Deep Learning in 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 an effective method 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 stabilized by appropriately tuned recurrent inhibition. Our method avoids such instabilities by initializing a small network and gradually increasing network size via the dynamics-neutral addition of neurons during training. We first show how SSNs can be trained to perform typical machine learning tasks by training an SSN on MNIST classification. We then demonstrate the effectiveness 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. 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 of such tasks. For example, they described the presence of line attractors during context-dependent

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

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.

To bring the modeling of neural circuit mechanisms underlying challenging computations at (or near) single-cell resolution within reach, we develop a novel method to train an experimentally supported neural network model, the stabilized supralinear network (SSN) [21]. Critically, the SSN satisfies many of the key constraints of cortical circuits: it has separate but recurrent excitatory and inhibitory 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 excitatory-inhibitory (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].

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, excitatory 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). These properties make SSNs particularly susceptible to dynamical instabilities resulting in run-away excitation. 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 under-parameterized 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. Additionally, the effectiveness of our method allows SSNs to be trained for general tasks beyond specific narratives in neuroscience, which we demonstrate by training an SSN (a cortical neural circuit) for MNIST classification [36] (an artificial deep learning task) and achieving only slightly lower accuracy scores.

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) [37, 38], 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 of neural noise is not suppressed and recurrent connections are sufficiently strong – a combination that pushes the network 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 optimize. Our trained SSN performs accurate inference under a GSM with a set of basis functions that is sufficiently rich for allowing high-quality reconstructions of CIFAR-10 [39] images. This GSM represents an impossible target for previous, heavily constrained approaches for training SSNs. 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 Methods

2.1 Stochastic supralinear stabilized network

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 described by,

$$\boldsymbol{\tau}\frac{d\boldsymbol{u}}{dt} = -\boldsymbol{u} + f(\boldsymbol{h}) + \boldsymbol{W}\boldsymbol{r} + \boldsymbol{\eta} \tag{1}$$

where neurons are characterized by their membrane potentials $\boldsymbol{u}$ and time constants $\boldsymbol{\tau}$. $f(\boldsymbol{h})$ represents a non-linear transformation of the external input $\boldsymbol{h}$,

$$f : h_i \to \theta_1(h_i + \theta_2)^{\theta_3} \tag{2}$$

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,

$$r_i = k\lfloor u_i \rfloor_+^{\gamma} \tag{3}$$

for $\gamma > 1$ 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, which 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 (positive for excitatory neurons, negative for inhibitory neurons). Finally, $\boldsymbol{\eta}$ represents noise, correlated across time with time constant τ_η, as well as across neurons with covariance $\boldsymbol{\Sigma}_\eta$.

2.2 Theoretical motivation

We note that under the assumption of reasonable noise characteristics, it is always possible to add a new neuron to an existing stable SSN while preserving stability in a non-trivial manner. We begin by defining the newly-added neuron as a "twin" of an existing neuron. The new twin needs to have the same incoming weights as its original counterpart. This way, the newly-added neuron is guaranteed to be stable, since it is receiving the same input as an existing stable neuron. Secondly, all outgoing weights of both twins are halved, so that other neurons in the network now receive half output from two identical neurons, effectively keeping the total output unchanged. These other neurons are therefore still stable as they were originally. When implemented together, the result is a stable network with an extra neuron. We refer to this process as a dynamics-neutral addition of a neuron to a network. This idea forms the mathematical foundation of our proposed training method. Formally, let neuron n be the cloning candidate. The membrane potential vector $\boldsymbol{u}$ now has an additional element,

$$\boldsymbol{u} = \begin{bmatrix} \boldsymbol{u}_{\neg n} \\ u_n \end{bmatrix} \to \begin{bmatrix} \boldsymbol{u}_{\neg n} \\ u_n \\ u_n \end{bmatrix} \tag{4}$$

where $\neg n$ represents the indices of all neurons that are not n. Similarly, we split the weight matrix $\boldsymbol{W}$ into four parts, $\{\boldsymbol{W}_{(\neg n,\neg n)}, \boldsymbol{W}_{(\neg n,n)}, \boldsymbol{W}_{(n,\neg n)}, \boldsymbol{W}_{(n,n)}\}$ We update the network weights according to,

$$\begin{bmatrix} \boldsymbol{W}_{(\neg n,\neg n)} & \boldsymbol{W}_{(n,\neg n)} \\ \boldsymbol{W}_{(\neg n,n)} & \boldsymbol{W}_{(n,n)} \end{bmatrix} \to \begin{bmatrix} \boldsymbol{W}_{(\neg n,\neg n)} & (1/2)\boldsymbol{W}_{(n,\neg n)} & (1/2)\boldsymbol{W}_{(n,\neg n)} \\ \boldsymbol{W}_{(\neg n,n)} & (1/2)\boldsymbol{W}_{(n,n)} & (1/2)\boldsymbol{W}_{(n,n)} \\ \boldsymbol{W}_{(\neg n,n)} & (1/2)\boldsymbol{W}_{(n,n)} & (1/2)\boldsymbol{W}_{(n,n)} \end{bmatrix} \tag{5}$$

The synaptic input into all neurons is now given by,

$$\begin{bmatrix} \boldsymbol{W}_{(\neg n,\neg n)} & (1/2)\boldsymbol{W}_{(n,\neg n)} & (1/2)\boldsymbol{W}_{(n,\neg n)} \\ \boldsymbol{W}_{(\neg n,n)} & (1/2)\boldsymbol{W}_{(n,n)} & (1/2)\boldsymbol{W}_{(n,n)} \\ \boldsymbol{W}_{(\neg n,n)} & (1/2)\boldsymbol{W}_{(n,n)} & (1/2)\boldsymbol{W}_{(n,n)} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_{\neg n} \\ u_n \\ u_n \end{bmatrix} = \begin{bmatrix} \boldsymbol{W}_{(\neg n,\neg n)}\boldsymbol{u}_{\neg n} + \boldsymbol{W}_{(n,\neg n)}u_n \\ \boldsymbol{W}_{(\neg n,n)}\boldsymbol{u}_{\neg n} + \boldsymbol{W}_{(n,n)}u_n \\ \boldsymbol{W}_{(\neg n,n)}\boldsymbol{u}_{\neg n} + \boldsymbol{W}_{(n,n)}u_n \end{bmatrix} \tag{6}$$

where,

$$\begin{bmatrix} \boldsymbol{W}_{(\neg n,\neg n)}\boldsymbol{u}_{\neg n} + \boldsymbol{W}_{(n,\neg n)}u_n \\ \boldsymbol{W}_{(\neg n,n)}\boldsymbol{u}_{\neg n} + \boldsymbol{W}_{(n,n)}u_n \end{bmatrix} = \boldsymbol{W}\boldsymbol{u} \tag{7}$$

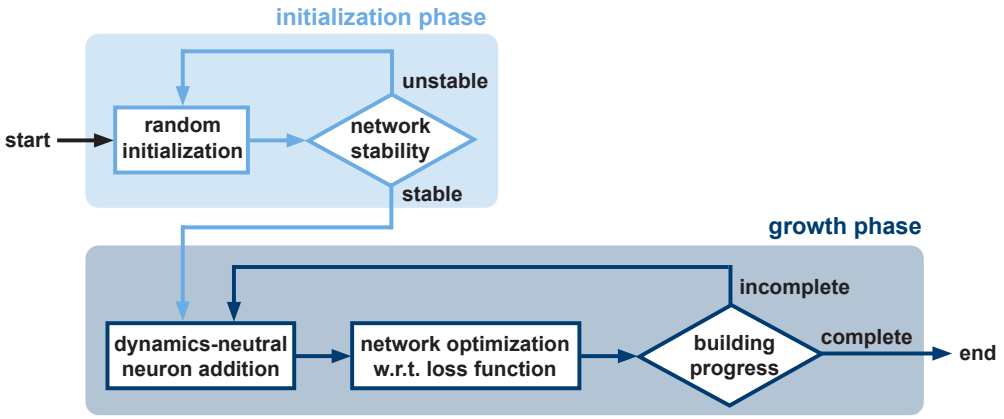

Figure 1: **Proposed method for building an optimized SSN.** Due to the difficulty of initializing a large SSN, we start with a small network of two neurons, comprising of an excitatory and an inhibitory neuron by generating network parameters until a set of parameters corresponding to a stable network is found (initialization phase). We subsequently add neurons in a dynamics-neutral manner while concurrently optimizing for the training objective until the desired network size is achieved (growth phase).

which is the same input received before the addition of the new neuron, as claimed. Note that during optimization, symmetry between the twin neurons are broken due to them receiving statistically equal but nonetheless uniquely-generated noise inputs. The external input may also be different depending on the training objective. These factors wear down the absolute guarantee of stability, but it is still reasonable to build on this intuitively high chance of stability.

2.3 Training method

Even with a reliable way of growing a network, we still explicitly require an existing (smaller) network as a starting point. By "reverse-induction", we recognize that the simplest and most reliable solution is to initialize the whole process by generating the smallest SSN possible – a network with only one excitatory and one inhibitory neuron. This can be achieved by brute-force or by some targeted initialization specific to the training objective in order to get a head-start on performance optimization. The full network can then be gradually constructed by adding neurons in a dynamics-neutral way as previously described. The entire training method is summarized in Fig 1.

2.4 Training reliability

In order to quantify the reliability of our training method, we propose and calculate three different metrics that will help to show how our method is effective against instability problems typically encountered in training SSNs.

1. **Proportion unstable in separate instances of network building.** We repeat the entire training method, starting from re-initializing a small network to fully-building it up in a total of 100 separate instances and counted the proportion of unstable trials.
2. **Proportion unstable when shuffling weights of stable networks.** In each of the 100 instances, we shuffled the elements of the same type (E-E, E-I, I-E and I-I) within the weight matrix for 100 trials and counted the number of unstable configurations when subjected to the first 100 images from either MNIST or CIFAR-10. We then report the mean and standard error over all instances.
3. **Proportion unstable of randomly generated networks.** We randomly generate 10000 initializations of the full network and counted the proportion of unstable configurations. The elements of the weight matrices are initialized such that the variance of their absolute values match that of our trained stable network. We then enforce polarity rules according to whether the connecting neurons are excitatory or inhibitory.

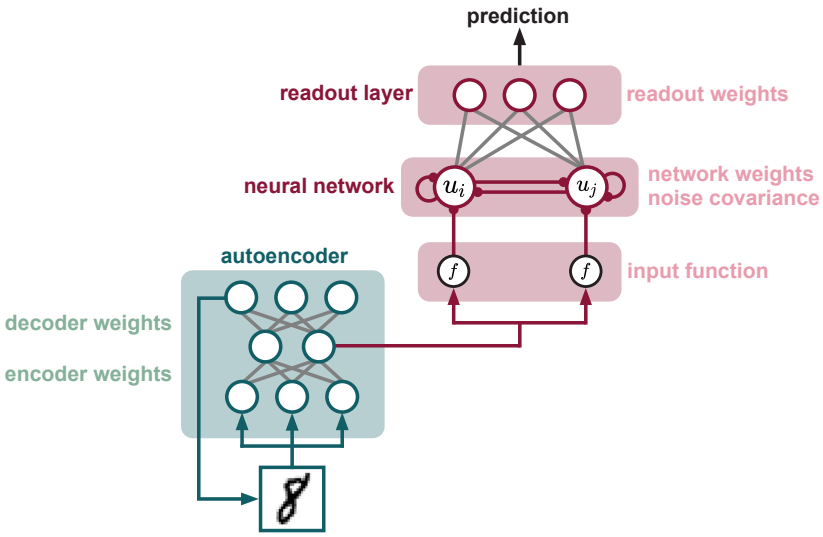

Figure 2: **Training an SSN to perform MNIST classification task.** The network comprises of 40 excitatory and 10 inhibitory neurons (not depicted). Before training the SSN, we optimize a 3-layer autoencoder (green) to reduce the dimensions of each MNIST image to fit the number of excitatory neurons, which will be receiving the encoded image as input after being transformed by the input function. The SSN is then allowed to evolve over time until it reaches stationary, after which the mean membrane potentials are passed through a feedforward readout layer to make a prediction (red). Trainable variables are highlighted on the two sides of the figure.

3 MNIST Classification Task

We seek to demonstrate how SSNs can be generalized for typical deep learning tasks by training an SSN to perform MNIST classification [36]. The nature of its dynamics in (1) implies that neural activity evolves over time (at a rate determined by the time constants) in response to external input instead of producing an instant output. The presence of noise further complicates the accessibility of obtaining a straightforward readout from the network. Naturally, these issues are ever-present in experimental neural recordings and contribute to the biological-realism of the model. In artificial neural networks, an input is processed across different layers according to their respective network architectures and the result is an immediate output afterwards. As such, in order to train an SSN like an artificial neural network, we choose to evolve the network over time until it reaches steady-state, and pass its mean membrane potentials into a readout (softmax) layer in order to produce predictions.

An overview of the training objective is depicted in Fig 2. We aim to train an SSN with 80 excitatory and 20 inhibitory neurons which is a biologically-realistic E-I ratio. We also train an SSN with 50 excitatory and 50 inhibitory neurons so that we may compare with the performance of previous models [35]. Only the excitatory neurons will receive external input (i.e. MNIST images) and be decoded in the readout layer, although the entire network (including the inhibitory neurons) is evolved over time, as is commonly observed in biology. In order to transform MNIST images into input for the SSN, we optimize an autoencoder (also comprising of 3 layers) to reduce input dimensionality from 784 to the number of excitatory neurons. As a base comparison, we also separately train 3-layer MLPs and perform logistic regression on the encoded data from the autoencoder.

Training results are summarized in Table 1. Our trained SSNs at both E-I ratios perform at only slightly lower levels than their corresponding MLPs, and also better than logistic regression despite having noise injected at every time step and also having to maintain dynamic stability. In all 200 separate instances of networks built, none of the instances encountered instability problems. Shuffling the weights within their connection-blocks also resulted in mostly unstable configurations, suggesting that the magnitude of the weights do not play as much of a role as its exact distribution.

Previous models [35] are not able to reasonably perform the task because they have a highly-constrained parameterization designed to exploit rotational symmetries in the images which do

Table 1: **Performance and method reliability of training the SSN on MNIST**. Numbers in parentheses indicate the number of excitatory and inhibitory neurons in the SSN respectively. We reduce input dimensionality using an autoencoder to fit the number of excitatory neurons (50 in one model and 80 in another model). The encoded data are labeled accordingly. For comparison, we separately trained MLPs (with 3 layers) and perform logistic regression to classify both the original and encoded data. Despite the presence of noise, our trained SSN performs only slightly below MLP and above logistic regression.

performance		
model	MNIST data	test accuracy
Logit	encoded-50	0.914
MLP	encoded-50	0.974
[35] (50/50)	encoded-50	0.223
SSN (50/50)	encoded-50	**0.949**
Logit	encoded-80	0.922
MLP	encoded-80	0.976
SSN (80/20)	encoded-80	**0.952**
training reliability		
model	reliability metric	proportion unstable
SSN (50/50)	separate instances of network building	0.0
	shuffling weights of stable networks	0.790 ($\pm$ 0.007)
	randomly generated networks	1.0
SSN (80/20)	separate instances of network building	0.0
	shuffling weights of stable networks	0.794 ($\pm$ 0.008)
	randomly generated networks	1.0

not exist in the MNIST dataset. However, we note that it is still performing above chance level (which is 0.1). Furthermore, we observe that it is virtually not possible to initialize a stable random configurations of networks in both E-I ratios.

4 Amortized Inference under Generative Model

We now shift our focus to a biological narrative. For a more direct comparison with our method, we will train an SSN to perform amortized sampling-based probabilistic inference under the GSM – something that was previously attempted with limited success [35]. An overview of the GSM and training objective of the SSN is summarized in Fig 3.

4.1 Gaussian scale mixture model

The GSM [37, 38] is proposed as an internal generative model of natural image patches in V1. An image $\boldsymbol{x}$ is generated by,

$$\boldsymbol{x} = z\boldsymbol{A}\boldsymbol{y} + \boldsymbol{\eta}_x \tag{8}$$

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,

$$g(x, y; x_0, y_0, \phi, \sigma) = \exp\left(-\frac{x'(\phi)^2 + y'(\phi)^2}{2\sigma^2}\right)\cos\left(\frac{2.5}{\sigma}x'(\phi)\right) \tag{9}$$

where x' and y' are pixel coordinates rotated by ϕ. The parameters of $\boldsymbol{A}$ are summarized as $\boldsymbol{\theta}_A = \{\boldsymbol{x}_0, \boldsymbol{y}_0, \boldsymbol{\phi}, \boldsymbol{\sigma}\}$. Note that $\boldsymbol{A}$ is a two-dimensional matrix with each row corresponding to the flattened pixels of single 2-D Gabor filter. An ideal observer, when presented with an image $\boldsymbol{x}$,

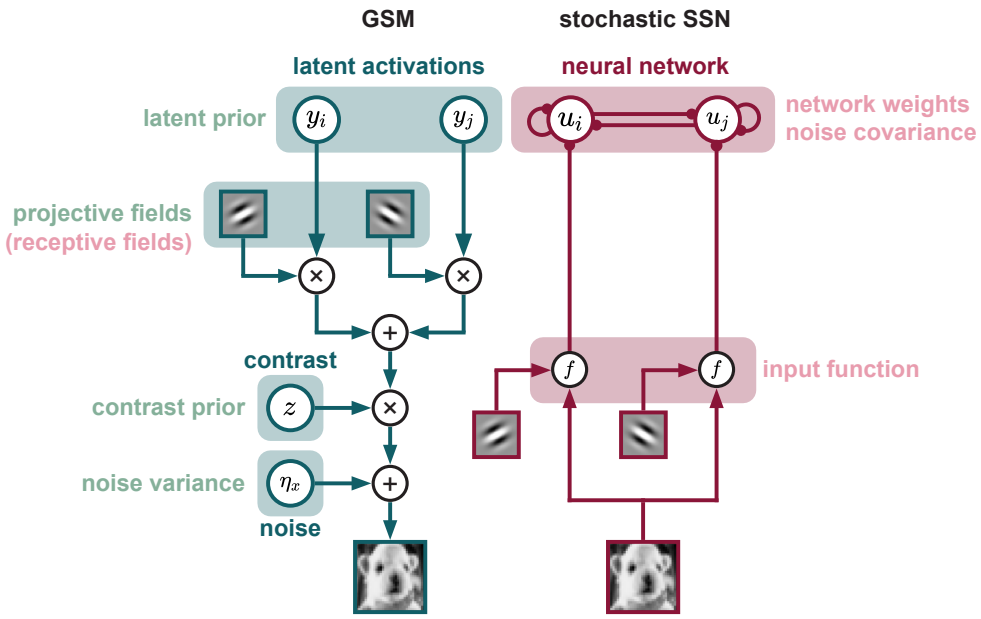

Figure 3: **Overview of the GSM and SSN training objective.** The GSM is an internal generative model of natural image patches proposed to underlie computations in the visual cortex. An image is constructed from projective fields, modulated by a global scalar contrast level and finally mixed with Gaussian pixel noise. We seek to train an SSN to perform sampling-based probabilistic inference under the GSM, where the neural response of the SSN given some external input image represents statistical samples from the posterior over latents of the GSM given the same observed image.

computes a posterior distribution over $\boldsymbol{y}$ and z by incorporating their respective priors $p(\boldsymbol{y})$ and $p(z)$,

$$p(\boldsymbol{\eta}_x) = \mathcal{N}(\boldsymbol{\eta}_x; 0, \sigma_x^2 \boldsymbol{I}) \tag{10}$$

$$p(\boldsymbol{y}) = \mathcal{N}(\boldsymbol{y}; 0, \boldsymbol{C}) \tag{11}$$

$$p(z) = \Gamma(z; \alpha, \beta) \tag{12}$$

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

$$p(\boldsymbol{y}|\boldsymbol{x}_i) = \mathbb{E}_z \left[p(z|\boldsymbol{x}_i) p(\boldsymbol{y}|z, \boldsymbol{x}_i) \right] \tag{13}$$

We construct our GSM by first maximizing fraction of variance explained by the projective fields on CIFAR-10 images and subsequently optimize the remaining model parameters by maximizing likelihood of observing the images. Additional details can be found in Appendix A.

4.2 Sampling-based probabilistic inference

Given an observed image as input, the membrane potential response of the SSN, as determined by (1), is optimized to represent statistical samples for the posterior distribution over the latent variables of the GSM, as computed by an ideal observer using (13). This inference method is therefore described as "sampling-based". We implement this by optimizing the model with respect to the mean-squared errors between the first and second order moments of both distributions,

$$\mathcal{L} = \lambda_\mu \, \Sigma_{n=1}^{N_e} \left(\mathbb{E}[u_{\text{SSN},n}] - \mu_{\text{GSM},n} \right)^2 + \lambda_\Sigma \, \Sigma_{n=1}^{N_e} \Sigma_{m=1}^{N_e} \left(\text{cov}[u_{\text{SSN},n}, u_{\text{SSN},m}] - \boldsymbol{\Sigma}_{\text{GSM},n,m} \right)^2 \tag{14}$$

for constant coefficients λ_μ and λ_Σ. In this case, both excitatory and inhibitory neurons receive an external input corresponding to its respective receptive fields, but similar to the previous example, only the membrane potentials of excitatory neurons are considered in the computation of the cost function. The mathematical form of the cost is motivated by experimental practice – neural data is often averaged over time or across trials in order to reduce noise. More recently, neural variability is proposed to also be computationally meaningful [38], and we thus seek to understand the dynamics of how the SSN expresses its mean and variance.

Table 2: **Performance and method reliability of training the SSN to perform inference under GSM**. SSNs have been previously been trained for the same task [35], which will serve as the base comparison. We train two SSNs, one for the GSM in [35] (labeled as SSN_1), and another for our GSM which consists of a diverse set of basis functions optimized from CIFAR-10 images (SSN_2). Since training targets are different for the two GSMs, we compute a modified cost (15) in order to directly compare optimization performance.

performance		
model	GSM target	normalized cost
[35]	[35]	1.0
SSN_1	[35]	**0.158**
[35]	optimized for CIFAR-10	$\sim 10^4$
SSN_2	optimized for CIFAR-10	**0.745**
training reliability		
model	reliability metric	proportion unstable
SSN_2	separate instances of network building	0.0
	shuffling weights of stable networks	0.446 (± 0.008)
	randomly generated networks	1.0

4.3 Results

We train an SSN for the GSM constructed in [35] (denoted as SSN_1) and train another SSN on our in-house GSM (denoted as SSN_2). We do the same for existing models in [35]. Since target moments are different for the two GSMs, we compute a normalized cost function so that we can compare networks trained across different GSMs,

$$\mathcal{L}' = \lambda_\mu \frac{\Sigma_{n=1}^{N_e} \left(\mathbb{E}[u_{\text{SSN},n}] - \mu_{\text{GSM},n}\right)^2}{\sqrt{\mathbb{V}_n[\mu_{\text{GSM},n}]}} + \lambda_\Sigma \frac{\Sigma_{n=1}^{N_e} \Sigma_{m=1}^{N_e} \left(\text{cov}[u_{\text{SSN},n}, u_{\text{SSN},m}] - \mathbf{\Sigma}_{\text{GSM},n,m}\right)^2}{\sqrt{\mathbb{V}_{n,m}[\Sigma_{\text{GSM},n,m}]}} \tag{15}$$

In order to fully appreciate these results, we briefly describe the GSM-SSN pairs in each row of Table 2 and how they are trained. A full comparison can be found in Appendix B.

1. The first row describes the SSN from [35] optimized for the GSM from the same authors. This SSN is highly constrained; for example, the entire 100×100 weight matrix is parameterized by only 8 parameters. The same 5 rotationally-symmetric images are used for both training and testing, attributing to the high computational costs associated with the method. Optimization is first done using Adam for a small number of iterations, followed by a biased semi-analytical method described in [40] (as well as Appendix B). The normalized cost of this model is set to 1 by definition.
2. The SSN in the second row is trained by our method, with the same 5 images (as the previous row) used as the training and test set. The weight matrix is constrained to be block-symmetric-circulant in order to fully exploit the rotational symmetry of the images. Each symmetric-circulant 50×50 block is parameterized by 26 parameters, giving a total of 104 parameters for the weight matrix. The additional degrees of freedom has resulted in nearly an order of magnitude lower cost.
3. The third row shows the SSN from [35] trained on our GSM. Due to its highly-constrained parameterization (as mentioned in the description of the first row) and built-in rotational symmetry assumptions, the model is not able to succeed in matching the moments of the GSM posterior moments when observing CIFAR-10 natural images and fails to converge during training.
4. The SSN in the fourth row is trained on all 50000 training images from CIFAR-10. The images have no symmetry, and the 100×100 weight matrix is freely represented by 10000 parameters. Despite the increase in difficulty of the task (matching a more complex GSM), our method was able to produce a network that achieves a normalized cost lower than the attempt in [35].

Here, our method provides two advantages. Firstly, the additional degrees of freedom gained by the networks from having more free parameterizations allow them to achieve a much lower cost when trained on the same GSM (first two rows). Secondly, our stability-focused approach has successfully trained a network to perform inference under a complex GSM of natural images, a feat which was previously impossible (next two rows). Once again, in all 100 independent instances, not a single instance went unstable. The full network is also unable to be randomly initialized in a stable manner, which further motivates our approach.

5 Discussion

Network stability during training is a problem faced in both artificial and biological neural networks. Typically, artificial neural networks (ANNs) can implement quick-fix solutions that will not negatively impact its performance, such as applying batch and layer normalizations [41–43]. 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. There also exists ANNs that are challenged by instabilities that cannot be trivially solved [44, 45], which could motivate similar network-building solutions.

In general, the concept of building networks during training has already been proposed in ANNs [46–48]. One such study combined both growing and pruning to produce efficient networks [47]. Another training approach decides on an existing neuron in the network to be split into multiple neurons [46]. Another proposed algorithm [48] involves a similar principle of adding nodes without impacting existing network capabilities. Broadly speaking, the two primary goals of these algorithms are to reduce training load and improve performance, which are perfectly reasonable. Here, we are instead targeting an even more basic need of "survival" by staying within a stable parameter regime, achieved using our novel method of dynamics-neutral neuron duplication.

On the same train of thought, it can be meaningful to explore the tuning of certain hyperparameters in our method, such as the number of neurons to duplicate at the same time, or the criteria in selecting which neuron to duplicate. Now that stability is guaranteed, we are motivated to refine the method by exploring other hyperparameters, which motivates future work. It is also worthwhile to consider an orthogonal approach of adding terms to the cost function such as regularization and explore its effects on keeping network optimization within a stable parameter regime.

Throughout our analysis, we have been using Adam [49] to train SSNs (additional miscellaneous training information can be found in Appendix C). 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.

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. 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. It will also be meaningful to perform investigations on how to maintain the balance between network building and its associated computational costs.

6 Conclusion

We present an effective method of training the SSN, a canonical model of cortical circuits which is highly prone to dynamic instabilities. We show how the SSN can be trained to perform the task of classifying MNIST images, thereby clearly showing the generality of both the SSN and our method to train it. Our success opens up the possibility of training SSNs on other meaningful cognitive tasks in hopes of unraveling biological network dynamics underlying 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] Deng, L. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine* **29**, 141–142 (2012).

[37] 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).

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

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

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

[41] 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).

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

[43] 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).

[44] Bai, S., Kolter, J. Z. & Koltun, V. Deep equilibrium models. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 688–699 (2019).

[45] Bai, S., Koltun, V. & Kolter, J. Z. Stabilizing equilibrium models by jacobian regularization. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, vol. 139 of *Proceedings of Machine Learning Research*, 554–565 (PMLR, 2021).

[46] Wu, L., Wang, D. & Liu, Q. Splitting steepest descent for growing neural architectures. In Wallach, H. *et al.* (eds.) *Advances in Neural Information Processing Systems*, vol. 32 (Curran Associates, Inc., 2019).

[47] Yuan, X., Savarese, P. H. P. & Maire, M. Growing efficient deep networks by structured continuous sparsification. In *International Conference on Learning Representations* (2021).

[48] Evci, U., van Merrienboer, B., Unterthiner, T., Pedregosa, F. & Vladymyrov, M. Gradmax: Growing neural networks using gradient information. In *International Conference on Learning Representations* (2022).

[49] 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 4. 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 3. Finally, we claim that our method is an order of magnitude better than previous approaches. This can be found in Table 2.
   (b) Did you describe the limitations of your work? [Yes] We have highlighted some potential limitations in a dedicated section, see section 5.
   (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? [Yes]
   (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. 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 CIFAR-10 and MNIST and have cited the creators.
   (b) Did you mention the license of the assets? [Yes] We have linked the licenses 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 and MNIST are publicly available datasets.
   (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]