A deep learning toolbox for stochastic stabilized supralinear networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

There continues to be a trade-off between the biological realism and performance of neural networks. Contemporary deep learning techniques allow neural networks to be trained to perform challenging computations at (near) human-level, but these networks typically violate key biological constraints. More detailed models of biological neural networks can incorporate many of these constraints but typically suffer from subpar performance and trainability. Here, we narrow this gap by developing a deep learning toolbox for training a canonical model of cortical neural circuits, the stabilized supralinear network (SSN), that in previous work had to be constructed manually or trained with undue constraints. SSNs are particularly challenging to train for the same reasons that make them biologically realistic: they are characterized by strongly-connected excitatory cells and expansive firing rate non-linearities that together make them prone to dynamical instabilities unless stabilised by appropriately tuned recurrent inhibition. Our method avoids such instabilities by using a small network and an iterative fixed point method for initialisation and by gradually increasing network size via the dynamics-neutral addition of neurons during training. We demonstrate the success of our method by training an SSN on the challenging task of performing amortized Markov chain Monte Carlo-based inference under a Gaussian scale mixture generative model of natural image patches with a rich and diverse set of basis functions – something that was not possible with previous methods. On a simpler task, we achieve performance that is about an order of magnitude better than previous approaches. These results open the way to training realistic cortical-like neural networks on challenging tasks at scale.

1 Introduction

Biological neural network models of the brain have been proposed and studied long before the advent of machine learning [1–3]. The primary goal of these models was to understand some of the fundamental network mechanisms employed by the brain but were not capable of actually achieving the brain's performance on its wide range of complex computations. More recently, significant progress has been made in the related field of artificial neural networks, especially at achieving competitive performance in a multitude of image, language and speech-related tasks. While biologically-motivated models may occasionally find success in these settings [4, 5], recent advances in state-of-the-art models have mainly been artificial [6–8]. This has led to investigations on improving their potential biological-plausibility [9–11], usually at a cost on performance [12].

In systems neuroscience, neural networks have also been trained on laboratory tasks employed in typical experiments [13, 14]. These approaches provided important insights into the contributions of experimentally found macroscopic neural dynamics and representations to the successful performance

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

of such tasks. For example, they described the presence of line attractors during context-dependent evidence accumulation [15], mixed task selectivity for flexible task performance [16], and grid cell responses for path integration [17] and spatial localization [18]. However, these networks did not incorporate some of the most salient constraints on the detailed organization of cortical circuits. For example, they were purely feedforward [19], utilized neuronal transfer functions that were either outright saturating (e.g. tanh) or at best lacked the superlinear characteristics of cortical neurons (as in ReLUs, or the rectified softplus) [19, 13, 16, 20, 15], had noiseless dynamics [14], but no separation of excitatory (E) and inhibitory (I) cells [19, 20, 15], and sometimes even employed artificial gating mechanisms such as LSTMs [17]. Due to these properties, these networks remained abstracted away from biological neurons in key aspects, and hence offered limited insight into the neuron-level mechanisms that drive their computations.

To bring the modelling of neural circuit mechanisms underlying challenging computations at (or near) single-cell resolution within reach, we develop a novel toolbox to train an experimentally supported neural network model, the stabilized supralinear network (SSN) [21], on challenging tasks. Critically, the SSN satisfies many of the key constraints of cortical circuits: it has separate but recurrent E and I populations, expansive (rectified power-law) single neuron non-linearities, realistic single-neuron time constants, and no reliance on artificial gating mechanisms. In its original form, the SSN continues a venerable tradition of handcrafted E-I networks, whose dynamics have been extensively studied [22, 23], and that helped reveal key consequences of the cortex operating in a balanced E-I regime [3, 24–26]. In particular, the SSN accounts for the experimentally-observed effects of stimulus tuning, sublinear response summation and surround suppression of neural responses in sensory cortices [27, 28], as well as oscillations, bistable and persistent responses [29]. Furthermore, when extended to include noise, the resultant stochastic SSN produces realistic, stimulus-modulated population-wide patterns of noise variability [30]. This combination of features makes the SSN an ideally suited candidate for developing cortical-like models of neural computations.

However, the same features that make the SSN an attractive substrate for its biological realism, also make it particularly challenging to train it. This is because, in SSNs, E cells are typically strongly connected with one-another in order to implement non-trivial nonlinear transformations of their inputs, such as divisive normalization, which underlie many of the experimentally observed phenomena for which they account. In addition, single neuron non-linearities are expansive to reflect the experimental finding that cortical neurons almost exclusively use the convex (non-saturating) part of their firing rate non-linearities under physiological conditions, including for the strongest stimuli [31] (even if they can be trivially driven to saturation by artificial stimuli, due to the refractory period of the mechanism generating action potentials). Thus, stable dynamics cannot be achieved by relying on saturating single neuron non-linearities as in some previous work [32]. These properties make SSNs particularly susceptible to dynamical instabilities resulting in run-away excitation. This is usually prevented by appropriately tuned recurrent inhibition from I cells, requiring careful tuning of parameters [21, 29]. At the same time, neural noise can further exacerbate the instability problem as even if appropriate E-I balance is achieved on average, a momentary imbalance due to noise may still snowball into a full-fledged explosion of activities. This makes the training of such networks difficult: as the training procedure explores different parts of the parameter space, it will likely visit parameter settings which disrupt E-I balance and thus lead to instability and an immediate failure of training. Indeed, in the few cases in which the training of SSNs was attempted, either noiseless neurons were used [33, 34], or the network was so heavily underparameterized (15 parameters for a total of 100 neurons) that it substantially limited its expressivity [35].

Here, we face this challenge by developing an approach that allows us to employ deep learning algorithms for task optimization while still staying within an inhibition-stabilized parameter regime of the SSN. To demonstrate the superiority of our approach in a directly comparable way, we train an SSN to perform the same probabilistic inference task as an earlier attempt for training stochastic SSNs [35]. This task requires the SSN to act as the recognition model for a probabilistic generative model of natural image patches, the Gaussian scale mixture generative model (GSM) [36, 37], by performing amortized approximate Markov chain Monte Carlo-based inference. That is, in response to an input (observation), the network needs to produce trajectories in the state space of its neurons (encoding latent variables of the GSM) such that the distribution of multi-neuron response patterns sampled by these trajectories approximately matches the joint posterior distribution of the GSM for corresponding input. Notably, the network needs to achieve this with a single set of parameters (prominently, synaptic weights) for a large number of inputs. This is a challenging task because

the objective requires the network to maintain finite levels of variability while modulating the co-variability of its neurons in a stimulus-dependent way, which is only possible if stochasticity is neural noise is not suppressed and recurrent connections are sufficiently strong – and thus the network is pushed towards instability. We show that our approach is able to successfully train SSNs that have three orders of magnitude more parameters than what was previously possible to optimise, which we will refer to as generalized stochastic SSNs (gsSSNs). Our trained gsSSN performs accurate inference under a GSM with a set of basis functions that is sufficiently rich for allowing high-qualtiy reconstructions of CIFAR-10 [38] images. This GSM represents an impossible target for previous, heavily constrained approaches for training SSNs. When training our gsSSN for a more constrained GSM, for which previous approaches were appropriate, our method still achieves nearly an order of magnitude better performance. These results validate our approach and open the way to training SSNs for a large variety of different tasks, and to using the resulting networks to study the circuit mechanisms of neural computations at a single cell resolution.

2 Background

2.1 Gaussian scale mixture model

The GSM [36, 37] is proposed as an internal generative model of natural image patches in V1 (Fig 1, left). An image $\boldsymbol{x}$ is generated by,

$$\boldsymbol{x} = z\boldsymbol{A}\boldsymbol{y} + \boldsymbol{\eta}_x \tag{1}$$

for projective fields $\boldsymbol{A}$, latent activations $\boldsymbol{y}$, global scalar contrast level z and pixel noise $\boldsymbol{\eta}_x$. The projective fields are 2-D Gabor filters, each of which is characterized by four parameters,

$$g(x, y; x_0, y_0, \phi, \sigma) = \exp\left(-\frac{x'^2 + y'^2}{2\sigma^2}\right)\cos\left(\frac{2.5}{\sigma}x'\right) \tag{2}$$

where x' and y' are pixel coordinates rotated by ϕ. The parameters of $\boldsymbol{A}$ are summarized as $\boldsymbol{\theta}_A = \{\boldsymbol{x}_0, \boldsymbol{y}_0, \boldsymbol{\phi}, \boldsymbol{\sigma}\}$. An ideal observer, when presented with an image $\boldsymbol{x}$, computes a posterior distribution over $\boldsymbol{y}$ and z by incorporating their respective priors $p(\boldsymbol{y})$ and $p(z)$,

$$p(\boldsymbol{\eta}_x) = \mathcal{N}(\boldsymbol{\eta}_x; 0, \sigma_x^2\boldsymbol{I}) \tag{3}$$

$$p(\boldsymbol{y}) = \mathcal{N}(\boldsymbol{y}; 0, \boldsymbol{C}) \tag{4}$$

$$p(z) = \Gamma(z; \alpha, \beta) \tag{5}$$

after which the contrast level z is marginalized, resulting in a posterior distribution over $\boldsymbol{y}$,

$$p(\boldsymbol{y}|\boldsymbol{x}_i) = \mathbb{E}\left[p(z|\boldsymbol{x}_i)p(\boldsymbol{y}|z, \boldsymbol{x}_i)\right]_z \tag{6}$$

2.2 Stochastic supralinear stabilized networks

The SSN is a canonical model of cortical circuits that is able to explain a variety of neural response characteristics [21, 27] through its straightforward dynamics (Fig 1, right), described by,

$$\boldsymbol{\tau}\frac{d\boldsymbol{u}}{dt} = -\boldsymbol{u} + f(\boldsymbol{h}) + \boldsymbol{W}\boldsymbol{r} + \boldsymbol{\eta} \tag{7}$$

where neurons are characterized by their membrane potentials $\boldsymbol{u}$ and time constants $\boldsymbol{\tau}$. An input image $\boldsymbol{x}$ is converted into external input for the network through its receptive fields (which by construction is set to be the projective fields of the GSM), computed using its left pseudoinverse,

$$\boldsymbol{h} = (\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})^{-1}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{x} \tag{8}$$

while $f(\boldsymbol{h})$ represents the some non-linear transformation of the external input,

$$f : h_i \to \theta_1(h_i + \theta_2)^{\theta_3} \tag{9}$$

for parameters $\boldsymbol{\theta}_h = \{\theta_1, \theta_2, \theta_3\}$. $\boldsymbol{W}$ denotes the network weights, which transmit neuron firing rates $\boldsymbol{r}$, given by a threshold expansive non-linearity,

$$r_i = k\lfloor u_i \rfloor_+^n \tag{10}$$

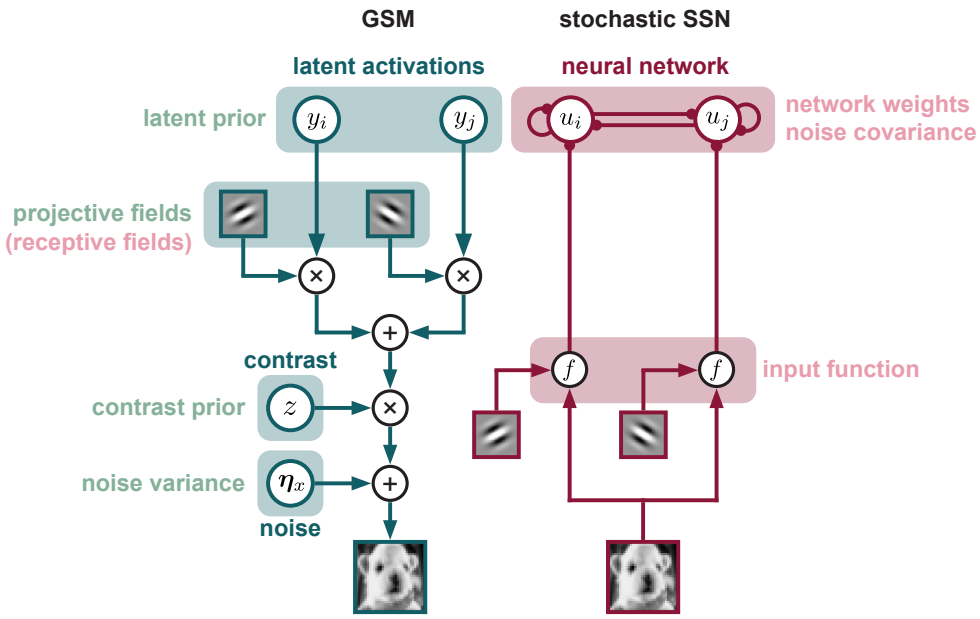

Figure 1: **Overview of the training objective.** There are two main components to the model. The first component is the main network itself, known as the stochastic stabilized supralinear network (stochastic SSN, right). Note that the network comprises of 50 excitatory and 50 inhibitory neurons (not depicted). The other component is the internal generative model under which the network is trained to perform sampling-based probabilistic inference, which in this case is the Gaussian scale mixture model (GSM, left). We optimize the SSN to perform this computation by minimizing the mean-squared errors between the first two moments of GSM latents $\boldsymbol{y}$ and SSN membrane potential responses $\boldsymbol{u}$, as described in (11). Trainable variables are shown on the sides of the respective components.

for $n > 1$ and constant k. Note that there is no built-in upper saturation to the model, which presents potential instability problems – excitations within the network can exponentially grow unbounded through mutual and self excitation. The network relies purely on inhibition to stabilize, and this therefore restricts the parameter space in which the network is stable.

The network distinctly comprises of excitatory and inhibitory neurons, which in turn imposes restrictions on the polarity of the elements of $\boldsymbol{W}$. Specifically, the columns of $\boldsymbol{W}$ must have the same sign. Finally, $\boldsymbol{\eta}$ represents noise, correlated across time with time constant τ_η, as well as across neurons with covariance $\boldsymbol{\Sigma}_\eta$.

2.3 Sampling-based probabilistic inference

The computational goal of the SSN is to perform sampling-based probabilistic inference under the GSM. In other words, given an observed image as input, the membrane potential response of the SSN, as determined by (7), represents statistical samples for the posterior distribution over the latent variables of the GSM, as computed by an ideal observer using (6). We implement this by optimizing the model with respect to the mean-squared errors between the first and second order moments of both distributions,

$$\mathcal{L} = \lambda_\mu \left\| \mathbb{E}[\boldsymbol{u}_{\text{SSN}}] - \boldsymbol{\mu}_{\text{GSM}} \right\|_F^2 + \lambda_\Sigma \left\| \mathbb{C}[\boldsymbol{u}_{\text{SSN}}] - \boldsymbol{\Sigma}_{\text{GSM}} \right\|_F^2 \tag{11}$$

for constant coefficients λ_μ and λ_Σ. Not all neurons are matched to GSM targets. We refer to the neurons involved in the cost function as coding neurons, while those that are not are called non-coding neurons.

3 Methods

The difficulty of training an SSN has resulted in simplifications to the computational goal in past attempts. For example, the GSM adopted in [35] consists of filters with fixed position and scale, and only differing in their orientations. This is mainly motivated by existing work on the analysis of ring SSNs [27] – SSNs whose receptive fields vary only in orientation. As such, training a ring SSN on an equivalently-defined ring GSM is logical and compatible. However, we seek to maximize the difficulty and realism of this task and instead train the SSN on a GSM that has been optimized for natural images with no convenient symmetries or simplifying shortcuts.

3.1 GSM optimization

Details of the optimization can be found in Appendix A of the supplementary material, but we will briefly describe the methodology here. We first optimize for a set of Gabor filters that captures the maximum amount of variability in the training data. We do this by minimizing the trial-averaged fraction of variance unexplained,

$$\hat{\boldsymbol{\theta}}_A = \min_{\boldsymbol{\theta}_A} \mathbb{E}\left[\frac{\mathbb{V}[\boldsymbol{x}_i - (\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})^{-1}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{x}_i]_x}{\mathbb{V}[\boldsymbol{x}_i]_x}\right]_i \tag{12}$$

The remaining parameters of the GSM, summarized as $\boldsymbol{\theta}_{\text{GSM}} = \{\boldsymbol{C}, \alpha, \beta, \sigma_x\}$, can then be concurrently optimized by maximizing the likelihood of observing the training images (or equivalently, minimizing negative log-likelihood),

$$\hat{\boldsymbol{\theta}}_{\text{GSM}} = \min_{\boldsymbol{\theta}_{\text{GSM}}} \left(-\log \mathbb{E}\left[\mathcal{N}(\boldsymbol{x}; 0, z^2\boldsymbol{A}\boldsymbol{C}\boldsymbol{A}^{\mathrm{T}} + \sigma_x^2\boldsymbol{I})\right]_z\right) \tag{13}$$

where $\mathbb{E}[\cdot]_z$ denotes taking expectation across all contrast levels.

3.2 Targeted SSN initialization

We next focus on building and training the SSN. Immediately, we are presented with the problem of initializing a stable network. This feat gets progressively harder to achieve as the size of the network increases (see Appendix B of the supplementary material for a review of stability problems in training of SSNs). Considering additionally the complexity of training a large network, we have opted to start off by initializing a smaller network. We then gradually add more neurons during optimization until the network is fully built.

We choose to start with a network of just one excitatory and one inhibitory neuron. Instead of randomly generating parameters, checking if the resultant network converges, and finally picking the best parameters with the lowest cost, we present an iterative algorithm that performs better. Assuming that the network converges, its stationary moments satisfy,

$$\mathbb{E}[\boldsymbol{u}] = \mathbb{E}[f(\boldsymbol{h})] + \boldsymbol{W}\mathbb{E}[\boldsymbol{r}] \tag{14}$$

We split the membrane potential vector into coding and non-coding components,

$$\mathbb{E}[\boldsymbol{u}] = \begin{pmatrix} \mathbb{E}[\boldsymbol{u}_c] \\ \mathbb{E}[\boldsymbol{u}_{nc}] \end{pmatrix} \tag{15}$$

The goal is to match $\mathbb{E}[\boldsymbol{u}_c]$ to $\boldsymbol{\mu}_{\text{GSM}}$. It is tempting to substitute $\mathbb{E}[\boldsymbol{u}_c] = \boldsymbol{\mu}_{\text{GSM}}$ and solve for the network parameters, but at this juncture, $\mathbb{E}[\boldsymbol{r}]$ is non-trivially dependent on $\boldsymbol{u}$ and cannot be evaluated. However, given an existing set of parameters, we can obtain a fixed-point numerical estimate of $\mathbb{E}[\boldsymbol{r}]$ and then use it to solve for a new set of parameters. This forms the basis of our iterative algorithm.

We begin by randomly initializing parameters $\boldsymbol{\theta}_{\text{SSN}} = \{\boldsymbol{W}, \boldsymbol{\theta}_h, \boldsymbol{\Sigma}_\eta\}$. We then simulate the full network to check for convergence. If it converges, also record its cost given by (11), $\mathbb{E}[\boldsymbol{u}_{nc}]$ and $\mathbb{E}[\boldsymbol{r}]$, where the latter two is used to generate a new set of parameters $\boldsymbol{\theta}^*_{\text{SSN}}$ by optimizing,

$$\boldsymbol{\theta}^*_{\text{SSN}} = \min_{\boldsymbol{\theta}_{\text{SSN}}} \left\| \begin{pmatrix} \boldsymbol{\mu}_{\text{target}} \\ \mathbb{E}[\boldsymbol{u}_{nc}] \end{pmatrix} - \mathbb{E}[f(\boldsymbol{h})] + \boldsymbol{W}\mathbb{E}[\boldsymbol{r}] \right\|_F^2 \tag{16}$$

Table 1: **Comparison of optimized stochastic SSNs**. For fair comparison, we build the same network described in [35] and optimize it using our approach (second row, denoted with *). The resultant network attains a fraction of the lowest mean-squared error cost previously achieved (which is normalized to 1). Although our main model is trained on different targets (third row), we scale the cost elements such that the targets have the same mean and variance, therefore allowing us to compare between two mean-squared error costs with different targets.

model	training size	parametrization	parameters	algorithm	cost [35]
[35]	5	circular Gaussian	15	Adam + ADF [39]	1
gsSSN*	5	symmetric circulant	211	Adam	**0.155**
gsSSN	50000	free	20003	Adam	**0.788**

after which it repeats – we simulate this new network and obtain new estimates $\mathbb{E}[\boldsymbol{u}_{nc}]^*$ and $\mathbb{E}[\boldsymbol{r}]^*$ and so on. It is also interesting to note that in the case where $n = 2$, we may make the approximation,

$$\mathbb{E}[r_i] \approx k\left(\mathbb{E}[u_i]^2 + \mathbb{V}[u_i]\right) \tag{17}$$

leading to a more efficient way of generating new parameters which includes target variances,

$$\boldsymbol{\theta}^*_{\text{SSN}} = \min_{\boldsymbol{\theta}_{\text{SSN}}} \left\| \begin{pmatrix} \boldsymbol{\mu}_{\text{target}} \\ \mathbb{E}[\boldsymbol{u}_{nc}] \end{pmatrix} - \mathbb{E}[f(\boldsymbol{h})] + \boldsymbol{W}\left(\begin{pmatrix} \boldsymbol{\mu}_{\text{target}} \\ \mathbb{E}[\boldsymbol{u}_{nc}] \end{pmatrix} \odot \begin{pmatrix} \boldsymbol{\mu}_{\text{target}} \\ \mathbb{E}[\boldsymbol{u}_{nc}] \end{pmatrix} + \begin{pmatrix} \text{diag}(\boldsymbol{\Sigma}_{\text{target}}) \\ \mathbb{V}[\boldsymbol{u}_{nc}] \end{pmatrix} \right) \right\|^2 \tag{18}$$

where $\text{diag}(\cdot)$ denotes the diagonal elements of the input matrix, and $\odot$ denotes element-wise product. In this version, only $\mathbb{E}[\boldsymbol{u}_{nc}]$ and $\mathbb{V}[\boldsymbol{u}_{nc}]$ need to be recorded, as all the other elements on the RHS are replaced by GSM targets.

3.3 Simultaneous network training and building

With a suitable initialization chosen, we begin training the network by backpropagation-through-time based on the initial cost function (11),

$$\boldsymbol{\theta}^*_{\text{SSN}} = \min_{\boldsymbol{\theta}_{\text{SSN}}} \mathcal{L} \tag{19}$$

Due to memory constraints, we randomly select 100 images from the training set to train with at every iteration. After some fixed number of training iterations, we add more neurons to the network until we attain our desired network size. Unfortunately, due to the non-linearities present in the SSN, there is no general analytical approach of doing so that guarantees stability.

We may still add neurons in such a way that it intuitively has a high chance of remaining stable. A simple but effective way of doing so is to simply duplicate an existing neuron in the network. Let neuron n be the cloning candidate. We split the weight matrix $\boldsymbol{W}$ into four parts, $\{\boldsymbol{W}_{(\neg n,\neg n)}, \boldsymbol{W}_{(\neg n,n)}, \boldsymbol{W}_{(n,\neg n)}, \boldsymbol{W}_{(n,n)}\}$, such that after duplicating the neuron, we update the network weights according to,

$$\begin{bmatrix} \boldsymbol{W}_{(\neg n,\neg n)} & \boldsymbol{W}_{(n,\neg n)} \\ \boldsymbol{W}_{(\neg n,n)} & \boldsymbol{W}_{(n,n)} \end{bmatrix} \rightarrow \begin{bmatrix} \boldsymbol{W}_{(\neg n,\neg n)} & (1/2)\boldsymbol{W}_{(n,\neg n)} & (1/2)\boldsymbol{W}_{(n,\neg n)} \\ \boldsymbol{W}_{(\neg n,n)} & (1/2)\boldsymbol{W}_{(n,n)} & (1/2)\boldsymbol{W}_{(n,n)} \\ \boldsymbol{W}_{(\neg n,n)} & (1/2)\boldsymbol{W}_{(n,n)} & (1/2)\boldsymbol{W}_{(n,n)} \end{bmatrix} \tag{20}$$

which will result in all neurons receiving the same inputs as before. Using this method, we may duplicate multiple neurons at once, or split a single neuron into more than two copies. In our training, we choose to duplicate an excitatory and an inhibitory neuron at every building step.

In the case where the weight matrix is symmetric circulant (such as in [35]), we propose a similar method of adding neurons that additionally preserves its structure. It requires specifically that an even number of neurons be added at the same time,

$$\text{symcirc}\left([w_0, w_1, \ldots w_n]\right) \rightarrow \text{symcirc}\left(\left[w_0, w_1, \ldots w_{n-1}, \frac{w_n}{2}, \frac{w_n}{2}\right]\right) \tag{21}$$

where $\text{symcirc}[\cdot]$ is a function that constructs a symmetric circulant matrix given an input row vector. On average, this method results in the same input entering each neuron assuming that the stimulus is presented uniformly across all orientations.

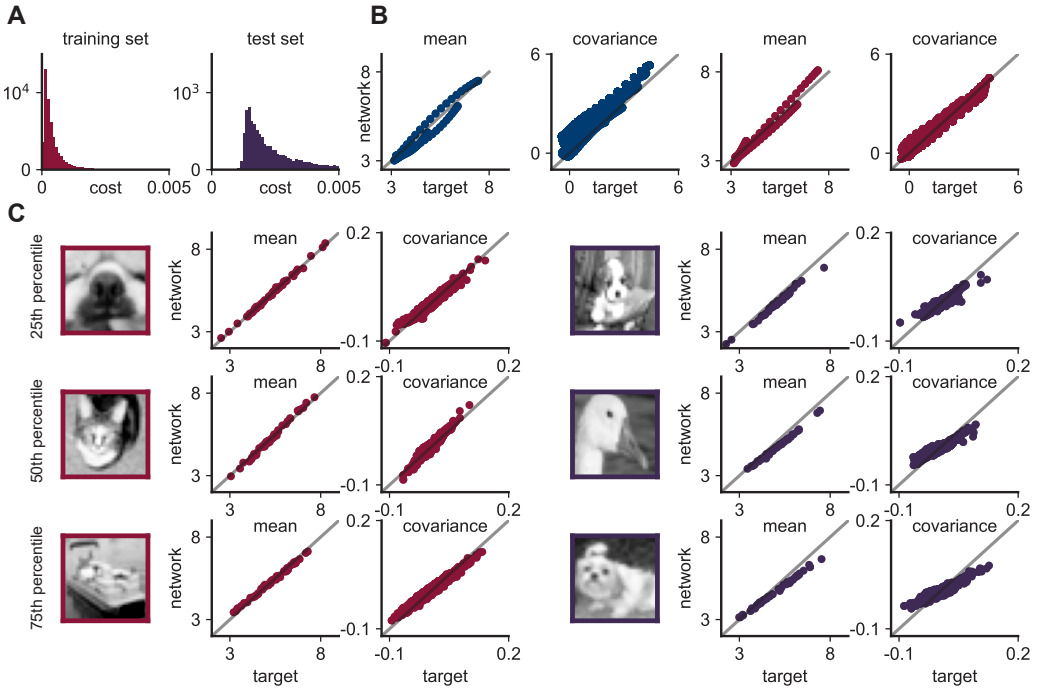

Figure 2: **Comparison of optimized stochastic SSNs. A.** Histogram of costs obtained on each of the 50000 training images (red) and 10000 test images (purple). **B.** Plots of simulated moments against target GSM moments for the SSN in [35] (blue) and a gsSSN that we trained under the same setup. The gsSSN is able to obtain a fraction of the cost of the original network due to more accurate matching of the covariance terms. **C.** Plots of input image and simulated moments against target moments. Each row depicts the plots for different percentiles of training (left) and test (right) costs. The optimized gsSSN is able to reasonably match the target moments in the training set, but is less accurate on the test set.

4 Results

4.1 Optimization on existing task

We first establish a direct comparison by training a gsSSN on an existing setup in [35]. We manage to obtain a mean-squared error cost that is almost an order of magnitude lower (Table 1, 2nd row). This can also be seen when comparing network moments to their targets (Fig 2B). Both the existing SSN and our gsSSN are able to match the target mean with similar accuracies, but the gsSSN is able to better match covariance terms to a significant degree. There are two key differences between how the two networks are trained. Firstly, the gsSSN is trained with an unbiased deep learning algorithm, while the existing model is trained using a biased semi-analytical method. Secondly, the gsSSN has more free parameters, and hence is able to better fit its targets.

4.2 gsSSN optimized for CIFAR-10

We next optimize a gsSSN on targets computed from the GSM that is optimized for CIFAR-10 images. In order to compute a quantity that can be directly compared with the mean-squared error costs from the previous model, we rescale our targets to have the same mean and standard deviation as the previous targets, and scale the network moments accordingly. We find that when we compute costs in this way, we are able to conclude that the gsSSN is able to obtain a better fit towards its own targets of 50000 natural images with a lower parameter-to-training-size ratio (Table 1,last row). Additionally, the moments are generally well-matched even at higher cost percentiles (Fig 2C). We also evaluate the costs on the test dataset (Fig 2A,C), and find that they are higher than the cost of the training set. This suggests some degree of over-fitting, which would be detrimental in a typical

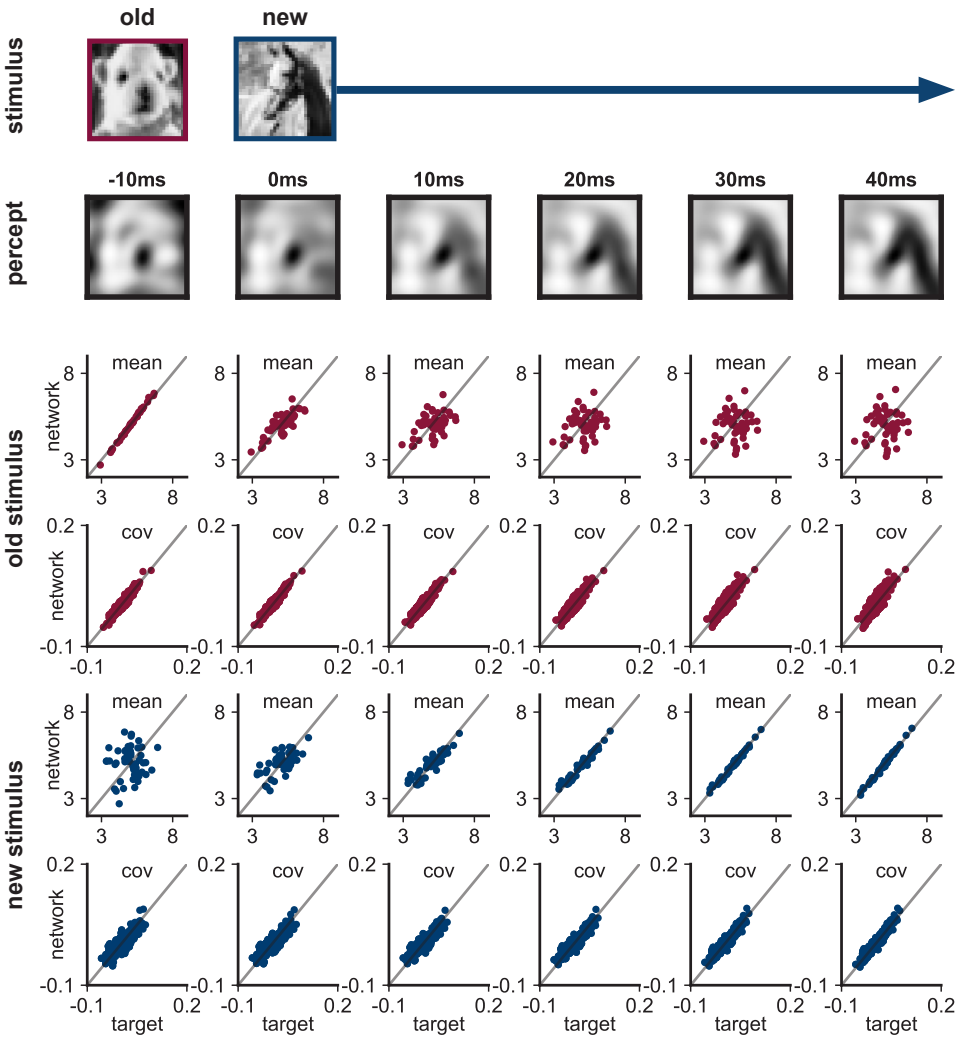

Figure 3: **Evolution of the gsSSN upon new stimulus onset.** Row 1: old and new stimuli. Row 2: internal percept reverse-engineered from the gsSSN. We see that the percept gradually morphs from the old stimulus to the new stimulus. Rows 3-6: network moments against GSM targets of the old (red) and new (blue) stimuli. The moments that were originally matched to the old stimulus start to deviate from their old values as soon as the new stimulus is introduced and slowly converge towards their new targets.

machine-learning setting, but not a problem specific to our training method. Note that our aim was to demonstrate that our approach allows us to train gsSSNs with a large number of free parameters while avoiding dynamic instabilities during training. In that regard, the gsSSN can be considered to be successfully trained. As always, the ability to train a network with many degrees of freedom is inherently a double-edged sword and there exists extensive literature on preventing over-fitting in amortized inference algorithms [40–42] which will motivate future work.

It is also interesting to analyze the trained gsSSN and explore its dynamics. We first note that it is possible to reverse-engineer the network in order to obtain the image that is being perceived. The mean membrane potential of the network at any given time can be mapped to the mean of latent activations of the GSM, and hence we may reconstruct a percept using the projective fields of the GSM (Fig 3), where we observe the interesting phenomenon of the percept morphing from the old to the new stimuli.

5 Discussion

Network stability during training is a problem faced in both artificial and biological neural networks. However, ANNs have the luxury of implementing quick-fix solutions that will not negatively impact its performance. Among the most well-known of these is batch normalization [43–45]. Furthermore, artificial neurons may be designed to be stable, such as simply choosing a saturating activation function (tanh,sigmoid), or other ways of implementing bounded-input-bounded-output designs.

Our purpose of starting with small networks and building up is to skirt around stability problems that SSNs are highly-prone to. In general, the idea of topology and weight evolving artificial neural network algorithms (TWEANNs) have been around for decades [46, 47]. As such, exploring and analyzing these methods may help refine our network building approach beyond the intuitive (but effective) approaches that we have adopted. One aspect that separates our method from TWEANNs is that we follow up a topological change with gradient-based optimization in order to compensate for what might essentially be an inefficient evolutionary step, as opposed to finding some optimal way of evolving the network.

In the demonstration of our training methodology, we have chosen to train gsSSNs to perform sampling-based inference because it is one of the most challenging tasks possible. The next intuitive step to go from here would be to train SSNs on other cognitive tasks, such as ones found in [16].

Another relevant point that needs to be pointed out is the difficulty of analysing non-linear networks such as SSNs. Certain theoretical results about SSNs have been justified using linear approximations [35] or Gaussianity assumptions [39], which have been partially successful. However, these are ultimately simplifications to the model that may fail to capture key non-linear/non-gaussian effects present within the network.

Throughout our analysis, we have been using Adam [48] to train gsSSNs. This is because our primary objective is to build biologically-realistic networks so as to study their dynamics and network characteristics after they have been optimized. In other words, our objective is the end goal, and not necessarily the journey. However, it might still be worth exploring and understanding how biologically-plausible learning can fit into our current paradigm in future investigations.

6 Limitations

In the fast-evolving research landscape of machine learning, speed and scalability are two of the most important criteria that determines how effective an algorithm is. In our case, building a network up during optimization goes against these principles, because that would require computationally-expensive function retracing operations every time the network changes in design. We justify this with two reasons. Firstly, to the best of our knowledge, there is presently no other effective way of training SSNs, especially unparametrized variants like gsSSNs. Secondly, the computational resources spent on optimization is still much greater than the resources spent on tracing, especially since there is only a finite number of tracing required (up till the network is fully built), but potentially an unlimited number of gradient computations and optimization iterations. On the other hand, we may also improve on our method by minimizing the number of retracing operations required (for example, by adding more neurons per building step). It will also be meaningful to perform investigations on how to maintain the balance between network building and its associated computational costs.

7 Conclusion

We have presented a deep learning toolbox consisting of two main training strategies to optimize SSNs. In an effort to demonstrate the effectiveness of our methods, we have undertaken the (until now) unachieved task of training SSNs to perform sampling-based probabilistic inference under a complex internal generative model, and succeeded in doing so (even with some over-fitting). This success opens up the possibility of training SSNs on other meaningful cognitive tasks in hopes of unravelling biological network dynamics underlying the complex computations performed by the brain.

References

[1] McCulloch, W. S. & Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics* **5**, 115–133 (1943).

[2] Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences* **79**, 2554–2558 (1982).

[3] Van Vreeswijk, C. & Sompolinsky, H. Chaos in neuronal networks with balanced excitatory and inhibitory activity. *Science* **274**, 1724–1726 (1996).

[4] Ciregan, D., Meier, U. & Schmidhuber, J. Multi-column deep neural networks for image classification. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 3642–3649 (2012).

[5] Hunsberger, E. & Eliasmith, C. Spiking deep networks with lif neurons (2015).

[6] Goodfellow, I. *et al.* Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. & Weinberger, K. (eds.) *Advances in Neural Information Processing Systems*, vol. 27 (Curran Associates, Inc., 2014).

[7] Ramesh, A. *et al.* Zero-shot text-to-image generation. In Meila, M. & Zhang, T. (eds.) *Proceedings of the 38th International Conference on Machine Learning*, vol. 139 of *Proceedings of Machine Learning Research*, 8821–8831 (PMLR, 2021).

[8] Vaswani, A. *et al.* Attention is all you need. In Guyon, I. *et al.* (eds.) *Advances in Neural Information Processing Systems*, vol. 30 (Curran Associates, Inc., 2017).

[9] Bengio, Y., Lee, D.-H., Bornschein, J., Mesnard, T. & Lin, Z. Towards biologically plausible deep learning (2015).

[10] Liao, Q. & Poggio, T. Bridging the gaps between residual learning, recurrent neural networks and visual cortex (2016).

[11] Gershman, S. J. The generative adversarial brain. *Frontiers in Artificial Intelligence* **2** (2019).

[12] Pogodin, R., Mehta, Y., Lillicrap, T. & Latham, P. E. Towards biologically plausible convolutional networks. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P. & Vaughan, J. W. (eds.) *Advances in Neural Information Processing Systems*, vol. 34, 13924–13936 (Curran Associates, Inc., 2021).

[13] Song, H. F., Yang, G. R. & Wang, X.-J. Training excitatory-inhibitory recurrent neural networks for cognitive tasks: A simple and flexible framework. *PLOS Computational Biology* **12**, e1004792 (2016).

[14] Orhan, A. E. & Ma, W. J. Efficient probabilistic inference in generic neural networks trained with non-probabilistic feedback. *Nature Communications* **8** (2017).

[15] Mante, V., Sussillo, D., Shenoy, K. V. & Newsome, W. T. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature* **503**, 78–84 (2013).

[16] Yang, G. R., Joglekar, M. R., Song, H. F., Newsome, W. T. & Wang, X.-J. Task representations in neural networks trained to perform many cognitive tasks. *Nature Neuroscience* **22**, 297–306 (2019).

[17] Banino, A. *et al.* Vector-based navigation using grid-like representations in artificial agents. *Nature* **557**, 429–433 (2018).

[18] Cueva, C. J. & Wei, X.-X. Emergence of grid-like representations by training recurrent neural networks to perform spatial localization. In *International Conference on Learning Representations* (2018).

[19] Yamins, D. L. K. *et al.* Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proc Natl Acad Sci U S A* **111**, 8619–24 (2014).

[20] Remington, E., Narain, D., Hosseini, E. & Jazayeri, M. Flexible sensorimotor computations through rapid reconfiguration of cortical dynamics. *Neuron* **98**, 1005–1019 (2018).

[21] Ahmadian, Y., Rubin, D. B. & Miller, K. D. Analysis of the stabilized supralinear network. *Neural Computation* **25**, 1994–2037 (2013).

[22] Wilson, H. R. & Cowan, J. D. Excitatory and inhibitory interactions in localized populations of model neurons. *Biophys J* **12**, 1–24 (1972).

[23] Li, Z. & Hopfield, J. J. Modeling the olfactory bulb and its neural oscillatory processings. *Biol Cybern* **61**, 379–392 (1989).

[24] Adini, Y., Sagi, D. & Tsodyks, M. Excitatory–inhibitory network in the visual cortex: Psychophysical evidence. *Proceedings of the National Academy of Sciences* **94**, 10426–10431 (1997).

[25] Litvak, V., Sompolinsky, H., Segev, I. & Abeles, M. On the transmission of rate code in long feedforward networks with excitatory–inhibitory balance. *Journal of neuroscience* **23**, 3006–3015 (2003).

[26] Ahmadian, Y. & Miller, K. D. What is the dynamical regime of cerebral cortex? *Neuron* **109**, 3373–3391 (2021).

[27] Rubin, D. B., Hooser, S. D. V. & Miller, K. D. The stabilized supralinear network: A unifying circuit motif underlying multi-input integration in sensory cortex. *Neuron* **85**, 402–417 (2015).

[28] Obeid, D. & Miller, K. D. Stabilized supralinear network: Model of layer 2/3 of the primary visual cortex. *bioRxiv* (2021).

[29] Kraynyukova, N. & Tchumatchenko, T. Stabilized supralinear network can give rise to bistable, oscillatory, and persistent activity. *Proceedings of the National Academy of Sciences* **115**, 3464–3469 (2018).

[30] Hennequin, G., Ahmadian, Y., Rubin, D. B., Lengyel, M. & Miller, K. D. The dynamical regime of sensory cortex: Stable dynamics around a single stimulus-tuned attractor account for patterns of noise variability. *Neuron* **98**, 846–860.e5 (2018).

[31] Priebe, N. J. & Ferster, D. Inhibition, spike threshold, and stimulus selectivity in primary visual cortex. *Neuron* **57**, 482–97 (2008).

[32] Hennequin, G., Aitchison, L. & Lengyel, M. Fast sampling-based inference in balanced neuronal networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. & Weinberger, K. (eds.) *Advances in Neural Information Processing Systems 27*, 2240–2248 (Curran Associates, Inc., 2014).

[33] Festa, D., Hennequin, G. & Lengyel, M. Analog memories in a balanced rate-based network of E-I neurons. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. & Weinberger, K. (eds.) *Advances in Neural Information Processing Systems 27*, 2231–2239 (Curran Associates, Inc., 2014).

[34] Eckmann, S. & Gjorgjieva, J. Synapse-type-specific competitive Hebbian learning forms functional recurrent networks. *bioRxiv* **10.1101**, 2022.03.11.483899 (2022).

[35] Echeveste, R., Aitchison, L., Hennequin, G. & Lengyel, M. Cortical-like dynamics in recurrent circuits optimized for sampling-based probabilistic inference. *Nature Neuroscience* **23**, 1138–1149 (2020).

[36] Wainwright, M. J. & Simoncelli, E. Scale mixtures of gaussians and the statistics of natural images. In Solla, S., Leen, T. & Müller, K. (eds.) *Advances in Neural Information Processing Systems*, vol. 12 (MIT Press, 1999).

[37] Orbán, G., Berkes, P., Fiser, J. & Lengyel, M. Neural variability and sampling-based probabilistic representations in the visual cortex. *Neuron* **92**, 530–543 (2016).

[38] Krizhevsky, A. & Hinton, G. Learning multiple layers of features from tiny images. Tech. Rep. 0, University of Toronto, Toronto, Ontario (2009).

[39] Hennequin, G. & Lengyel, M. Characterizing variability in nonlinear recurrent neuronal networks (2016).

[40] Shu, R., Bui, H. H., Zhao, S., Kochenderfer, M. J. & Ermon, S. Amortized inference regularization. *Advances in Neural Information Processing Systems* **31** (2018).

[41] Zhang, M., Hayes, P., Barber, D. & Center, A. Generalization gap in amortized inference .

[42] Douglas, L. *et al.* A universal marginalizer for amortized inference in generative models. *arXiv preprint arXiv:1711.00695* (2017).

[43] Ioffe, S. & Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Bach, F. & Blei, D. (eds.) *Proceedings of the 32nd International Conference on Machine Learning*, vol. 37 of *Proceedings of Machine Learning Research*, 448–456 (PMLR, Lille, France, 2015).

[44] Cooijmans, T., Ballas, N., Laurent, C., Gülçehre, u. & Courville, A. Recurrent batch normalization (2016).

[45] Santurkar, S., Tsipras, D., Ilyas, A. & Madry, A. How does batch normalization help optimization? In Bengio, S. *et al.* (eds.) *Advances in Neural Information Processing Systems*, vol. 31 (Curran Associates, Inc., 2018).

[46] Stanley, K. O. & Miikkulainen, R. Evolving neural networks through augmenting topologies. *Evolutionary computation* **10**, 99–127 (2002).

[47] Stanley, K. O., Clune, J., Lehman, J. & Miikkulainen, R. Designing neural networks through neuroevolution. *Nature Machine Intelligence* **1**, 24–35 (2019).

[48] Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] In our abstract and introduction, we claim that we have developed a deep learning approach to train supralinear stabilized networks. This method can be found in section 3. We also claim that we have successfully trained the network to perform sampling-based inference on an internal generative model of natural images. These results can be found summarized in Fig 2. Finally, we claim that our method is an order of magnitude better than previous approaches. This can be found in Table 1.
   (b) Did you describe the limitations of your work? [Yes] We have highlighted some potential limitations in a dedicated section, see section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] We have not found any potential negative impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A] We do not have theoretical results.
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The link to our code can be found in the supplementary material (hidden while under review). Additional information required to replicate our results are included in the supplementary.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] All parameters of the model are clearly introduced early in the text. Values of non-optimized parameters can be found in the supplementary materials.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Computational resources and related information can be found in the supplementary material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We have used the CIFAR-10 dataset and have cited the creators.
   (b) Did you mention the license of the assets? [Yes] We have mentioned the license in the supplementary.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] CIFAR-10 is a publicly available dataset that only requests that the creators be cited for usage.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] This can be found in the supplementary.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]