Space of Functions Computed by Deep-Layered Machines

Alexander Mozeika,[1,*] Bo Li,[2,†] and David Saad[2,‡]
[1] *London Institute for Mathematical Sciences, London W1K 2XF, United Kingdom*
[2] *Nonlinearity and Complexity Research Group, Aston University, Birmingham B4 7ET, United Kingdom*

(Received 21 April 2020; revised 18 August 2020; accepted 9 September 2020; published 12 October 2020)

We study the space of functions computed by random-layered machines, including deep neural networks and Boolean circuits. Investigating the distribution of Boolean functions computed on the recurrent and layer-dependent architectures, we find that it is the same in both models. Depending on the initial conditions and computing elements used, we characterize the space of functions computed at the large depth limit and show that the macroscopic entropy of Boolean functions is either monotonically increasing or decreasing with the growing depth.

DOI: 10.1103/PhysRevLett.125.168301

Deep-layered machines comprise multiple consecutive layers of basic computing elements aimed at representing an arbitrary function, where the first and final layers represent its input and output arguments, respectively. Notable examples include deep neural networks (DNNs) composed of perceptrons [1] and Boolean circuits constructed from logical gates [2]. Being universal approximators [3,4], DNNs have been successfully employed in different machine learning applications [1]. Similarly, Boolean circuits can compute any Boolean function even when constructed from a single gate [5].

While the majority of DNN research focuses on their application in carrying out various learning tasks, it is equally important to establish the space of functions they typically represent for a given architecture and the computing elements used. One way to address such a generic study is to consider a random ensemble of DNNs. The study of random neural networks using methods of statistical physics has played an important role in understanding their *typical* properties for storage capacity and generalization ability [6,7] and properties of energy-based [8–12] and associative memory models [13,14], as well as the links between energy-based models and feed-forward layered machines [15]. In parallel, there have been theoretical studies within the computer science community of the range of Boolean functions generated by random Boolean circuits [16,17]. Both the DNNs and the Boolean circuits share common basic properties.

Characterizing the space of functions computed by random-layered machines is of great importance since it sheds light on their approximation and generalization properties. However, it is also highly challenging due to the inherent recursiveness of computation and randomness in their architecture and/or computing elements. Existing theoretical studies of the function space of deep-layered machines are mostly based on the mean field approach, which allows for a sensitivity analysis of the functions realized by deep-layered machines due to input or parameter perturbations [4,18–20].

To gain a complete and detailed understanding of the function space, we develop a path-integral formalism that directly examines *individual functions* computed. This is carried out by processing *all possible input configurations simultaneously and the corresponding outputs.* For simplicity, we always consider Boolean functions with binary input and output variables.

The main contribution of this Letter is in providing a detailed understanding of the distribution of Boolean functions computed at each layer. It points to the equivalence between recurrent and layer-dependent architectures and consequently to the potential significant reduction in the number of trained free variables. Additionally, the complexity of Boolean functions implemented measured by their entropy, which depends on the number of layers and computing elements used, exhibits a rapid simplification when rectified linear unit (ReLU) components are employed, which arguably explains their generalization successes.

Framework.—The layered machines considered consist of $L+1$ layers, each with N nodes. Node i at layer l is connected to the set of nodes $\{i_1, i_2, \ldots, i_k\}$ of layer $l-1$; its activity is determined by the gate α_i^l, computing a function of k inputs, according to the propagation rule

$$P(S_i^l|\vec{S}^{l-1}) = \delta[S_i^l, \alpha_i^l(S_{i_1}^{l-1}, S_{i_2}^{l-1}, \ldots, S_{i_k}^{l-1})], \qquad (1)$$

where δ is the Dirac or Kronecker delta function, depending on the domain of S_i^l. The probabilistic form of Eq. (1) adopted here is convenient for the generating functional analysis and inclusion of noise [19,21]. We primarily consider two structures here: (i) densely connected models where $k=N$ and node i is connected to all nodes from the previous layer—one such example is the fully connected neural network with $S_i^l = \alpha^l(H_i^l)$, where $H_i^l = \sum_{j=1}^N W_{ij}^l S_j^{l-1}/\sqrt{N} + b_i^l$ is the preactivation field and α^l

is the activation function at layer l (we will mainly focus on the case $b_i^l = 0$; the effect of nonzero bias is discussed in [22]); (ii) sparsely connected models where $k \in O(N^0)$—examples include the sparse neural networks and layered Boolean circuits where α_i^l is a Boolean gate with k inputs, e.g., majority gate.

Consider a binary input vector $\vec{s} = (s_1, \ldots, s_n) \in \{-1, 1\}^n$, which is fed to the initial layer $l = 0$. To accommodate a broader set of functions, we also consider an augmented input vector, e.g., (i) $\vec{S}^I = (\vec{s}, 1)$, which is equivalent to adding a bias variable in the context of neural networks; (ii) $\vec{S}^I = (\vec{s}, -\vec{s}, 1, -1)$, which has been used to construct all Boolean functions [16]. Each node i at layer 0 points to a randomly chosen element of $\vec{S}^I$ such that

$$P^0(\vec{S}^0|\vec{s}) = \prod_{i=1}^{N} P^0[S_i^0|S_{n_i}^I(\vec{s})] = \prod_{i=1}^{N} \delta[S_i^0, S_{n_i}^I(\vec{s})], \quad (2)$$

where $n_i = 1, \ldots, |\vec{S}^I|$ is an index chosen from the flat distribution $P(n_i) = 1/|\vec{S}^I|$.

The computation of the layered machine is governed by the propagator $P(\vec{S}^L|\vec{s}) = \sum_{\vec{S}^{L-1} \ldots \vec{S}^0} P(\vec{S}^0|\vec{s}) \prod_{l=1}^{L} P(\vec{S}^l|\vec{S}^{l-1})$, where each node at layer L computes a Boolean function $\{-1, 1\}^n \to \{-1, 1\}$. When the gates α_i^l or the network topology are *random*, then the layered machine can be viewed as a disordered dynamical system with *quenched* disorder [19,21]. To probe the functions being computed, we consider the simultaneous layer propagation of *all* possible inputs $\vec{s}_\gamma \in \{-1, 1\}^n$, labeled by $\gamma = 1, \ldots, 2^n$ governed by the product propagator $\prod_{\gamma=1}^{2^n} P(\vec{S}_\gamma^L|\vec{s}_\gamma)$. The binary string $\boldsymbol{S}_i^L \in \{-1, 1\}^{2^n}$ represents the Boolean function computed at node i at layer L, as illustrated in Fig. 1. Note that we use the vector notation $\vec{S}^l = (S_1^l, \ldots, S_i^l, \ldots, S_N^l)$ and $\boldsymbol{S}_i^l = (S_{i,1}^l, \ldots, S_{i,\gamma}^l, \ldots, S_{i,2^n}^l)$ to represent the states and functions, respectively. Using the above formalism, the distribution of Boolean functions $\boldsymbol{f}$ computed on the final layer is given by

$$P_N^L(\boldsymbol{f}) = \frac{1}{N} \sum_{i=1}^{N} \left\langle \prod_{\gamma=1}^{2^n} \delta(f_\gamma, S_{i,\gamma}^L) \right\rangle, \quad (3)$$

where components of $\boldsymbol{f}$ satisfy $f_\gamma = f(\vec{s}_\gamma)$, and angular brackets represent the average generated by $\prod_{\gamma=1}^{2^n} P(\vec{S}_\gamma^L|\vec{s}_\gamma)$. To compute $P_N^L(\boldsymbol{f})$ and averages of other macroscopic *observables*, which are expected to be self-averaging for $N \to \infty$ [26], we introduce the disorder-averaged generating functional (GF) $\overline{\Gamma[\{\psi_{i,\gamma}^l\}]} = \overline{\sum_{\{\vec{S}_\gamma^l\}} \prod_\gamma P(\vec{S}_\gamma^0|\vec{S}_\gamma^I) \prod_l P(\vec{S}_\gamma^l|\vec{S}_\gamma^{l-1}) e^{-\mathrm{i} \sum_i \psi_{i,\gamma}^l S_{i,\gamma}^l}}$, where the overline denotes an average over the quenched disorder. To keep the presentation concise, we outline the GF formalism only for DNNs in the following and refer the reader to [22] for the details of the derivation used in Boolean circuits.

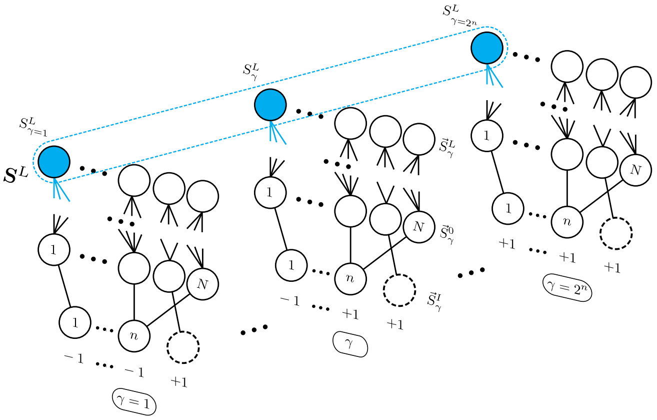

FIG. 1. A deep-layered machine computing all possible 2^n inputs. The direction of computation is from bottom to top. The binary string $\boldsymbol{S}^L \in \{-1, 1\}^{2^n}$ represents the Boolean function computed on the blue nodes of the output layer L. The augmented vector $\vec{S}^I = (\vec{s}, 1)$ is used as an example of input here. The constant 1 is represented by the dashed circle.

Layer-dependent and recurrent architectures.—We focus on two different architectures: layer-dependent architectures, where the gates and/or connections are different from layer to layer, and recurrent, where the gates and connections are shared across all layers. Both architectures represent feed-forward machines that implement input-output mappings.

Specifically, we assume that the weights W_{ij}^l in fully connected DNNs with layer-dependent architectures are independent Gaussian random variables sampled from $\mathcal{N}(0, \sigma^2)$. In DNNs with recurrent architectures, the weights are sampled once and are shared among layers, i.e., $W_{ij}^{l+1} = W_{ij}^l$. We apply the sign activation function in the final layer, i.e., $\alpha^L(h_i^L) = \mathrm{sgn}(h_i^L)$, to ensure that the output of the DNN is Boolean.

We first outline the derivation for fully connected recurrent architectures. It is sufficient to characterize the disorder-averaged GF by introducing cross-layer overlaps $q_{\gamma\gamma'}^{l,l'} = (1/N) \sum_i \overline{\langle S_{i,\gamma}^l S_{i,\gamma'}^{l'} \rangle}$ as order parameters and the corresponding conjugate order parameter $Q_{\gamma\gamma'}^{l,l'}$, which leads to a saddle-point integral $\bar{\Gamma} = \int \{\mathrm{d}\boldsymbol{q} \mathrm{d}\boldsymbol{Q}\} \mathrm{e}^{N\Psi[\boldsymbol{q},\boldsymbol{Q}]}$ with the potential [22]

$$\Psi = \mathrm{i}\mathrm{Tr}\{\boldsymbol{q}\boldsymbol{Q}\} + \sum_{m=1}^{|\vec{S}^I|} P(m) \ln \sum_{\boldsymbol{S}} \int \mathrm{d}\boldsymbol{H} \mathcal{M}_m[\boldsymbol{H}, \boldsymbol{S}], \quad (4)$$

where $\mathcal{M}_m[\boldsymbol{H}, \boldsymbol{S}]$ is an effective single-site measure

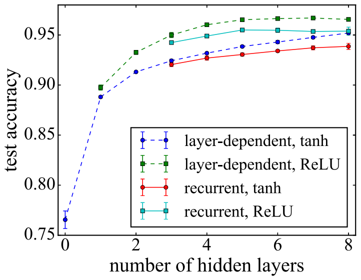

FIG. 2. Test accuracy of trained fully connected DNNs applied on the MNIST dataset. Images have been downsampled by a factor of 2 to reduce training time, and each hidden layer has 128 nodes. Each data point is averaged over 5 random initializations. The accuracies of recurrent architectures, with weight sharing *between* hidden layers, are comparable to those of layer-dependent architectures.

$$\mathcal{M}_m = \mathrm{e}^{-\mathrm{i}\sum_{l,\gamma}\psi_\gamma^l S_\gamma^l - \mathrm{i}\sum_{ll',\gamma\gamma'} Q_{\gamma\gamma'}^{l,l'} S_\gamma^l S_{\gamma'}^{l'}} \times \mathcal{N}(\boldsymbol{H}|\boldsymbol{0},\boldsymbol{C}) \prod_{\gamma=1}^{2^n} P^0(S_\gamma^0|S_{m,\gamma}^I) \prod_{l=1}^{L} \delta[S_\gamma^l, \alpha^l(h_\gamma^l)]. \quad (5)$$

Due to weight sharing, the preactivation fields $\boldsymbol{H} = (\boldsymbol{h}^1, \ldots, \boldsymbol{h}^L)$, where $\boldsymbol{h}^l \in \mathbb{R}^{2^n}$, are governed by the Gaussian distribution $\mathcal{N}(\boldsymbol{H}|\boldsymbol{0},\boldsymbol{C})$ and correlated across layers with covariance $[\boldsymbol{C}]_{\gamma\gamma'}^{l,l'} = \sigma^2 q_{\gamma\gamma'}^{l-1,l'-1}$. Setting ψ_γ^l to zero and differentiating Ψ with respect to $\{q_{\gamma\gamma'}^{l,l'}, Q_{\gamma\gamma'}^{l,l'}\}$ yields the saddle point of the potential Ψ dominating $\bar{\Gamma}$ for $N \to \infty$, at which the conjugate order parameters $Q_{\gamma\gamma'}^{l,l'}$ vanish [22], leading to

$$q_{\gamma\gamma'}^{l,l'} = \begin{cases} \sum_m P(m)\langle S_\gamma^l S_{\gamma'}^0 \rangle_{\mathcal{M}_m}, & l' = 0 \\ \int \mathrm{d}\boldsymbol{H} \alpha^l(h_\gamma^l)\alpha^{l'}(h_{\gamma'}^{l'})\mathcal{N}(\boldsymbol{H}|\boldsymbol{0},\boldsymbol{C}). & l' > 0 \end{cases} \quad (6)$$

Notice that in the above Gaussian average, all preactivation fields but the pair $\{h_\gamma^l, h_{\gamma'}^{l'}\}$ can be integrated out, reducing it to a tractable two-dimensional integral.

The GF analysis can be performed similarly for layer-dependent architectures. Here the result has the same form as Eq. (6) with $q_{\gamma\gamma'}^{l,l'} = \delta_{l,l'} q_{\gamma\gamma'}^{l,l'}$, i.e., the overlaps between different layers are absent [22], implying $[\boldsymbol{C}]_{\gamma\gamma'}^{l,l'} = \sigma^2 \delta_{l-1,l'-1} q_{\gamma\gamma'}^{l-1,l'-1}$ for the covariances of preactivation fields. In this case, we denote the equal-layer covariance matrix as $\boldsymbol{c}^l \coloneqq \boldsymbol{C}^{l,l}$.

We remark that the behavior of DNNs with layer-dependent architectures in the limit of $N \to \infty$ can also be studied by mapping to Gaussian processes [4,18,27]. However, it is not clear if such analysis is possible in the highly correlated recurrent case while the GF or path-integral framework is still applicable [28–30].

Marginalizing the effective single-site measure in Eq. (5) gives rise to the distribution of Boolean functions $\boldsymbol{f} \in \{-1,1\}^{2^n}$ computed at layer L of DNNs with recurrent architectures

$$P^L(\boldsymbol{f}) = \int \mathrm{d}\boldsymbol{h}\mathcal{N}(\boldsymbol{h}|\boldsymbol{0},\boldsymbol{c}^L) \prod_{\gamma=1}^{2^n} \delta[f_\gamma, \alpha^L(h_\gamma)], \quad (7)$$

where in the above the element of the covariance matrix is $[\boldsymbol{c}^L]_{\gamma\gamma'} = [\boldsymbol{C}]_{\gamma\gamma'}^{L,L} = \sigma^2 q_{\gamma\gamma'}^{L-1,L-1}$. Note that the *physical meaning* of $P^L(\boldsymbol{f})$ is the distribution of Boolean functions defined in Eq. (3) averaged over disorder $P^L(\boldsymbol{f}) = \lim_{N\to\infty} \overline{P_N^L(\boldsymbol{f})}$.

Moreover, Eq. (7) also applies to layer-dependent architectures since the *equal-layer* covariance matrix $\boldsymbol{c}^L$ is the same in two scenarios. Therefore, we arrive at the first important conclusion that *the typical sets of Boolean functions computed at the output layer L by the layer-dependent and recurrent architectures are identical*. Furthermore, if the gate functions α^l are odd, then it can be shown that all the cross-layer overlaps $q_{\gamma\gamma'}^{l,l'}$ of the recurrent architectures vanish, implying the statistical equivalence of the hidden layer activities to the layered architectures as well [22].

A similar GF analysis can be applied to sparsely connected Boolean circuits constructed from a single Boolean gate α, keeping in mind that distributions of gates can be easily accommodated. In such models, the source of disorder are random connections. In layer-dependent architectures, a gate is connected randomly to exactly $k \in O(N^0)$ gates from the previous layer and this connectivity pattern is changing from layer to layer. In recurrent architectures, on the other hand, the random connections are sampled once and the connectivity pattern is shared among layers. Note that in Boolean circuits, the activities at every layer $\boldsymbol{S}_i^l$ *always* represent a Boolean function. For layer-dependent architectures, investigating the distribution of activities gives rise to

$$P^{l+1}(\boldsymbol{f}) = \sum_{\boldsymbol{f}_1,\ldots,\boldsymbol{f}_k} \left\{ \prod_{j=1}^{k} P^l(\boldsymbol{f}_j) \right\} \times \prod_{\gamma=1}^{2^n} \delta[f_\gamma, \alpha(f_{1,\gamma},\ldots,f_{k,\gamma})], \quad (8)$$

which describes how the probability of the Boolean function $\boldsymbol{f} \in \{-1,1\}^{2^n}$ is evolving from layer to layer [22,31]. We note that for recurrent architecture the equation for the probability of Boolean functions computed is exactly the same as above [22], suggesting that in random Boolean circuits *the typical sets of Boolean functions computed on layers in the layer-dependent and recurrent architectures are identical*. Note that the coupling asymmetry plays a crucial role in this equivalence property [22,32,33].

The equivalence between two architectures points to a potential reduction in the number of free parameters in layered machines by weight sharing or connectivity sharing among layers, useful in devices with limited computation resources [34]. For illustration, we consider the image recognition task of Modified National Institute of Standards and Technology (MNIST) handwritten digit data [35] using DNNs with both layer-dependent and recurrent architectures (weight shared from hidden to hidden layers only; for details see [22]). The experiment shown in Fig. 2 demonstrates the feasibility of using recurrent architectures to perform image classification tasks with a slightly lower accuracy but significant saving in the number of trained parameters.

Boolean functions computed at large depth.—We consider the typical Boolean functions computed in random-layered machines by examining $P^L(\boldsymbol{f})$ in the large depth limit $L \to \infty$ for specific gates in the following examples.

In DNNs using the ReLU activation function $\alpha^l(x) = \max(x, 0)$, in the hidden layers (the sign activation function is always used in the output layer), which is commonly used in applications, all covariance matrix elements $[\boldsymbol{c}^L]_{\gamma\gamma'}$ in the Eq. (7) converge to the same value in the limit $L \to \infty$, implying that all components of the preactivation field vector $\boldsymbol{h}$ are also the same and hence the components of $\boldsymbol{f}$ are identical. Therefore, random deep ReLU networks compute only *constant* Boolean functions in the infinite depth limit, echoing recent findings of a bias toward simple functions in random DNNs constructed from ReLUs [22], which arguably plays a role in their generalization ability [36,37].

In DNNs using sign activation function also in hidden layers, i.e., Eq. (1) enforces the rule $S_i^l = \mathrm{sgn}(\sum_j W_{ij}^l S_j^{l-1}/\sqrt{N})$, those cross-pattern overlaps $q_{\gamma\gamma'}^l = (1/N)\sum_i \overline{\langle S_{i,\gamma}^l S_{i,\gamma'}^l \rangle}$ satisfying $|q_{\gamma\gamma'}^l| < 1$ monotonically decrease with an increasing number of layers and vanish as $l \to \infty$, such "chaotic" nature of dynamics also holds in random DNNs with other sigmoidal activation functions such as the error and hyperbolic tangent functions [4,27]. The consequences of this behavior is that for the input vector $\vec{S}^I = \vec{s}$, $P^L(\boldsymbol{f})$ is uniform on the set of *all odd* functions [22], i.e., functions satisfying $f(-\vec{s}) = -f(\vec{s})$. Furthermore, for $\vec{S}^I = (\vec{s}, 1)$, $P^L(\boldsymbol{f})$ is uniform on the set of *all* Boolean functions [22].

For Boolean circuits, there are also scenarios where the distribution $P^L(\boldsymbol{f})$ has a single Boolean function in its support or it is uniform over some set of functions [16,17,38]. The latter depends on the gates α used in Eq. (1) and input vector $\vec{S}^I$. For example, in the AND gate with $\alpha(S_1, S_2) = \mathrm{sgn}(S_1 + S_2 + 1)$ or the OR gate with $\alpha(S_1, S_2) = \mathrm{sgn}(S_1 + S_2 - 1)$ [22], their output is biased, respectively, toward $+1$ or -1 [16,22,38]. The consequence of the latter is that the distribution $P^L(\boldsymbol{f})$ has only a *single* Boolean function in its support [22,38]. On the other hand, when the majority gate $\alpha(S_1, \ldots, S_k) = \mathrm{sgn}(\sum_{j=1}^k S_j)$, which is balanced $\sum_{S_1,\ldots,S_k} \alpha(S_1, \ldots, S_k) = 0$ and nonlinear 39]], is used with the input vector $\vec{S}^I = (\vec{s}, -\vec{s}, 1, -1)$, then the distribution $P^L(\boldsymbol{f})$ is uniform over *all Boolean functions* [38], which is consistent with the result of [16].

Entropy of Boolean functions.—Having considered the distribution of Boolean functions for a few different examples, we observed that random-layered machines either reduce to a single Boolean function or compute all (or a subset of) functions with a uniform probability on the layer L, as $L \to \infty$. We note that for the Shannon entropy over Boolean functions $\mathcal{H}^L = -\sum_{\boldsymbol{f}} P^L(\boldsymbol{f}) \log P^L(\boldsymbol{f})$, these two scenarios saturate its lower and upper bounds, respectively, given by 0 and $2^n \log 2$. Thus, the entropy $\mathcal{H}^L$ can be seen, at least intuitively, as a measure of function space complexity.

In Fig. 3, we study the entropy $\mathcal{H}^L$, computed using Eqs. (7) and (8), as a function of the depth L in random-layered machines constructed from different activation functions or gates and computing different inputs. The initial increase in entropy after layer $L = 0$, seen in

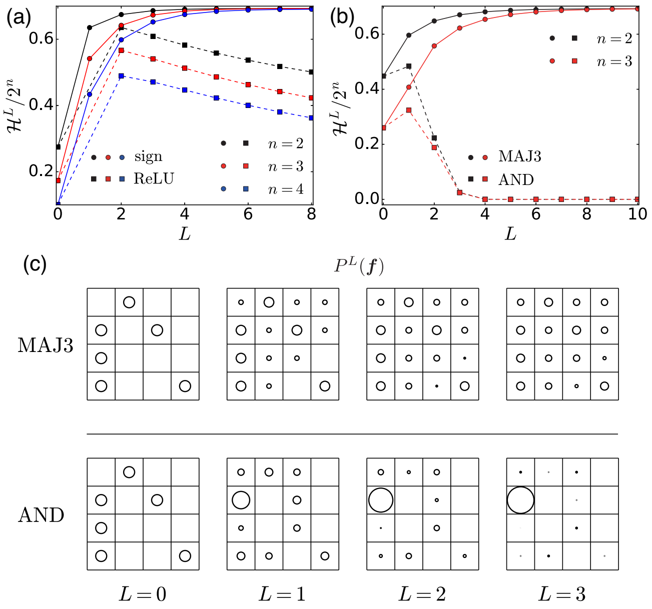

FIG. 3. Normalized entropy and distribution of functions of deep-layered machines. (a) Normalized entropy $\mathcal{H}^L/2^n$ of Boolean functions computed by DNNs with sign or ReLU activation in the hidden layers as a function of the network depth L; the initial condition is set as $\vec{S}^I = (\vec{s}, 1)$. (b) $\mathcal{H}^L/2^n$ vs L for Boolean circuits constructed by MAJ3 or the AND gate with initial condition $\vec{S}^I = (\vec{s}, -\vec{s}, 1, -1)$. (c) The distribution of Boolean functions $P^L(\boldsymbol{f})$ computed by Boolean circuits with two inputs $n = 2$ (the number of all possible functions is 16) is represented by the sizes of circles on a 4×4 grid. Upper panel: MAJ3-gate-based circuits, in which more functions are created at larger depth L and $P^L(\boldsymbol{f})$ converges to a uniform distribution. Lower panel: AND-gate-based circuits, in which new functions are created from $L = 0$ to $L = 1$, while $P^L(\boldsymbol{f})$ converges to a distribution with supports in a single Boolean function as network depth increases.

Figs. 3(a) and 3(b), can be explained by the properties of gates used and the initial set of (simple) Boolean functions at layer $L = 0$; functions from the layer $L = 0$ are "copied" onto layer $L = 1$, while new functions are also created, as illustrated in Figs. 3(c) and 3(d). Note that the minimal depth in ReLU networks to produce a Boolean function is $L = 2$. The dependence of entropy $\mathcal{H}^L$ on L after the initial increase depends on the specific gate functions used. For the ReLU activation function in DNNs and the AND gate in Boolean circuits, the entropies $\mathcal{H}^L$ monotonically decrease with L, suggesting that sizes of sets of typical Boolean functions computed are decreasing with increasing numbers of layers L. Random initialization of layered machines with such gates or activation functions serves as a biasing prior toward a more restricted set of functions [36,37]. On the other hand, for balanced gates, with appropriate initial conditions, e.g., sign in DNNs and majority vote in Boolean circuits, the entropy $\mathcal{H}^L$ is monotonically increasing with the depth L, indicating that the sizes of sets of the typical Boolean functions computed are increasing.

In summary, we present an analytical framework to examine Boolean functions represented by random deep-layered machines by considering *all possible inputs simultaneously* and applying the generating functional analysis to compute various relevant macroscopic quantities. We derived the probability of Boolean functions computed on the output nodes. Surprisingly, we discover that the typical sets of Boolean functions computed by the layer-dependent and recurrent architectures are identical. It points to the possibility of computing complex functions with a reduced number of parameters by weight or connection sharing, as showcased in an image classification experiment. We also study the Boolean functions computed by specific random-layered machines. Biased activation functions (e.g., ReLU) or biased Boolean gates (e.g., AND/OR) can lead to more restricted typical sets of Boolean functions found at deeper layers, which may explain their generalization ability. On the other hand, balanced activation functions (e.g., sign) or Boolean gates (e.g., majority) complemented with appropriate initial conditions lead to a uniform distribution on all Boolean functions at the infinite depth limit. It will be interesting to investigate the functions realized by different DNN architectures with structured data and by different learning algorithms [7,40–43].

We also showed the monotonic behavior of the entropy of Boolean functions as a function of depth, which is of interest in the field of computer science. We envisage that the insights gained and the methods developed will facilitate further study of deep-layered machines.

B. L. and D. S. acknowledge support from the Leverhulme Trust (RPG-2018-092), European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie Grant Agreement No. 835913. D. S. acknowledges support from the EPSRC program grant TRANSNET (EP/R035342/1).

[*]Corresponding author.
alexander.mozeika@kcl.ac.uk
[†]Corresponding author.
b.li10@aston.ac.uk
[‡]Corresponding author.
d.saad@aston.ac.uk

[1] Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, Nature (London) **521**, 436 (2015).

[2] R. O'Donnell, *Analysis of Boolean Functions* (Cambridge University Press, New York, 2014).

[3] K. Hornik, M. Stinchcombe, and H. White, Multilayer feedforward networks are universal approximators, Neural Netw. **2**, 359 (1989).

[4] B. Poole, S. Lahiri, M. Raghu, J. Sohl-Dickstein, and S. Ganguli, in Exponential expressivity in deep neural networks through transient chaos, edited by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, *Advances in Neural Information Processing Systems* Vol. 29 (Curran Associates, Inc., New York, 2016), pp. 3360–3368.

[5] N. Nisan and S. Schocken, *The Elements of Computing Systems: Building a Modern Computer from First Principles* (MIT Press, Cambridge, 2008).

[6] A. Engel and C. Van den Broeck, *Statistical Mechanics of Learning* (Cambridge University Press, New York, 2001).

[7] D. Saad, *On-Line Learning in Neural Networks* (Cambridge University Press, New York, 1998).

[8] E. Agliari, A. Barra, A. Galluzzi, F. Guerra, and F. Moauro, Multitasking Associative Networks, Phys. Rev. Lett. **109**, 268101 (2012).

[9] H. Huang and T. Toyoizumi, Advanced mean-field theory of the restricted Boltzmann machine, Phys. Rev. E **91**, 050101 (2015).

[10] M. Gabrié, E. W. Tramel, and F. Krzakala, in Training restricted Boltzmann machine via the Thouless-Anderson-Palmer free energy, edited by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, *Advances in Neural Information Processing Systems* Vol. 28 (Curran Associates, Inc., New York, 2015), pp. 640–648.

[11] M. Mézard, Mean-field message-passing equations in the Hopfield model and its generalizations, Phys. Rev. E **95**, 022117 (2017).

[12] A. Barra, G. Genovese, P. Sollich, and D. Tantari, Phase diagram of restricted Boltzmann machines and generalized Hopfield networks with arbitrary priors, Phys. Rev. E **97**, 022310 (2018).

[13] J. J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, Proc. Natl. Acad. Sci. U.S.A. **79**, 2554 (1982).

[14] J. A. Hertz, A. S. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation* (Addison-Wesley, Wokingham, 1991).

[15] J. J. Hopfield, Learning algorithms and probability distributions in feed-forward and feed-back networks, Proc. Natl. Acad. Sci. U.S.A. **84**, 8429 (1987).

[16] P. Savický, Random Boolean formulas representing any boolean function with asymptotically equal probability, Discrete Math. **83**, 95 (1990).

[17] A. Brodsky and N. Pippenger, The Boolean functions computed by random boolean formulas or how to grow the right function, Random Struct. Algorithms **27**, 490 (2005).

[18] J. Lee, J. Sohl-dickstein, J. Pennington, R. Novak, S. Schoenholz, and Y. Bahri, Deep neural networks as gaussian processes, in *Proceedings of the 6th International Conference on Learning Representations, Vancouver* (2018).

[19] B. Li and D. Saad, Exploring the Function Space of Deep-Learning Machines, Phys. Rev. Lett. **120**, 248301 (2018).

[20] B. Li and D. Saad, Large deviation analysis of function sensitivity in random deep neural networks, J. Phys. A **53**, 104002 (2020).

[21] A. Mozeika, D. Saad, and J. Raymond, Computing with Noise: Phase Transitions in Boolean Formulas, Phys. Rev. Lett. **103**, 248701 (2009).

[22] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevLett.125.168301 for details, which includes Refs. [23–25].

[23] B. Derrida, E. Gardner, and A. Zippelius, An exactly solvable asymmetric neural network model, Europhys. Lett. **4**, 167 (1987).

[24] R. Kree and A. Zippelius, Continuous-time dynamics of asymmetrically diluted neural networks, Phys. Rev. A **36**, 4421 (1987).

[25] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, in *Proceedings of the 3rd International Conference on Learning Representations, San Diego* (2015).

[26] M. Mézard, G. Parisi, and M. Virasoro, *Spin Glass Theory and Beyond: An Introduction to the Replica Method and its Applications* (World Scientific Publishing Co Inc, Singapore, 1987), Volume 9.

[27] G. Yang and H. Salman, A fine-grained spectral perspective on neural networks, arXiv:1907.10599.

[28] A. C. C. Coolen, Chapter 15 statistical mechanics of recurrent neural networks. II—Dynamics, edited by F. Moss and S. Gielen, in *Neuro-Informatics and Neural Modelling*, Volume 4 of Handbook of Biological Physics (North-Holland, Amsterdam, 2001), pp. 619–684.

[29] T. Toyoizumi and H. Huang, Structure of attractors in randomly connected networks, Phys. Rev. E **91**, 032802 (2015).

[30] A. Crisanti and H. Sompolinsky, Path integral approach to random neural networks, Phys. Rev. E **98**, 062120 (2018).

[31] Viewing the layers as time steps, the functions can be seen as molecules of gas undergoing k-body collisions.

[32] B. Cessac, Increase in complexity in random neural networks, J. Phys. I (France) **5**, 409 (1995).

[33] J. P. L. Hatchett, B. Wemmenhove, I. Pérez Castillo, T. Nikoletopoulos, N. S. Skantzos, and A. C. C. Coolen, Parallel dynamics of disordered ising spin systems on finitely connected random graphs, J. Phys. A **37**, 6201 (2004).

[34] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, Model compression and acceleration for deep neural networks: The principles, progress, and challenges, IEEE Signal Process. Mag. **35**, 126 (2018).

[35] Y. LeCun, C. Cortes, and C. J. Burges, The MNIST Database of Handwritten Digits, 1998, http://yann.lecun.com/exdb/mnist/.

[36] G. Valle-Perez, C. Q. Camargo, and A. A. Louis, Deep learning generalizes because the parameter-function map is biased towards simple functions, in *Proceedings of the 7th International Conference on Learning Representations, New Orleans* (2019).

[37] G. De Palma, B. Kiani, and S. Lloyd, in Random deep neural networks are biased towards simple functions, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, *Advances in Neural Information Processing Systems* Vol. 32 (Curran Associates, Inc., New York, 2019), pp. 1962–1974.

[38] A. Mozeika, D. Saad, and J. Raymond, Noisy random boolean formulae: A statistical physics perspective, Phys. Rev. E **82**, 041112 (2010).

[39] Since $\{S_j\}$ are binary variables in the context of Boolean circuits, linearity is defined in the finite field $GF(2)$ [16,17].

[40] S. Goldt, M. Mézard, F. Krzakala, and L. Zdeborová, Modelling the influence of data structure on learning in neural networks: the hidden manifold model, arXiv:1909.11500 [Phys. Rev. X (to be published)].

[41] M. Pastore, P. Rotondo, V. Erba, and M. Gherardi, Statistical learning theory of structured data, Phys. Rev. E **102**, 032119 (2020).

[42] P. Rotondo, M. Pastore, and M. Gherardi, Beyond the Storage Capacity: Data-Driven Satisfiability Transition, Phys. Rev. Lett. **125**, 120601 (2020).

[43] L. Zdeborová, Understanding deep learning is also a job for physicists, Nat. Phys. **16**, 602 (2020).