

Exact distribution of the output of a deep-layered machine

Thomas M. A. Fink

London Institute for Mathematical Sciences, Royal Institution, 21 Albemarle St, London W1S 4BS, UK

Deep-layered machines, in which each node computes a Boolean function of all nodes below it, underpin deep learning and digital computation. Yet the statistics of their global output function remain poorly understood. We derive the exact finite-depth distribution of the output of a machine with width k and depth n . The distribution depends only on the Hamming weight of the output, and as n increases favors functions with low and high Hamming weights. But this bias peaks at a crossover depth proportional to 2^k before collapsing onto the constant functions true and false.

The repeated composition of rules lies at the heart of deep learning, digital computation and many emergent phenomena in physics. Although the local rules may be simple, successive layers generate an enormous space of configurations whose statistical structure remains poorly understood.

The composition of Boolean update rules provides a tractable setting in which this problem can be studied exactly. Random compositions of Boolean functions have been studied extensively in the context of Boolean networks [1] and the Kauffman model [2–5]. In those systems, the network architecture is random and the local Boolean functions typically depend on only a few inputs.

Deep-layered machines represent a complementary regime: the architecture is highly ordered, but each node computes an arbitrary Boolean function of all nodes in the layer below. Existing work has focused on the average or asymptotic properties of the function space [6–8]. But an exact characterization of the functions generated by deep-layered architectures remains elusive [9].

Here we solve the problem exactly. We consider a deep-layered machine with k arguments and network depth n , in which every node is assigned a Boolean function chosen uniformly from the 2^{2^k} possibilities (see Fig. 1). We derive the exact finite-depth distribution of the global output as a function of k and n . A key observation is that the probability of an output depends only on its Hamming weight: the number of input configurations for which the function returns true. This symmetry reduces the problem from a Markov process on 2^{2^k} states to one on $2^k + 1$ states. We provide the exact transition matrix that takes the system from depth n to $n + 1$ and obtain the full spectrum of eigenvalues.

Our analysis reveals two competing effects as the depth n increases. First, the distribution of non-constant output functions develops a pronounced bias: functions with low or high Hamming weight become exponentially more likely than those with middling weight. Second, probability steadily accumulates in the constant functions true and false, which eventually dominate the distribution.

Strikingly, these opposing tendencies give rise to a crossover network depth of order 2^k . At this scale the distribution of the non-constant outputs is maximally biased. Beyond it, the distribution collapses to a coin toss

between true and false; additional layers cease to generate meaningful functional diversity.

We test our theoretical predictions through exhaustive enumeration for small systems and large-scale Monte Carlo sampling for larger ones. Across all accessible values of k and n , our measurements are in exact agreement with our theory or, in the case of sampling, within statistical significance.

Exact distribution

Throughout this paper we use f to indicate the local Boolean functions assigned to the network nodes, and F to indicate the global Boolean function, or output, of the entire network (see Fig. 1). Whereas f is a function of the k inputs immediately below it, F is a function of the k terminal arguments a, b, \dots . The goal of this paper is to understand the distribution of $F(a, b, \dots)$ given a random assignment of Boolean functions to the f s.

We start by working out the distribution of the output for small values of k and n . For $k = 1, 2$ and 3 arguments, the 4, 16 and 256 possible output functions are shown on the left of Table I. The outputs are indicated algebraically as well as via their truth tables. A truth table is the binary string of length 2^k that determines the output of the function for all possible combinations of

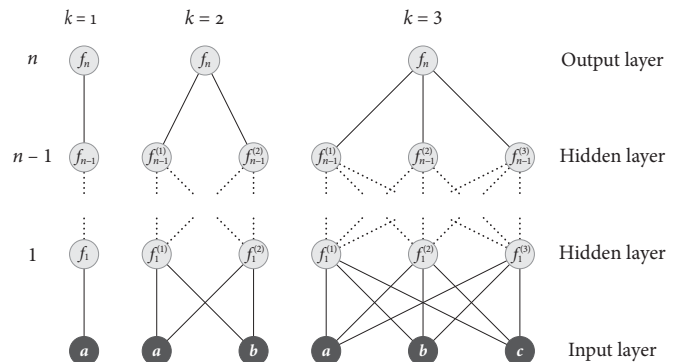


FIG. 1: **Deep-layered machines.** In a network of k arguments (a, b, \dots), each node depends on all k of the arguments below it, and so on, for a total of n levels. The goal of this paper is to determine the probability distribution of the global output Boolean function $F(a, b, \dots)$ given a random assignment of local Boolean functions to the f s at the light gray nodes.

A ($k = 1$)			Probability of F			
Output $F(a)$	w		$n=1$	$n=2$	$n=3$	$n=4$
false	00	0	$\frac{1}{4}$	$\frac{6}{4^2}$	$\frac{28}{4^3}$	$\frac{120}{4^4}$
a	10	1		$\frac{2}{4^2}$	$\frac{4}{4^3}$	$\frac{8}{4^4}$
\bar{a}	01	1	$\frac{1}{4}$	$\frac{6}{4^2}$	$\frac{28}{4^3}$	$\frac{120}{4^4}$
true	11	2		$\frac{2}{4^2}$	$\frac{4}{4^3}$	$\frac{8}{4^4}$

B ($k = 2$)			Probability of F			
Output $F(a, b)$	w		$n=1$	$n=2$	$n=3$	$n=4$
false	0000	0	$\frac{1}{16}$	$\frac{680}{16^3}$	$\frac{261056}{16^5}$	$\frac{83663360}{16^7}$
ab	1000	1		$\frac{216}{16^3}$	$\frac{42048}{16^5}$	$\frac{8087040}{16^7}$
$a\bar{b}$	0100	1	$\frac{1}{16}$	$\frac{168}{16^3}$	$\frac{31680}{16^5}$	$\frac{6068736}{16^7}$
$\bar{a}b$	0010	1		$\frac{216}{16^3}$	$\frac{42048}{16^5}$	$\frac{8087040}{16^7}$
$\bar{a}\bar{b}$	0001	1	$\frac{1}{16}$	$\frac{680}{16^3}$	$\frac{261056}{16^5}$	$\frac{83663360}{16^7}$
a	1100	2		$\frac{216}{16^3}$	$\frac{42048}{16^5}$	$\frac{8087040}{16^7}$
\bar{a}	0011	2	$\frac{1}{16}$	$\frac{168}{16^3}$	$\frac{31680}{16^5}$	$\frac{6068736}{16^7}$
b	1010	2		$\frac{216}{16^3}$	$\frac{42048}{16^5}$	$\frac{8087040}{16^7}$
\bar{b}	0101	2	$\frac{1}{16}$	$\frac{680}{16^3}$	$\frac{261056}{16^5}$	$\frac{83663360}{16^7}$
$a \oplus b$	0110	2		$\frac{216}{16^3}$	$\frac{42048}{16^5}$	$\frac{8087040}{16^7}$
$a \oplus \bar{b}$	1001	2	$\frac{1}{16}$	$\frac{168}{16^3}$	$\frac{31680}{16^5}$	$\frac{6068736}{16^7}$
$a + b$	1110	3		$\frac{216}{16^3}$	$\frac{42048}{16^5}$	$\frac{8087040}{16^7}$
$a + \bar{b}$	1101	3	$\frac{1}{16}$	$\frac{680}{16^3}$	$\frac{261056}{16^5}$	$\frac{83663360}{16^7}$
$\bar{a} + b$	1011	3		$\frac{216}{16^3}$	$\frac{42048}{16^5}$	$\frac{8087040}{16^7}$
$\bar{a} + \bar{b}$	0111	3	$\frac{1}{16}$	$\frac{168}{16^3}$	$\frac{31680}{16^5}$	$\frac{6068736}{16^7}$
true	1111	4		$\frac{216}{16^3}$	$\frac{42048}{16^5}$	$\frac{8087040}{16^7}$

C ($k = 3$)			Probability of F		
Output $F(a, b, c)$	w		$n=1$	$n=2$	$n=3$
00000000	0		$\frac{1}{256}$	$\frac{136761984}{256^4}$	0.0755
10000000, 01000000, ...	1		$\frac{1}{256}$	$\frac{40611200}{256^4}$	0.0112
11000000, 10100000, ...	2		$\frac{1}{256}$	$\frac{19714688}{256^4}$	0.00432
11100000, 11010000, ...	3		$\frac{1}{256}$	$\frac{13086080}{256^4}$	0.00250
11110000, 11101000, ...	4		$\frac{1}{256}$	$\frac{11457152}{256^4}$	0.00210
11111000, 11110100, ...	5		$\frac{1}{256}$	$\frac{13086080}{256^4}$	0.00250
11111100, 11111010, ...	6		$\frac{1}{256}$	$\frac{19714688}{256^4}$	0.00432
11111110, 11111101, ...	7		$\frac{1}{256}$	$\frac{40611200}{256^4}$	0.0112
11111111	8		$\frac{1}{256}$	$\frac{136761984}{256^4}$	0.0755

TABLE I: **Probability of output functions.** In our notation, \bar{a} means NOT a , ab means a AND b , $a \oplus b$ means a XOR b (exclusive or), and $a + b$ means a OR b . **A** There are 4 output functions for $k = 1$ argument, which we express algebraically and in terms of their truth table. For network depth $n = 1$ to $n = 4$, we show the vector of probabilities $\mathbf{p}(n)$ of producing each of them. The probability depends only on the Hamming weight w of the output. **B** There are 16 output functions for $k = 2$ arguments. Again we show the probabilities $\mathbf{p}(n)$ of producing the outputs for different n . **C** There are 256 outputs for $k = 3$ arguments, which we only express by their truth tables. They are grouped by their Hamming weight w . We show the probabilities $\mathbf{p}(n)$ of producing each of them in their Hamming weight group for different n .

the k inputs, where 0 is false and 1 is true. We group together output functions with the same Hamming weight,

that is, the number of 1s in the truth table. On the right of Table I are the probabilities of the different outputs for various values of n . The probabilities start out uniform, but become biased as n increases.

As an example, consider $k = 2$ arguments and network depth $n = 2$. If we set f_1 to AND, $f_{2,1}$ to OR, and $f_{2,2}$ to NAND, then $F(a, b) = f_1(f_{2,1}(a, b), f_{2,2}(a, b)) = (a + b)(\bar{a} + \bar{b}) = a\bar{b} + \bar{a}b = a \oplus b$. Repeating this process over the 16^3 possible assignments to the three f s gives the distribution of $F(a, b)$ shown in Table IB.

Now let's turn to the general solution for arbitrary k and n . Whereas the different functions at a given level ($f_{i,1}, f_{i,2}, \dots$ in Fig. 1) will in general be different, by reason of symmetry they will have the same probability distribution. Therefore, the probabilities $P(f_{i,1}), P(f_{i,2}), \dots$ are identical, and we indicate them all by $P(f_i)$.

Our approach to working out the exact distribution of the output $F_n(a, b, \dots)$ is to consider the truth table of F_n , which we designate $\sigma_1, \dots, \sigma_\ell$. The binary-valued σ_i are independent and identically distributed, with

$$P(\sigma_i) = \begin{cases} w(F_{n-1})/\ell, & \text{for } \sigma_i = 1, \\ 1 - w(F_{n-1})/\ell, & \text{for } \sigma_i = 0, \end{cases}$$

where $w(F_{n-1})$ is the Hamming weight of F_{n-1} .

Now we're in a position to write the probability of the output being F_n in terms of the probabilities of F_{n-1} :

$$P(F_n | F_{n-1}) = \prod_{i=1}^{\ell} \left(\frac{w(F_{n-1})}{\ell} \right)^{\sigma_i} \left(1 - \frac{w(F_{n-1})}{\ell} \right)^{1-\sigma_i} P(F_{n-1}).$$

Note that the sum over i of $\sigma_i(F)$ is just $w(F)$. Then, pulling out $1/\ell^\ell$ and summing over f_{n-1} , we have

$$P(F_n) = \frac{1}{\ell^\ell} \sum_{F_{n-1}} w(F_{n-1})^{w(F_n)} (\ell - w(F_{n-1}))^{\ell - w(F_n)} P(F_{n-1}),$$

where recall $\ell = 2^k$, and we take $0^0 = 1$, a common convention in combinatorics.

Let $\mathbf{p}(n)$ be the vector of probabilities of the 2^ℓ different outputs F_n at level n , in lexicographical order. For example, for $k = 1$, $\mathbf{p}(1) = (1/4, 1/4, 1/4, 1/4)$ and $\mathbf{p}(2) = (6/4^2, 2/4^2, 2/4^2, 6/4^2)$, with more examples in Table I (though not in lexicographical order). The elements of $\mathbf{p}(n)$ satisfy

$$\mathbf{p}_j(n) = \frac{1}{\ell^\ell} \sum_{i=1}^{2^\ell} w_i^{w_j} (\ell - w_i)^{\ell - w_j} \mathbf{p}_i(n-1),$$

where $w_i = 0, 1, 1, 2, 1, 2, 2, 3, \dots$ is the Hamming weight of the binary representation of $i - 1$, and $\mathbf{p}_i(1) = 1/2^\ell$.

Notice how $\mathbf{p}(n)$ does not depend on the details of the output functions at level $n - 1$, but only on their Hamming weight. So, instead of working with the distribution \mathbf{p} of output functions, we can work with the simpler

distribution \mathbf{q} of the Hamming weight w of the output functions. The symmetry in \mathbf{p} reduces the problem from tracking 2^{2^k} to $2^k + 1$ components, since w can range from 0 to 2^k . For example, for $k = 1$, $\mathbf{q}(1) = (1/4, 2/4, 1/4)$ and $\mathbf{q}(2) = (6/4^2, 4/4^2, 6/4^2)$, with more examples in Table I. The distribution of the Hamming weight satisfies

$$\mathbf{q}_j(n) = \frac{1}{\ell^\ell} \sum_{i=0}^{\ell} \binom{\ell}{j} i^j (\ell - i)^{\ell-j} \mathbf{q}_i(n-1).$$

In other words, $\mathbf{q}(n) = \mathbf{A}\mathbf{q}(n-1)$, where \mathbf{A} is the $\ell + 1$ by $\ell + 1$ transition matrix

$$\mathbf{A}_{i,j} = \frac{1}{\ell^\ell} \binom{\ell}{j} i^j (\ell - i)^{\ell-j}. \quad (1)$$

For simplicity of notation, the rows and columns of \mathbf{A} are indexed 0 to ℓ , rather than 1 to $\ell + 1$, and recall that $\ell = 2^k$ and we take 0^0 to be 1. This matrix representation immediately allows us to write $\mathbf{q}(n)$ explicitly:

$$\mathbf{q}(n) = \mathbf{A}^n \mathbf{q}(1), \quad (2)$$

where $\mathbf{q}_i(1) = \binom{\ell}{i}/2^\ell$.

To get a sense of the structure of \mathbf{A} , it helps to see some examples. For $k = 1$,

$$\mathbf{A} = \frac{1}{2^2} \begin{pmatrix} \binom{2}{0} \\ \binom{2}{1} \\ \binom{2}{2} \end{pmatrix} \begin{pmatrix} 0^0 2^2 & 1^0 1^2 & 2^0 0^2 \\ 0^1 2^1 & 1^1 1^1 & 2^1 0^1 \\ 0^2 2^0 & 1^2 1^0 & 2^2 0^0 \end{pmatrix}.$$

For $k = 2$,

$$\mathbf{A} = \frac{1}{4^4} \begin{pmatrix} \binom{4}{0} \\ \binom{4}{1} \\ \binom{4}{2} \\ \binom{4}{3} \\ \binom{4}{4} \end{pmatrix} \begin{pmatrix} 0^0 4^4 & 1^0 3^4 & 2^0 2^4 & 3^0 1^4 & 4^0 0^4 \\ 0^1 4^3 & 1^1 3^3 & 2^1 2^3 & 3^1 1^3 & 4^1 0^3 \\ 0^2 4^2 & 1^2 3^2 & 2^2 2^2 & 3^2 1^2 & 4^2 0^2 \\ 0^3 4^1 & 1^3 3^1 & 2^3 2^1 & 3^3 1^1 & 4^3 0^1 \\ 0^4 4^0 & 1^4 3^0 & 2^4 2^0 & 3^4 1^0 & 4^4 0^0 \end{pmatrix}.$$

Throughout this paper we work with \mathbf{q} , the distribution of the Hamming weight of the output. But we're ultimately interested in \mathbf{p} , the distribution of the output. To translate between them,

$$\mathbf{p}_j = \mathbf{q}_i / \binom{\ell}{i}, \quad (3)$$

for all $j \in [0, \ell]$ such that the Hamming weight $w(F_j) = i$.

Properties of the exact distribution

With the general solution to the output distribution at hand, we now turn to understanding its properties. The matrix \mathbf{A} has $2^k + 1$ eigenvalues and eigenvectors. The first two we can work out for free. That's because the first and last columns of \mathbf{A} are all 0s apart from a 1 along the diagonal. So the first two eigenvalues are $\lambda_0 = \lambda_1 = 1$, corresponding to the eigenvectors $(1, 0, \dots, 0)$

and $(0, \dots, 0, 1)$. The j th eigenvalue is

$$\lambda_j = \frac{\binom{\ell}{j}}{\ell^j}, \quad (4)$$

where $\ell = 2^k$ and $\binom{\ell}{j} = \ell(\ell-1)\dots(\ell-j+1)$ is the falling factorial. Thus in the limit of large network depth n , the first two eigenvectors dominate, and the output F is a coin toss between true and false, with the probabilities of all other outputs vanishing.

However, the situation is more interesting than the large- n limit suggests. The first two eigenvectors govern only the absorbing states true and false and tell us nothing about the $2^{2^k} - 2$ non-constant outputs. For large depth n , the interior of $\mathbf{q}(n)$ (everything but the endpoints) is rather given by the third eigenvector \mathbf{v}_2 of \mathbf{A} . We are unable to write it down explicitly, but we can prove that it's flat to within a factor of $1 - 1/e$, apart from the endpoints.

To do so, let \mathbf{B} be the $2^k - 1$ by $2^k - 1$ interior of \mathbf{A} , that is, everything but the outer edge. The principal eigenvector of \mathbf{B} is the interior of \mathbf{v}_2 of \mathbf{A} . A corollary of the Perron-Frobenius theorem is that the principal eigenvector is at least as flat as the column sums of the matrix that it satisfies. For our matrix \mathbf{B} , the column sums are

$$\sum_{j=1}^{\ell-1} \mathbf{B}_{ij} = \frac{1}{\ell^\ell} \sum_{j=1}^{\ell-1} \binom{\ell}{j} i^j (\ell - i)^{\ell-j}.$$

If we extend the bounds in the sum to 0 and ℓ , by the binomial theorem the sum is just 1. So

$$\sum_{j=1}^{\ell-1} \mathbf{B}_{ij} = 1 - \left(\frac{i}{\ell}\right)^\ell - \left(\frac{\ell - i}{\ell}\right)^\ell.$$

This is minimized when $i = 1$ and $i = \ell - 1$, and maximized when $i = \ell/2$. For even modest values of k , $\ell = 2^k$ is large, and the minimum and maximum values of the sum tend to $(e - 1)/e$ and 1. So in the limit of large ℓ ,

$$\sum_{j=1}^{\ell-1} \mathbf{B}_{ij} \in \left[\frac{e-1}{e}, 1\right].$$

Thus the ratio of the smallest and largest internal components of \mathbf{v}_2 is at least $(1 - e)/e$ and at most 1.

As the network depth n increases, the distribution of the output changes in two ways. First, the distribution of the non-constant functions shifts from uniform to U-shaped with respect to Hamming weight. Outputs with low or high Hamming weights become exponentially more likely than those with intermediate ones.

Second, the probability of true and false shifts from negligible ($1/2^{2^k}$ each) to dominant ($1/2$ each), causing the bulk to vanish. The relative shape of the non-constant distribution continues to approach a U as n increases, but

in absolute terms it shrinks along the vertical axis, with all of the probabilities going to zero.

Critical network depth

Intriguingly, the two opposing tendencies of the output distribution—the dominance of the constant functions true and false and the convergence of the non-constant functions to a U-shape—result in a characteristic network depth n_c . Beyond n_c the convergence to a U-shape in absolute terms breaks down and the distribution becomes trivial: a coin toss between true and false.

We can work out the crossover network depth as follows. The first two eigenvalues govern the leading behavior of the constant functions, and the third governs that of the non-constant ones. We start with a uniform initial condition for $\mathbf{p}(1)$: $\mathbf{p}_i(1) = 1/2^{2^k}$. Using eq. (3), this translates to a binomial distribution for $\mathbf{q}(1)$: $\mathbf{q}_i(1) = \binom{2^k}{i}/2^{2^k}$. Projecting the initial condition $\mathbf{q}(1)$ onto the eigenvectors, the first two terms are $1/2$, and call the third c_3 . Keeping just the first three terms,

$$\mathbf{q}(n) \simeq 1/2 \mathbf{v}_0 1^n + 1/2 \mathbf{v}_1 1^n + c_3 ((1 - 1/2^k)^n \mathbf{v}_2).$$

From above, we know the interior of \mathbf{v}_2 is approximately flat. If we take it to be strictly flat, then

$$\begin{aligned} \mathbf{q}(n) \simeq & \left(\frac{1}{2}, 0, \dots, 0, \frac{1}{2}\right) \\ & + c_3 (1 - \frac{1}{2^k})^n \left(-1, \frac{2}{2^k-1}, \dots, \frac{2}{2^k-1}, -1\right). \end{aligned}$$

True and false start to dominate when the total proba-

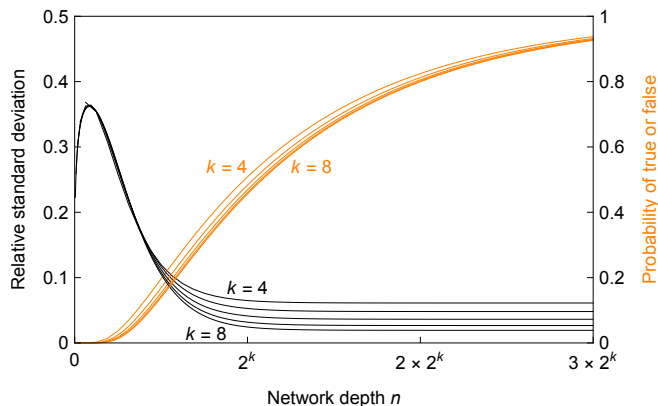


FIG. 2: **Critical network depth.** When the network depth reaches $n_c \simeq 2^k$, the Hamming weight distribution $\mathbf{q}(n)$ has flattened out, but true and false have yet to dominate. At this crossover, the output distribution $\mathbf{p}(n)$ is maximally U-shaped in absolute terms. The orange curves show the probability that the output is true or false as a function of n and for various k ; they approach 1 with n . The black curves show the standard deviation of the interior of \mathbf{q} normalized by its maximum value, as a function of n and for various k . Lower standard deviation means \mathbf{q} is flatter. The curves decrease with n to an asymptote that approaches 0 with k .

bility of the endpoints equals that of the interior:

$$2(1/2 - c_3(1 - 1/2^k)^{n_c}) = (2^k - 2)(1 - 1/2^k)^{n_c} \frac{2}{2^k - 1}.$$

This gives

$$n_c \simeq 2^k \ln(4 c_3),$$

where c_3 is a constant of order one.

Importantly, the convergence of the bulk towards a U-shape happens sooner than true and false take over. From the Hamming weight perspective, the distribution of \mathbf{q} flattens out before it diminishes. How do we know this? Because, in general, the equilibration time is proportional to the inverse of the spectral gap. In our case, the spectral gap $\lambda_0 - \lambda_2 = \lambda_1 - \lambda_2 = 1/2^k$, which governs the equilibration of true and false, is smaller than the gap $\lambda_2 - \lambda_3 = (1 - 1/2^k) 2/2^k \simeq 2/2^k$, which governs the equilibration of the bulk. So the bulk has a shorter equilibration time. This is confirmed in Fig. 2, where we show the probability of true or false (orange) and the relative standard deviation of the interior of the distribution (black), both as a function of n and for various k .

Comparison with experiments

To confirm our prediction of the output distribution, we conducted extensive experiments for various values of the number of arguments k and network depth n . In all cases, our computer experiments match our theory. Because the computational cost of enumerating all possible configurations of the Boolean functions f is formidable—it grows as $(2^{2^k})^{k(n-1)+1}$ —our experiments include complete enumeration of when possible, and sampling from the ensemble of configurations otherwise.

For $k = 2$ arguments, there are $16^3, 16^5, 16^7$ and 16^9 configurations for network depths $n = 2, 3, 4$ and 5 . We enumerated all of these configurations and, for each, determined the network's output function. We plot the probability $\mathbf{q}_i(n)$ of obtaining an output with Hamming weight i in Fig. 3A (points). This exactly matches our theoretical predictions given by eq. (2): the solid line shows the probabilities of the non-constant outputs and the dotted line the outputs true ($w = 4$) and false ($w = 0$). As n increases, the likelihood of true and false approach $1/2$ and the likelihoods of all other outputs fall.

For $k = 3$ arguments, there are $256^4, 256^7$ and 256^{10} configurations for network depths $n = 2, 3$ and 4 . For $n = 2$, we enumerated all of the configurations. For $n = 3$ and 4 , this is computationally infeasible, so we sampled them instead. We randomly assigned one of the 256 functions to each of the nodes and then determined the network output, repeating this two million times. These are plotted in Fig. 3B (points). We calculated error bars, but these are negligible compared to the point size. Our theory predicts the computer experiments exactly for $n = 2$ and to within statistical significance for $n = 3$ and 4 .

For $k = 4$ arguments, there are 65536^5 and 65536^9 con-

figurations for network depths $n = 2$ and 3 . These are too many to enumerate, so we took two million samples for $n = 2$ and the same for $n = 3$. These are plotted in Fig. 3C. Once again, our theory predicts the experiments to

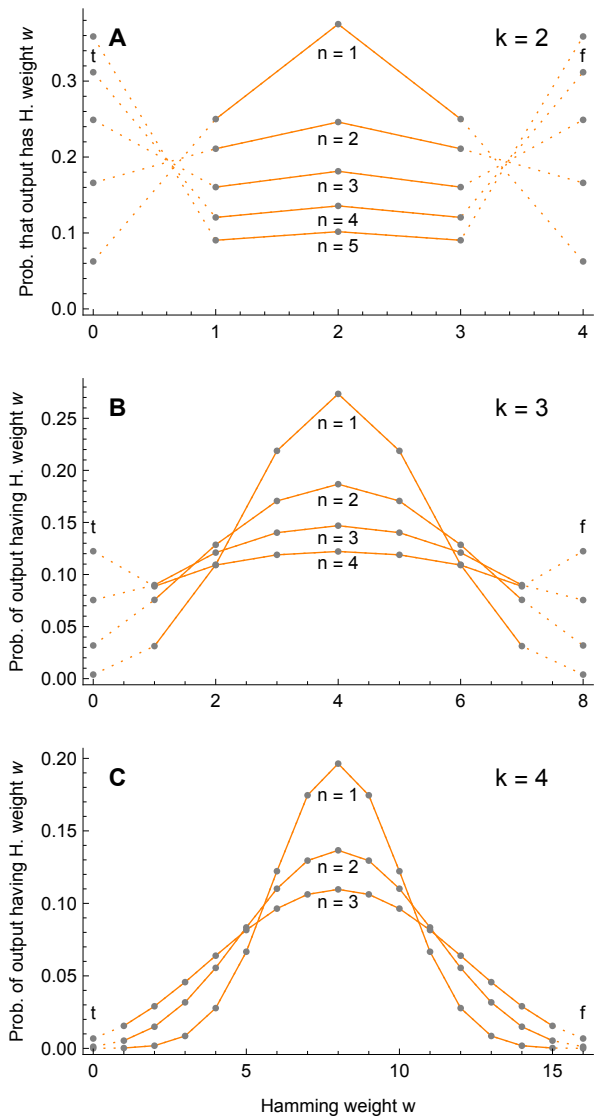


FIG. 3: **Experiments confirm our predictions.** We compare our prediction of the probability $q(n)$ of the output function (lines) with computer experiments (points), for various values of the number of arguments k and network depth n . The vertical axis is the probability of an output with a given Hamming weight w ; outputs with the same w have the same probability. In all cases, our experiments agree with our theory exactly or, when sampling, to within statistical significance. **A** For $k = 2$, we enumerated all of the configurations up to network depth $n = 5$. As n grows, the distribution of the output flattens out and falls. But for true and false ($w = 4$ and $w = 0$), the probabilities approach $1/2$. **B** For $k = 3$, we show exact results for $n = 1$ and 2 , and sample the configurations for $n = 3$ and 4 . **C** For $k = 4$, we show exact results for $n = 1$, and sample for $n = 2$ and 3 .

within statistical significance.

Discussion

The evolution of the output distribution with network depth resembles relaxation in a finite Markov process possessing two absorbing states. The constant functions true and false play the role of ordered phases, while the non-constant functions constitute a large transient manifold whose structure is governed by sub-leading eigenmodes of the transition operator.

A central result is that random deep composition exhibits a finite-depth crossover from increasing functional structure to functional collapse. Although successive layers initially bias the distribution toward extreme Hamming weights, beyond a characteristic depth $n_c \simeq 2^k$, the probability concentrates on the two constant functions. Deeper composition is usually associated with more expressive power. Our results show that when the local Boolean rules are chosen randomly, depth has the opposite effect beyond a characteristic scale. Thus network depth is not an unlimited source of expressivity but a resource with a finite optimal range.

Our insights are consistent with the work of Mozeika, Li and Saad [6, 8], but go farther. They observed that “random deep ReLU networks compute only *constant* Boolean functions in the infinite depth limit”. Our exact finite-depth solution agrees with their asymptotic analysis but also reveals the rich intermediate regime that precedes collapse, which is the relevant regime for learning applications.

As an exactly-solvable null model, our model is striking. Even though the local rules are chosen uniformly, the resulting global functions are highly non-uniformly distributed. What this tells us is that complex structure can emerge purely from the architecture and compositional process, without learning or adaptation. This is a crucial insight when we regard a deep-layered machine as an archetypal input-output map. In an input-output map, many inputs (instructions) map to a smaller number of outputs (functionalities). Input-output maps are prevalent in science (e.g., RNA folding) and technology (e.g., Boolean circuits), and mounting empirical evidence suggests that they tend to be biased towards simple outputs [10–12]. Because deep-layered machines are a rare example of an input-output map whose frequencies can be computed analytically, their bias towards simplicity can be rigorously tested. We investigate the pronounced simplicity bias of deep-layered machines in [13].

Our model suggests a number of open questions for follow-up work. One is an investigation of universality. Fig. 2 offers compelling evidence that the growth of the constant functions with n , measured in units of 2^k , rapidly approaches a limiting form. More generally, we conjecture that, for moderate k and above, much of the behavior of a deep-layered machine can be universally parameterized by $n/2^k$.

Another extension is to strengthen the analogy with

deep neural networks by considering a restricted ensemble of Boolean functions, rather than the uniform distribution considered here. A natural choice is threshold Boolean functions, which imitate the integrate and fire aspect of artificial neurons; threshold functions are essentially perceptrons. Solving this modified version of the problem would provide first-principles insights into the kind and degree of biases inherent in deep learning.

-
- [1] B. Drossel, Random Boolean networks, in *Reviews of Non-linear Dynamics and Complexity* (Wiley-VCH, Weinheim, 2008).
- [2] B. Samuelsson, C. Troein, Superpolynomial growth in the number of attractors in Kauffman networks, *Phys Rev Lett* **90**, 098701 (2003).
- [3] B. Drossel, T. Mihaljev, F. Greil, Number and length of attractors in a critical Kauffman model with connectivity one, *Phys Rev Lett* **94**, 088701 (2005).
- [4] T. Fink, Exact behavior of the critical Kauffman model with connectivity one, *Phys Rev Res*, **6**, 43315 (2024).
- [5] T. Fink, F. Sheldon. Number of cycles in the critical Kauffman model is exponential, *Phys Rev Lett*, **131**, 267402 (2023).
- [6] B. Li, D. Saad, Exploring the function space of deep-learning machines, *Phys Rev Lett* **120**, 248301 (2018).
- [7] K. Raman, A. Wagner, The evolvability of programmable hardware, *J Roy Soc Interface* **8**, 269 (2011).
- [8] A. Mozeika, B. Li, D. Saad, Space of functions computed by deep layered machines, *Phys Rev Lett* **125**, 168301 (2020).
- [9] L. Zdeborová et al., Understanding deep learning is also a job for physicists *Nat Phys* **16**, 602 (2020).
- [10] K. Dingle, C. Q. Camargo, A. A. Louis, Input-output maps are strongly biased towards simple outputs, *Nat Commun* **9**, 761 (2018).
- [11] I. G. Johnston et al., Symmetry and simplicity spontaneously emerge from the algorithmic nature of evolution, *P Natl Acad Sci USA* **119**, e2113883119 (2022).
- [12] C. Mingard et al., Deep neural networks have an inbuilt Occam's razor, *Nat Commun* **16**, 220 (2025).
- [13] T. M. A. Fink, Deep-layered machines have a built-in Occam's razor, arXiv:2603.01217.