vartheta$
    if $strong_enough$ is $True$ **then**
      $prob_set[causal_set_length] \leftarrow prob_set[causal_set_length] + 1$
    end if
  end for
  $prob_set[causal_set_length] \leftarrow prob_set[causal_set_length]$ / $num_samples$
end for
return $prob_set$

line 5). Each neuron's piece is given by the index we assign it (Algorithm 4). For neurons in deep layers, the causal set consists of both the indices of the inputs that caused it to spike, and the causal piece indices of these neurons (Algorithm 3, line 3). For layers (Algorithm 5), the causal set is given by the list of causal piece indices of all neurons in the layer. If any of these indices changes, the causal piece of the layer changes.

Algorithm 2 Transform causal sets (per neuron) to causal piece IDs

Require: Nested list with ordered causal sets, `sets`. Dimensions are: samples, layers, neurons.

$causal_set_to_ID \leftarrow$ empty dictionary
$causal_set_to_ID$[String([])] $\leftarrow -1$
$num_samples \leftarrow$ length($sets$)
$IDs \leftarrow$ list containing $num_samples$ empty lists
for $sample_id = 0$ to $num_samples - 1$ **do** ▷ Iterate over samples
  $sets_of_sample \leftarrow sets[sample_id]$
  for $layer_id = 0$ to length($sets_of_sample$) $- 1$ **do** ▷ Iterate over layers
    $layers \leftarrow sets_of_sample[layer_id]$
    Append empty list to $IDs[sample_id]$
    for each $causal_set$ in $layers$ **do** ▷ Turn causal set of every neuron to corresponding ID
      $cset_name \leftarrow$ ProcessCausalSet($causal_set, IDs, sample_id, layer_id$)
      $single_ID \leftarrow$ AssignID($cset_name, causal_set_to_ID$)
      Append $single_ID$ to $IDs[sample_id][layer_id]$
    end for
  end for
end for
return IDs

Algorithm 3 PROCESSCAUSALSET

Require: Causal set $causal_set$, List of causal set IDs IDs, Sample index $sample_id$, Layer index $layer_id$

if $layer_id > 0$ **then**
  $prev_layer_IDs \leftarrow IDs[sample_id][layer_id - 1]$
  $selected_IDs \leftarrow$ empty list
  for $index$ in $causal_set$ **do** ▷ Iterate over indices in causal set
    Append $prev_layer_IDs[index]$ to $selected_IDs$
  end for
  $cset_name \leftarrow$ String$([selected_IDs, causal_set])$
else
  $cset_name \leftarrow$ String$(causal_set)$
end if
if length$(causal_set) = 0$ **then**
  $cset_name \leftarrow$ String$([])$
end if
return $cset_name$

Algorithm 4 ASSIGNID

Require: Causal set name $cset_name$, Dictionary $causal_set_to_ID$

if $cset_name \notin$ keys$(causal_set_to_ID)$ **then**
  $causal_set_to_ID[cset_name] \leftarrow$ length$(causal_set_to_ID)$ ▷ For efficiency, one can also hash $cset_name$.
end if
return $causal_set_to_ID[cset_name]$

Algorithm 5 Get Causal Piece ID for Neural Network Layers

Require: `IDs`, List of dictionaries `layer_indices_dict` with length $num_layers - 1$

$piece_ID_layers \leftarrow$ empty list
for $sample_ID = 0$ to length$(IDs) - 1$ **do** ▷ Iterate over samples
  Append empty list to $piece_ID_layers$
  for $layer_ID = 0$ to length$(IDs[sample_ID]) - 1$ **do** ▷ Iterate over layers
    $lay_state \leftarrow$ String$(IDs[sample_ID][layer_ID])$
    if $lay_state \notin$ keys$(layer_indices_dict[layer_ID])$ **then**
      $layer_indices_dict[layer_ID][lay_state] \leftarrow$ length$(layer_indices_dict[layer_ID])$
    end if
    Append $layer_indices_dict[layer_ID][lay_state]$ to $piece_ID_layers[sample_ID]$
  end for
end for
return $piece_ID_layers$

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: All assumptions (theory for single-spike nLIF; single-spike nLIF and LIF simulations; used datasets) are stated in the abstract and introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in the Discussion section.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: All assumptions are always stated. Proofs are in the Appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All simulation details are documented in the Appendix. The main text states the most important details for understanding the simulation results. Code will be made publicly available with the final publication.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code will be made publicly available on Github for the final publication, with a link to the repository in the camera-ready version.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: These details are provided in the Appendix, or in the Results section if necessary.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The results reported in the paper are for many repetitions for different random seeds. Often, we show directly scatter plots showing all runs. Sometimes, we show median and quartiles (as shaded area), or mean and error bars.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We disclose the used compute resources in the Appendix (A.5). Furthermore, we provide an experiment measuring the required compute time of our introduced method (A.1.2).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: –

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: All experiments were conducted using publicly available and widely accepted benchmark datasets. No personal or sensitive data were used, and there are no known ethical or bias concerns associated with the datasets. The paper presents work whose goal is to

increase our theoretical understanding of deep neural networks. In particular, we provide theoretical results and tools for spiking neural networks – a brain-inspired neural network architecture that promises highly efficient hardware implementations for deep learning. Due to the strong focus on theory of our work, there is no immediate societal impact that has to be addressed in the paper.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This is a theory paper.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: –

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.

- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: A documented Python package with models and methods for reproducing the findings of the paper will be released with the camera-ready version of the paper.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: –

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: –

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: –

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.