Learning Credit Assignment

Chan Li and Haiping Huang*
PMI Lab, School of Physics, Sun Yat-sen University, Guangzhou 510275, People's Republic of China

(Received 16 January 2020; revised 1 July 2020; accepted 22 September 2020; published 22 October 2020)

Deep learning has achieved impressive prediction accuracies in a variety of scientific and industrial domains. However, the nested nonlinear feature of deep learning makes the learning highly nontransparent, i.e., it is still unknown how the learning coordinates a huge number of parameters to achieve decision-making. To explain this hierarchical credit assignment, we propose a mean-field learning model by assuming that an ensemble of subnetworks, rather than a single network, is trained for a classification task. Surprisingly, our model reveals that apart from some deterministic synaptic weights connecting two neurons at neighboring layers, there exists a large number of connections that can be absent, and other connections can allow for a broad distribution of their weight values. Therefore, synaptic connections can be classified into three categories: very important ones, unimportant ones, and those of variability that may partially encode nuisance factors. Therefore, our model learns the credit assignment leading to the decision and predicts an ensemble of sub-networks that can accomplish the same task, thereby providing insights toward understanding the macroscopic behavior of deep learning through the lens of distinct roles of synaptic weights.

DOI: 10.1103/PhysRevLett.125.178301

Introduction.—As deep neural networks become an increasingly important tool in diverse domains of scientific and engineering applications [1–5], the black box properties of the tool turn out to be a challenging obstacle puzzling researchers in the field [6]. In other words, the decision-making behavior of a network output cannot be easily understood in terms of interactions among building components of the network. This shares the same spirit as another long-standing puzzle in the theory of the brain: how the emergent behavior of a neuronal population hierarchy can be traced back to its elements [5,6]. This challenging issue is the well-known credit assignment problem, that is, determining how much credit a given component (either neuron or connection) should take for a particular behavior output [5]. To solve this problem, one needs to bridge the gap between the microscopic interactions of components and macroscopic behavior.

Excitingly, recent works show that there exist sub-networks of random weights that are able to produce better-than-chance accuracies [7–9]. This property seems to be universal across different architectures, datasets, and computational tasks [10]. One can even start with no connections and add complexity as needed by assuming a single shared weight parameter [11]. Moreover, it was recently revealed that the innate template of face-selective neurons can spontaneously emerge from sufficient statistical variations present in the random initial wirings of neural circuits, while the template may be fine-tuned during early visual experiences [12]. Therefore, from both an artificial and a biological neural network perspective, exploring how and why these random wirings exist will definitely provide us a powerful lens through which we can better understand and even further improve the computational capacities of deep neural networks.

Here, we propose a statistical model of learning credit assignment from training data of a computational task. According to the model, we search for an optimal random network ensemble as an inductive bias about the hypothesis space [6]. The hypothesis space is composed of all candidate networks with different assignments of weight values that accomplish the computation task. Consistent with previous studies [7–9], the optimal ensemble contains subnetworks of the original full network, which further allows for capturing uncertainty in the hypothesis space. The model can be solved by mean-field methods, thereby providing a physics interpretation of how credit assignment occurs in a hierarchical deep neural system.

Model.—To learn credit assignment, we search for an optimal random neural network ensemble to accomplish a classification task of handwritten digits [13]. More precisely, we design a deep neural network of L layers, including $L-2$ hidden layers. The depth of the network L can be made arbitrarily large. The size (width) of the lth layer is denoted by N_l. Therefore, N_1 is determined by the total number of pixels in an input image, and N_L is the number of classes (e.g., ten for the handwritten digit dataset). The weight value of the connection from neuron i at the upstream layer l to neuron k at the downstream layer $l+1$ is defined by w_{ik}^l, and the activation of the neuron k at the $(l+1)$th layer h_k^{l+1} is a nonlinear function of the preactivation $z_k^{l+1} = (1/\sqrt{N_l}) \sum_i w_{ik}^l h_i^l$, where the weight scaling factor $1/\sqrt{N_l}$ ensures that the weighted sum is

independent of the upstream layer width. We use the rectified linear unit (ReLU) function [14] as the transfer function from preactivation z to activation h, defined as $h = \max(0, z)$. The output transfer function specifies a probability over all classes of the input image by using a softmax function $h_k = (e^{z_k}/\sum_i e^{z_i})$, where z_i is the preactivation of the ith neuron at the output layer. For the categorization task, we define $\hat{h}_i$ as the target label (one-hot representation) and use the cross entropy $\mathcal{C} = -\sum_i \hat{h}_i \ln h_i$ as the objective function to be minimized.

Training the neural network corresponds to adjusting all connection weights to minimize the cross entropy until the network is able to classify unseen handwritten digits with a satisfied accuracy (the so-called generalization ability). Therefore, after the network is trained with a training data size of T, the network's generalization ability is verified with a test data size of V. Remarkably, the state-of-the-art test accuracy is able to surpass the human performance in some complex tasks [3]. However, the decision-making behavior of the output neurons in a deep network is still challenging to understand in terms of computational principles of single building components (either neurons or weights). Recent empirical machine learning works showed that a subnetwork of random weights can produce a better-than-chance accuracy [7–9]. This clearly suggests that there may exist a random ensemble of neural networks that fulfill the computational task given the width and depth of the deep network. This ensemble may occupy a tiny portion of the entire model space. Therefore, a naive random initialization of the neural network can only yield a chance-level accuracy. To incorporate all these challenging issues into a theoretical model, we propose to model the weight by a spike and slab (SaS) distribution as follows:

$$P(w_{ik}^l) = \pi_{ik}^l \delta(w_{ik}^l) + (1 - \pi_{ik}^l)\mathcal{N}(w_{ik}^l | m_{ik}^l, \Xi_{ik}^l), \quad (1)$$

where the discrete probability mass at zero defines the spike, and the slab is characterized by a Gaussian distribution with mean m_{ik}^l and variance Ξ_{ik}^l over a continuous domain (see Fig. 1 for an illustration).

The SaS distribution has been widely used in the statistics literature [15,16]. Here, the spike and slab, respectively, have their own physics interpretations in the learning credit assignment of neural networks. The spike is intimately related to the concept of network compression [10,17,18], where not all resources of connections are used in a task. This parameter allows one to identify very important weights and further evaluate remaining capacities for learning new tasks [19]. A recent physics study has already shown that a deep neural network can be robust against connection removals [20]. The slab continuous support characterizes the ensemble of neural networks with random weights producing better-than-chance accuracies. Among these weights, some are very important, indicated by a vanishing spike probability mass,

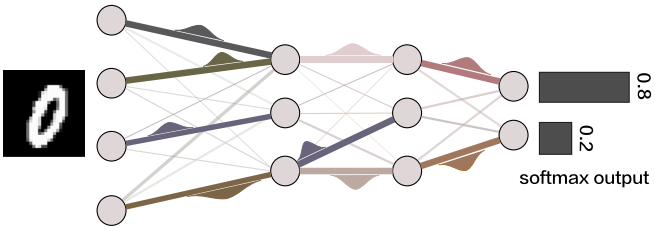

FIG. 1. Schematic illustration of a model of learning credit assignment. A deep neural network of four layers, including two hidden layers, is used to recognize a handwritten digit, say zero, with the softmax output indicating the probability of the categorization. Each connection is specified by a spike and slab distribution, where the spike indicates the probability of the absence of this connection and the slab is modeled by a Gaussian distribution of weight values, as pictorially shown only on strong connections with different means and variances. Other weak connections indicate nearly unit spike-probabilities, although they also carry a slab distribution (not shown in illustration for simplicity).

and could thus explain the decision-making of the output neurons, while the variance of the corresponding Gaussian distribution captures the uncertainty of the decision-making solutions [21]. Therefore, the inductive bias of connections and their associated weights can be learned through the SaS model of credit assignment. Note that the Gaussian slab is not used here as an additional regularization complexity term in the objective function [22]. Instead, the continuous slab is combined coherently with the spike probability to model the uncertainty of weights facing noisy sensory inputs (Fig. 1).

Next, we derive a mean-field method to learn the SaS parameters $\boldsymbol{\theta}_{ik}^l \equiv (\pi_{ik}^l, m_{ik}^l, \Xi_{ik}^l)$ for all layers. The first and second moments of the weight w_{ik}^l are given by $\mu_{ik}^l \equiv \mathbb{E}[w_{ik}^l] = m_{ik}^l(1 - \pi_{ik}^l)$ and $\varrho_{ik}^l \equiv \mathbb{E}[(w_{ik}^l)^2] = (1 - \pi_{ik}^l)[\Xi_{ik}^l + (m_{ik}^l)^2]$, respectively. Given a large width of the layer, the central-limit theorem implies that the preactivation follows approximately a Gaussian distribution $\mathcal{N}[z_i^l | G_i^l, (\Delta_i^l)^2]$, where the mean and variance are given respectively by

$$G_i^l = \frac{1}{\sqrt{N_{l-1}}} \sum_k \mu_{ki}^{l-1} h_k^{l-1}, \quad (2a)$$

$$(\Delta_i^l)^2 = \frac{1}{N_{l-1}} \sum_k [\varrho_{ki}^{l-1} - (\mu_{ki}^{l-1})^2](h_k^{l-1})^2. \quad (2b)$$

Then, the feed-forward transformation of the input signal can be reparameterized by [23,24]

$$z_i^l = G_i^l + \epsilon_i^l \Delta_i^l, \quad (3a)$$

$$h_i^l = \mathrm{Re\,LU}(z_i^l), \quad (3b)$$

for $l < L$. The last layer uses the softmax function. ϵ_i^l is a layer- and component-dependent standard Gaussian

random variable with zero mean and unit variance. $\boldsymbol{\epsilon}^l$ is quenched for every single training mini-epoch and the same value is used in both forward and backward computations. This reparameterization retains the statistical structure in Eq. (2). The objective function relies on $(\boldsymbol{\mu}, \boldsymbol{\varrho})$ that can be preserved by a transformation of $\boldsymbol{\theta}$. Hence, we use a regularization strength to control the ℓ_2 norm of $\boldsymbol{m}$ and $\boldsymbol{\Xi}$. In addition, the transformation does not change the fraction of $\pi = 1$ or 0 [25], maintaining the qualitative behavior of the model.

Learning of the hyperparameter $\boldsymbol{\theta}_{ik}^l$ can be achieved by a gradient descent of the objective function, i.e.,

$$\Delta\boldsymbol{\theta}_{ki}^l = -\eta \mathcal{K}_i^{l+1} \frac{\partial z_i^{l+1}}{\partial \boldsymbol{\theta}_{ki}^l}, \tag{4}$$

where η denotes the learning rate, and $\mathcal{K}_i^{l+1} \equiv (\partial \mathcal{C}/\partial z_i^{l+1})$. The entire dataset is divided into minibatches over which the gradients are evaluated. Note that unlike the standard back-propagation (BP) [26], we adapt the SaS distribution rather than a particular weight, in accord with our motivation of learning statistical features of the hypothesis space for a particular computation task (e.g., image classification here).

On the top layer, $\mathcal{K}_i^L$ can be directly estimated as $\mathcal{K}_i^L = -\hat{h}_i^L(1 - h_i^L)$ by definition. For lower layers, $\mathcal{K}_i^l$ can be estimated by using the chain rule, resulting in a BP equation of the error signal from the top layer:

$$\mathcal{K}_i^l = \delta_i^l f'(z_i^l), \tag{5a}$$

$$\delta_i^l = \sum_k \mathcal{K}_k^{l+1} \frac{\partial z_k^{l+1}}{\partial h_i^l}, \tag{5b}$$

where $\delta_i^l \equiv (\partial \mathcal{C}/\partial h_i^l)$, and $f'(\cdot)$ denotes the derivative of the transfer function.

To proceed, we have to compute $(\partial z_k^{l+1}/\partial h_i^l)$ and $(\partial z_i^{l+1}/\partial \boldsymbol{\theta}_{ki}^l)$. The first derivative characterizes how sensitive the preactivation is under the change of the input activity of one neuron, and this response is computed as follows:

$$\frac{\partial z_k^{l+1}}{\partial h_i^l} = \frac{\mu_{ik}^l}{\sqrt{N_l}} + \frac{[\varrho_{ik}^l - (\mu_{ik}^l)^2] h_i^l}{N_l \Delta_k^{l+1}} \epsilon_k^{l+1}. \tag{6}$$

The second derivative characterizes how sensitive the preactivation is under the change of the hyperparameters of the SaS distribution. Because the mean and variance, G_i^{l+1} and Δ_i^{l+1}, are functions of the hyperparameters, the second derivative for each hyperparameter can be derived similarly as follows:

$$\frac{\partial z_i^{l+1}}{\partial m_{ki}^l} = \frac{(1 - \pi_{ki}^l) h_k^l}{\sqrt{N_l}} + \frac{\mu_{ki}^l \pi_{ki}^l}{N_l \Delta_i^{l+1}} (h_k^l)^2 \epsilon_i^{l+1}, \tag{7a}$$

$$\frac{\partial z_i^{l+1}}{\partial \pi_{ki}^l} = -\frac{m_{ki}^l h_k^l}{\sqrt{N_l}} - \frac{(2\pi_{ki}^l - 1)(m_{ki}^l)^2 + \Xi_{ki}^l}{2 N_l \Delta_i^{l+1}} (h_k^l)^2 \epsilon_i^{l+1}, \tag{7b}$$

$$\frac{\partial z_i^{l+1}}{\partial \Xi_{ki}^l} = \frac{1 - \pi_{ki}^l}{2 N_l \Delta_i^{l+1}} (h_k^l)^2 \epsilon_i^{l+1}. \tag{7c}$$

Equations (6) and (7) share the same form with the preactivation z_i^l [Eq. (3)] due to the reparameterization trick used to handle the uncertainty of weights in the hypothesis space. Therefore, the learning process of our model naturally captures the fluctuation of the hypothesis space, highlighting the significant difference from the standard BP, which computes only a point estimate of the connection weights. In particular, if we enforce $\boldsymbol{\pi} = 0$ and $\boldsymbol{\Xi} = 0$, $\boldsymbol{m}$ becomes identical to the weight configuration, and thus our learning equations will immediately recover the standard BP algorithm [26]. Therefore, our learning protocol can be thought of as a generalized back-propagation (gBP) at the weight distribution level or the candidate-network ensemble level.

Our framework is a cheap way to compute the posterior distribution of the weights given the data, which can be alternatively realized by a Bayesian inference where a variational free energy is commonly optimized through Monte-Carlo samplings [24,27,28]. The optimization of the variational free energy for deep neural networks is computationally challenging, especially for the SaS prior whose entropy has no analytic form as well. In contrast, gBP stores two times more parameters than BP, while other computational steps are exactly the same; the training time is affordable, as a typical deep network training scales linearly with the number of parameters [29].

We remark that, during learning, the spike mass π should be clipped as $\max[0, \min(1, \pi)]$, and the variance $\Xi \leftarrow \max(0, \Xi)$. Learning the SaS model allows for credit assignment to each connection, considering a network ensemble realizing the same task. An effective network can be constructed by sampling the learned SaS distribution.

The size of the hypothesis space can be approximated by the model entropy $S = -\int_{\mathbb{R}^{\mathcal{D}}} P(\boldsymbol{w}) \ln P(\boldsymbol{w}) d\boldsymbol{w}$, where $\mathcal{D}$ is the number of weight parameters in the network. Because the joint distribution of weights is assumed to be factorized across individual connections, $S = \sum_\ell S_\ell$, where the entropy contribution of each individual connection (ℓ) is expressed as [25]

$$S_\ell = -\pi_\ell \ln[\pi_\ell + (1 - \pi_\ell)\mathcal{N}(0|m_\ell, \Xi_\ell)] - \frac{(1 - \pi_\ell)}{\mathcal{B}} \sum_s \Gamma(\epsilon_s), \tag{8}$$

where $\mathcal{B}$ denotes the number of standard Gaussian random variables ϵ_s, and $\Gamma(\epsilon_s) = \ln[\pi_\ell \delta(m_\ell + \sqrt{\Xi_\ell}\epsilon_s) + [(1 - \pi_\ell)/\sqrt{\Xi_\ell}]\mathcal{N}(\epsilon_s|0, 1)]$. If $\pi_\ell = 0$, the entropy S_ℓ can

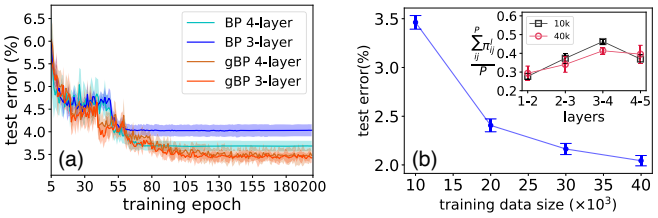

FIG. 2. Properties of gBP in testing performances on unseen data. (a) Training trajectories of a network architecture of three or four layers. The architecture is defined as 784-100-100-10 (four layers) or 784-100-10 (three layers), where each number indicates the corresponding layer width. Networks are trained on the data size of $T = 10^4$ images and tested on another unseen-data size of $V = 10^4$ images. The fluctuation is computed from five independent runs. (b) Test errors of gBP versus training data size. The error bar characterizes the fluctuation across ten independently sampled network architectures from the SaS distribution. The same network of four layers as in (a) is used. The inset shows the sparsity per connection as a function of layers, for which networks of five layers are used and the hidden layer width is still 100 nodes. Each marker is an average over ten independent runs. The "k" in the inset means the training data size in the unit of 10^3.

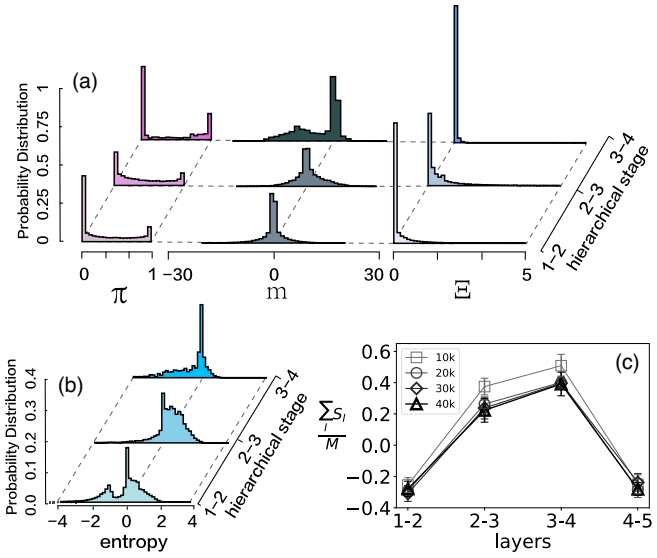

FIG. 3. Statistical properties of trained four-layer networks with gBP. Unless stated otherwise, other training conditions are the same as in Fig. 2(a). Distribution of hyperparameters $(\boldsymbol{\pi}, \mathbf{m}, \boldsymbol{\Xi})$ for a typical trained network. (b) Distribution of connection entropy S_ℓ for a typical trained network. (c) Entropy per connection versus layers ($\mathcal{B} = 100$). Different training data sizes are considered, and the result is averaged over ten independent runs of five-layer networks.

be analytically computed as $\frac{1}{2}\ln(2\pi e\Xi_\ell)$. If $\Xi_\ell = 0$, the Gaussian distribution reduces to a Dirac delta function, and the entropy becomes an entropy of discrete random variables.

Results.—Despite working at the synaptic weight distribution level, gBP can reach a similar even better test accuracy than that of BP [Fig. 2(a)]. As expected, the test error for gBP decreases with training data size [Fig. 2(b)]. The error bar implies that *effective networks* sampled from the learned SaS distribution still yield accurate predictions, confirming the network ensemble assumption. Our theory also reveals that the sparsity, obtained from the statistics of $\{\pi_{ij}^l\}$, grows first and then decreases [the inset of Fig. 2(b)], suggesting that the actual working network does not use up all synaptic connections, consistent with empirical observations of network compression [10,17,30] and theoretical studies of toy models [20,31]. Along hierarchical stages of the deep neural network, the initial stage is responsible for encoding, and therefore the sparsity must be low to ensure that there is sufficient space for encoding the important information. At the middle stage, the encoded information is recoded through hidden representations, suggesting that the noisy information is further distilled, which may explain why the sparsity goes up. Finally, to guarantee the feature-selective information being extracted, the sparsity must drop to yield an accurate classification. Therefore, our model can interpret the deep learning as an encoding-recoding-decoding process, as also argued in a biological neural hierarchy [32].

To inspect more carefully what distribution of hyperparameters gBP learned, we plot Fig. 3(a) showing the evolution of the distribution across the hierarchical stages. First, the spike mass $\boldsymbol{\pi}$ has a U-shaped distribution. One extreme is at $\pi = 0$, suggesting that the corresponding connection carries feature-selective information and thus cannot be pruned, while the other extreme is at $\pi = 1$, suggesting that the corresponding connection can be completely pruned. Apart from these two extremes, there exists a relatively small number of connections that can be present or absent with certain probabilities. These connections may reflect nuisance factors in sensory inputs [33]. The mean of the slab distribution has a relatively broad distribution, but the peak is located around zero; an L-shaped distribution of the variance hyperparameter is observed. Note that if $\Xi = 0$, the SaS distribution reduces to a Bernoulli distribution with two delta peaks. Due to Eq. (7b), the point solution ($\boldsymbol{\pi} = 0$, $\Xi = 0$) is not a stable attractor for gBP [25]. Moreover, as the training data size increases, the variance per connection displays a nonmonotonic behavior with a residual value depending on the hierarchical stage [25].

We can use the entropy S_ℓ for each connection to characterize the variability of the weight value. Note that the entropy can be negative for continuous random variables. The peak at zero [Fig. 3(b)] indicates that a large number of connections are deterministic, including two cases: (i) $\pi = 1$ [unimportant (UIP) weight]; (ii) $\pi = 0$ and $\Xi = 0$ [very important (VIP) weight]. Figure 3(c) shows that the entropy per connection grows first and then decreases. This nonmonotonic behavior is the same as that of the sparsity in Fig. 2(b). The large entropy at the middle

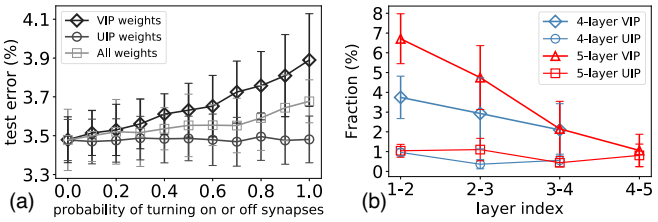

FIG. 4. Effects of targeted-weight perturbation on test performances. Training conditions are the same as in Fig. 2, and the result is averaged over ten independent runs. (a) VIP weights and all weights (without weight categorization) are stochastically turned off, keeping the same number, while UIP weights are stochastically turned on and sampled from a standard Gaussian distribution. The perturbation is carried out at the first hierarchical stage of a four-layer network. (b) The fraction of VIP and UIP weights between neighboring layers.

stage suggests that during the recoding process, the network has more degrees of freedom to manipulate the hypothesis space of the computational task [34]. Furthermore, more training data reduce the uncertainty (yet not to zero) until saturation [Fig. 3(c)].

Finally, our mean-field framework can be used to explore effects of targeted-weight perturbation [Fig. 4(a)], which previous studies of random deep models could not address [20,31]. VIP weights play a significant role in determining the generalization capability, while turning on UIP weights does not impair the performance. A random dilution of all weights behaves mildly in between. When going deeper, the perturbation effect becomes less evident (not shown) due to the lower number of VIP weights [Fig. 4(b)].

Conclusion.—In this Letter, we propose a statistical model of learning credit assignment in deep neural networks, resulting in an optimal random subnetwork ensemble explaining the behavior output of the hierarchical system. The model can be solved by using field methods, yielding a practical way to visualize consumed resources of connections and an ensemble of random weights—two key factors affecting the emergent decision-making behavior of the network. Our framework can be applied to a more challenging CIFAR-10 dataset [35] and obtains qualitatively similar results [25]. Our model thus provides deep insights toward understanding many recent interesting empirical observations of random templates in both artificial and biological neural networks [7–9,12,36]. The model also provides a principled method for network compression, saving memory and computation demands [17], and could also have implications for the continual learning of sequential tasks, where those important connections for old tasks are always protected to learn new tasks [30,37].

As artificial neural networks become increasingly important to model the brain [5,12], our current study of artificial neural networks may provide cues for addressing how the brain solves the credit assignment problem. Other promising directions include considering correlations among weights and generalization of this framework to the temporal credit assignment, where spatiotemporal information is considered in training recurrent neural networks [38–40].

This research was supported by the National Key R&D Program of China (2019YFA0706302), the start-up budget 74130-18831109 of the 100-talent program of Sun Yat-sen University, and the NSFC (Grant No. 11805284).

*huanghp7@mail.sysu.edu.cn

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, in *Advances in Neural Information Processing Systems 25*, edited by P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger (Curran Associates Inc., Red Hook, 2012), pp. 1097–1105.
[2] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates *et al.*, arXiv:1412.5567.
[3] K. He, X. Zhang, S. Ren, and J. Sun, in *2015 IEEE International Conference on Computer Vision (ICCV)* (IEEE Computer Society, Santiago, 2015), pp. 1026–1034.
[4] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, Nature (London) **550**, 354 (2017).
[5] B. A. Richards, T. P. Lillicrap, P. Beaudoin, Y. Bengio, R. Bogacz, A. Christensen, C. Clopath, R. P. Costa, A. de Berker, S. Ganguli *et al.*, Nat. Neurosci. **22**, 1761 (2019).
[6] F. H. Sinz, X. Pitkow, J. Reimer, M. Bethge, and A. S. Tolias, Neuron **103**, 967 (2019).
[7] J. Frankle and M. Carbin, arXiv:1803.03635, in ICLR 2019.
[8] H. Zhou, J. Lan, R. Liu, and J. Yosinski, arXiv:1905.01067, in NeurIPS 2019.
[9] V. Ramanujan, M. Wortsman, A. Kembhavi, A. Farhadi, and M. Rastegari, arXiv:1911.13299.
[10] A. S. Morcos, H. Yu, M. Paganini, and Y. Tian, arXiv:1906.02773, in NeurIPS 2019.
[11] A. Gaier and D. Ha, arXiv:1906.04358, in NeurIPS 2019.
[12] S. Baek, M. Song, J. Jang, G. Kim, and S.-B. Paik, bioRxiv (2019) https://www.biorxiv.org/content/early/2019/11/29/857466.
[13] Y. LeCun, The MNIST database of handwritten digits, retrieved from http://yann.lecun.com/exdb/mnist.
[14] V. Nair and G. E. Hinton, in *Proceedings of the 27th International Conference on International Conference on Machine Learning, ICML'10* (Omnipress, USA, 2010), pp. 807–814.
[15] T. J. Mitchell and J. J. Beauchamp, J. Am. Stat. Assoc. **83**, 1023 (1988).
[16] H. Ishwaran and J. S. Rao, Ann. Stat. **33**, 730 (2005).
[17] S. Han, J. Pool, J. Tran, and W. J. Dally, arXiv:1506.02626, in NIPS 2015.
[18] K. Ullrich, E. Meeds, and M. Welling, arXiv:1702.04008, in ICLR 2017.
[19] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, Neural Netw. **113**, 54 (2019).
[20] H. Huang and A. Goudarzi, Phys. Rev. E **98**, 042311 (2018).
[21] P. McClure and N. Kriegeskorte, arXiv:1611.01639 (2016).

[22] S. J. Nowlan and G. E. Hinton, Neural Comput. **4**, 473 (1992).
[23] O. Shayer, D. Levi, and E. Fetaya, arXiv:1710.07739 (2017), in ICLR 2018.
[24] H. Huang, Phys. Rev. E **102**, 030301 (2020).
[25] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevLett.125.178301 for derivations of the entropy formula, transformation, and more simulation results related to the MNIST and CIFAR-10 datasets. Codes are available at https://github.com/Chan-Li/SAS-model.
[26] E. Rumelhart David, E. Hinton Geoffrey, and J. Williams Ronald, Nature (London) **323**, 533 (1986).
[27] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, J. Am. Stat. Assoc. **112**, 859 (2017).
[28] R. J. N. Baldock and N. Marzari, arXiv:1904.04154.
[29] T. J. Sejnowski, Proc. Natl. Acad. Sci. U.S.A. (2020).
[30] A. Mallya and S. Lazebnik, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE Computer Society, Salt Lake City, 2018), pp. 7765–7773.
[31] B. Li and D. Saad, J. Phys. A **53**, 104002 (2020).
[32] X. Pitkow and D. E. Angelaki, Neuron **94**, 943 (2017).
[33] A. Achille and S. Soatto, J. Mach. Learn. Res. **19**, 1 (2018), https://jmlr.csail.mit.edu/papers/v19/17-646.html.
[34] W. Zou and H. Huang, arXiv:2007.08093.
[35] A. Krizhevsky, Learning multiple layers of features from tiny images, Tech. Rep., University of Toronto, Toronto, 2009.
[36] A. M. Zador, Nat. Commun. **10**, 3770 (2019).
[37] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, Proc. Natl. Acad. Sci. U.S.A. **114**, 3521 (2017).
[38] P. J. Werbos, Proc. IEEE **78**, 1550 (1990).
[39] D. Sussillo and L. Abbott, Neuron **63**, 544 (2009).
[40] A. H. Marblestone, G. Wayne, and K. P. Kording, Front. Comput. Neurosci. **10**, 94 (2016).