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
(Dated: June 1, 2026)

In a deep-layered machine, each node in the network is a Boolean function of all the nodes below it. We derive the exact distribution of the global output given a random assignment of Boolean functions to the local nodes, and confirm our prediction through extensive experiments. As the network depth increases, the distribution changes in two ways: it becomes exponentially biased towards preferred outputs, and the outputs true and false dominate, eventually drowning out everything else. These opposing forces give rise to a critical network depth, at which the distribution is maximally biased.

In a deep-layered machine, there are n layers, each of which has k nodes—apart from the top layer, which has one. At every node there is a Boolean function, or logic for short, which depends on the k arguments in the layer below it. Fig. 1 shows examples of deep-layered machines for $k = 1$, 2 and 3. In general there are 2^{2^k} logics of k arguments. Since the number of nodes in the network is $k(n-1)+1$, the number of network configurations is $\left(2^{2^k}\right)^{k(n-1)+1}$. All of these configurations generate one of the 2^{2^k} possible output functions, so the map from configurations to outputs is highly degenerate.

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 shown algebraically as well as via their truth tables. In our notation, $\overline{a}$ means NOT a, ab means a AND b, $a \oplus b$ means a XOR b (exclusive or), and $a + b$ means a OR b. The order of operations is AND precedes XOR, which precedes OR. A truth table is the binary string of length 2^k that determines the output of the function for all possible combinations of 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 more and more biased as n increases.

At the heart of deep-layered machine is the composition of Boolean functions, or logics. The composition of logics works just like the composition of ordinary functions. Consider, for example, $k = 2$ 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

$$\begin{aligned} F(a,b) &= f_1(f_{2,1}(a,b), f_{2,2}((a,b)) \\ &= (a+b)(\overline{a}+\overline{b}) = a\overline{b} + \overline{a}b = a \oplus b. \end{aligned}$$

Throughout this paper we use f to indicate the local logic functions assigned to the network nodes, and F to indicate the global logic function, or output, of the entire network. Whereas f is a function of the k inputs immediately below it, F is a function of the k terminal arguments $a, b, \ldots$. The goal of this paper is to understand the distribution of $F(a, b, \ldots)$.

In this paper we do four things, which correspond to the next four sections. First, we derive the the exact distribution of the output given a random assignment of logics to the network nodes. Second, we confirm our theory by doing extensive computer experiments for networks of different sizes, checking over four billion configurations. Third, we show that the distribution changes in two ways as n increases: it becomes U-shaped with respect to the Hamming weight of the output; and true and false take over and eventually dominate. Fourth, we show that these two opposing forces give rise to a critical network depth, proportional to 2^k, at which point the distribution exhibits maximum bias. We conclude with a discussion and some extensions and open problems.

Exact distribution

We start by working out the exact distribution of the output F given a random assignment of logics to the network nodes—the light gray nodes in Fig. 1. Whereas the different functions at a given level ($f_{i,1}, f_{i,2}, \ldots$ in Fig. 1) will in general different, by reason of symmetry they will have the same probability distribution. Therefore, the probabilities $P(f_{i,1}), P(f_{i,2}), \ldots$ are identical, and we indicate them all by $P(f_i)$.

Our approach to working out the exact distribution of

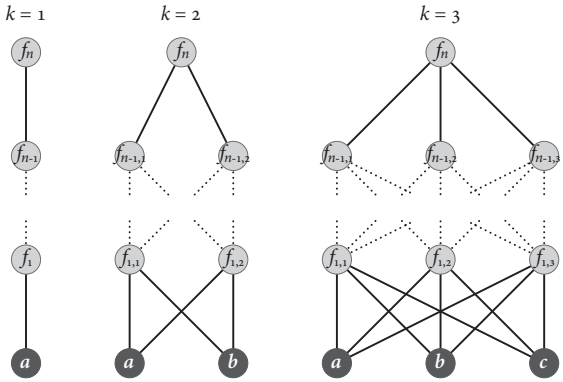

FIG. 1: **Deep-layered machines.** In a network of k arguments (a, b, $\ldots$), each logic depends on all k of the arguments below it, each of which depends on the k arguments below it, and so on, down to n levels. The goal of this paper is to determine the probability distribution of the output function $F(a, b, \ldots)$ given a random assignment of logics to the fs at the light gray nodes.

			Prob. $\mathbf{p}(n)$ *of* F *for* $k=1$			
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	$\frac{1}{4}$	$\frac{2}{4^2}$	$\frac{4}{4^3}$	$\frac{8}{4^4}$
$\overline{a}$	01	1				
true	11	2	$\frac{1}{4}$	$\frac{6}{4^2}$	$\frac{28}{4^3}$	$\frac{120}{4^4}$

			Prob. $\mathbf{p}(n)$ *of* F *for* $k=2$			
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	$\frac{1}{16}$	$\frac{216}{16^3}$	$\frac{42048}{16^5}$	$\frac{8087040}{16^7}$
$a\overline{b}$	0100	1				
$\overline{a}b$	0010	1				
$\overline{a}\overline{b}$	0001	1				
a	1100	2	$\frac{1}{16}$	$\frac{168}{16^3}$	$\frac{31680}{16^5}$	$\frac{6068736}{16^7}$
$\overline{a}$	0011	2				
b	1010	2				
$\overline{b}$	0101	2				
$a\oplus b$	0110	2				
$a\oplus\overline{b}$	1001	2				
$a+b$	1110	3	$\frac{1}{16}$	$\frac{216}{16^3}$	$\frac{42048}{16^5}$	$\frac{8087040}{16^7}$
$a+\overline{b}$	1101	3				
$\overline{a}+b$	1011	3				
$\overline{a}+\overline{b}$	0111	3				
true	1111	4	$\frac{1}{16}$	$\frac{680}{16^3}$	$\frac{261056}{16^5}$	$\frac{83663360}{16^7}$

		Prob. $\mathbf{p}(n)$ *of* F *for* $k=2$		
Output $F(a,b,c)$	w	$n=1$	$n=2$	$n=3$
00000000	0	$\frac{1}{256}$	$\frac{136761984}{256^4}$	0.0755
00000001, 00000010, ...	1	$\frac{1}{256}$	$\frac{40611200}{256^4}$	0.0112
00000011, 00000101, ...	2	$\frac{1}{256}$	$\frac{19714688}{256^4}$	0.00432
00000111, 00001011, ...	3	$\frac{1}{256}$	$\frac{13086080}{256^4}$	0.00250
00001111, 00010111, ...	4	$\frac{1}{256}$	$\frac{11457152}{256^4}$	0.00210
00011111, 00101111, ...	5	$\frac{1}{256}$	$\frac{13086080}{256^4}$	0.00250
00111111, 01011111, ...	6	$\frac{1}{256}$	$\frac{19714688}{256^4}$	0.00432
01111111, 10111111, ...	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.** There are 2^{2^k} logic functions of k arguments, which can be expressed algebraically or in terms of their binary truth table. **($k=1$).** There are 4 functions of one argument. For network depth $n=1$ to $n=4$, we show the vector of probabilities $\mathbf{p}(n)$ of producing each of the output functions. The probability depends only on the Hamming weight w of the output, that is, the number of 1s in its truth table. **($k=2$).** There are 16 functions of two arguments. Again we show the probabilities $\mathbf{p}(n)$ of producing each of the outputs for different network depths. **($k=3$).** There are 256 functions of three arguments, which we only express by their truth tables. They are grouped by their Hamming weight w. For network depth $n=1$ to $n=3$, we show the probabilities $\mathbf{p}(n)$ of producing each of the outputs in the Hamming weight group.

the output $F_n(a,b,\ldots)$ is to consider the truth table of F_n, which we designate $\sigma_1,\ldots,\sigma_\ell$. 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 $w_i=0,1,1,2,1,2,2,3,\ldots$ 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, which has 2^{2^k} components, we can work with the simpler distribution $\mathbf{q}$ of the Hamming weight of the output functions, which has 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

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

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 of immediately allows us to write $\mathbf{q}(n)$ explicitly:

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

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,

$$\mathbf{p}_j = \mathbf{q}_i / \tbinom{\ell}{i}, \tag{3}$$

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 $(1, 0, \ldots, 0)$ and $(0, \ldots, 0, 1)$. The jth eigenvalue is

$$\lambda_j = \frac{(\ell)_j}{\ell^j}, \tag{4}$$

where $\ell = 2^k$ and $(\ell)_j = \ell(\ell-1)\ldots(\ell-j+1)$ is the falling factorial. Thus in the limit of large network depth n, 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 tell us nothing about the bulk of the outputs, that is, all of the $2^{2^k} - 2$ functions that are not true and false. For large depth n, $\mathbf{q}(n)$ for the bulk is rather given by the third eigenvector $\mathbf{v_2}$ of $\mathbf{A}$. We are unable to write it down explicitly, but we can show that it's approximately flat, apart from the endpoints. In particular, the ratio of the smallest and largest internal components of $\mathbf{v_2}$ is at least $(1-e)/e$ and at most 1.

To show that the interior of $\mathbf{v_2}$ is flat, 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 $\mathbf{A}$. We know, in general, 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

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

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. Thus in the limit of large ℓ,

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

The interior of $\mathbf{v_2}$ of $\mathbf{A}$ is at least as flat as this.

As the network depth n increases, the distribution of the output function changes in two ways. First, the distribution of the bulk (everything but true and false) 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 middling 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 bulk 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.

Comparison with experiments

To confirm our prediction of the output distribution of a deep layered machine, 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 inputs is formidable—it grows as $\left(2^{2^k}\right)^{k(n-1)+1}$—our experiments include complete enumeration of the inputs when possible, and sampling from the ensemble of inputs otherwise.

For $k = 2$ arguments (Fig. 1A), there are $16^3, 16^5, 16^7$ and 16^9 input configurations for network depths $n = 2, 3, 4$ and 5. We enumerated all of these inputs and, for each, determined the network's output function. Since the probability of an output is the same for outputs with the same Hamming weight, we plot the probability $\mathbf{q}_i(n)$ of obtaining a given Hamming weight i in Fig. 2A

(points). This exactly matches our theoretical predictions given by eq. (2). The solid line indicates the probabilities

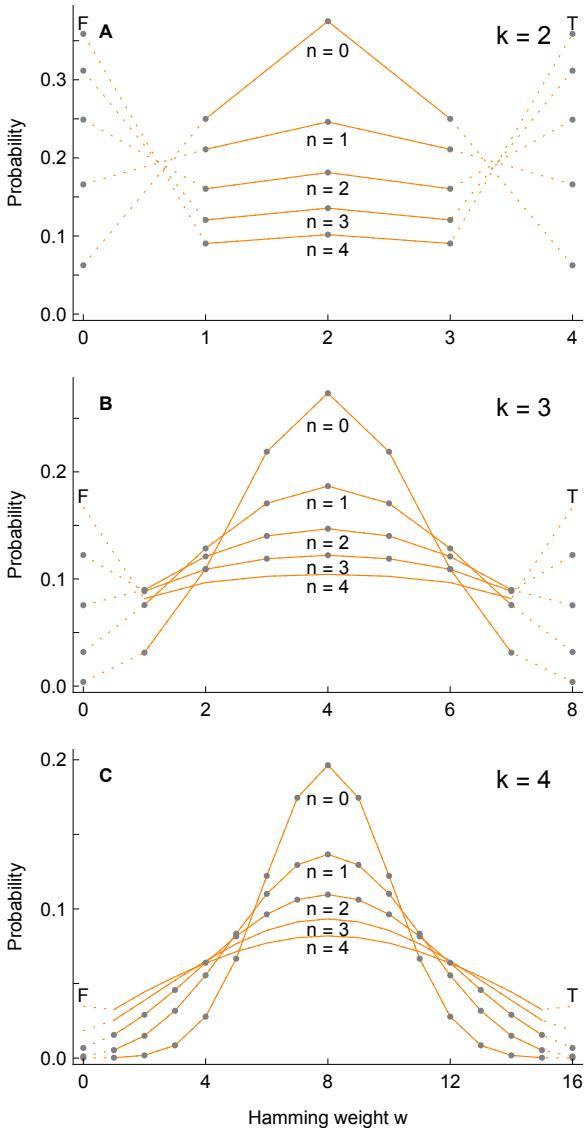

FIG. 2: **Experiments confirm our predictions.** We compare our prediction of the probability $\mathbf{q}(n)$ in eq. (2) of the output function (lines) with computer experiments (points), for various values of the number of arguments k and the network depth n. The vertical axis is the probability of 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 input configurations up to network depth $n = 4$. As n grows, the distribution of the output function 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 $n = 0$ and 1, and sample the inputs for $n = 2$ and 3. We show our $n = 4$ theory for comparison. **C** For $k = 4$, we show exact results for $n = 0$, and sample the inputs for $n = 1$ and 2. We also show our $n = 3$ and 4 theory.

of the bulk of the outputs and the dotted line connects to 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 the remaining outputs fall.

For $k = 3$ arguments (Fig. 1B), there are 256^4, 256^7 and 256^{10} input configurations for network depths $n = 2, 3$ and 4. For $n = 2$, we enumerated all of the inputs. For $n = 3$ and 4, this is computationally infeasible, so we sampled the inputs instead. We randomly assigned one of the 256 logics to each of the nodes and then determined the network output, repeating this two million times. These are plotted in Fig. 2B (points). We calculated errors bars, but these are negligible compared to the point size in the plots. Our theory predicts the computer experiments exactly for $n = 2$ and to within statistical significance for $n = 3$ and 4. We also show our $n = 5$ predictions for comparison.

For $k = 4$ arguments (Fig. 1C), there are 65536^5 and 65536^9 inputs 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. 2C. Once again, our theory predicts the experiments to within statistical significance. We also show our $n = 4$ and 5 predictions for comparison.

Critical network depth

Intriguingly, the two opposing tendencies of the output distribution—the dominance of true and false and the convergence of everything else to a U-shape—give rise to a critical network depth n_{crit}. Beyond n_{crit} 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 critical network depth as follows. The first two eigenvalues govern the leading behavior of true and false, and the third governs that of everything else (the bulk). 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

$$\mathbf{q}(n) \simeq (\tfrac{1}{2}, 0, \ldots, 0, \tfrac{1}{2}) + c_3(1 - \tfrac{1}{2^k})^n(-1, \tfrac{2}{2^k-1}, \ldots, \tfrac{2}{2^k-1}, -1).$$

True and false start to dominate when the total probability of the endpoints equals that of the interior:

$$2\big(1/2 - c_3(1 - 1/2^k)^{n_{\text{crit}}}\big) = (2^k - 2)(1 - 1/2^k)^{n_{\text{crit}}}\tfrac{2}{2^k-1}.$$

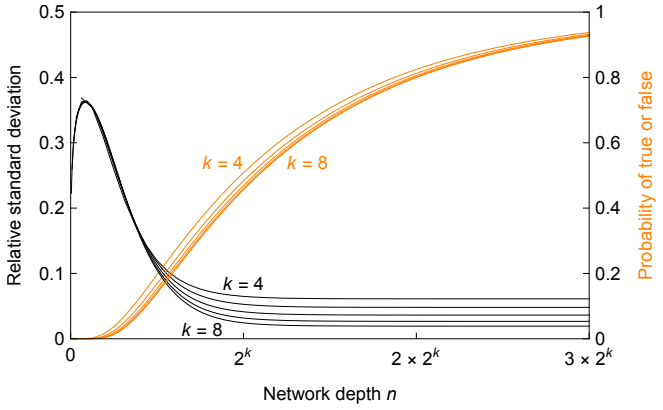

FIG. 3: **Critical network depth.** When the network depth reaches $n_{\text{crit}} \simeq 2^k$, the Hamming weight distribution $\mathbf{q}(n)$ has flattened out, but true and false have yet to dominate. At this critical network depth, the output distribution $\mathbf{p}(n)$ is maximally U-shaped, in absolute terms. In this plot, the black curves show the standard deviation of the interior of the $\mathbf{q}$ distribution as a function of n and for various values of k. Lower standard deviation means the distribution is flatter. The curves decrease with n to an asymptote that approaches 0 with k. 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.

This gives

$$n_{\text{crit}} \simeq 2^k \ln(4\, c_3),$$

where c_3 approaches 1 from below.

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. 3, 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.

Discussion

The distribution of the output function $F(a, b, \ldots)$ of a deep-layered machine has several notable and surprising properties. First, the probability of F depends not on the detailed form of F but only on its Hamming weight. Outputs with the same Hamming weight have the same probability of occurring. This means we can study the more compact distribution $\mathbf{q}$ of the Hamming weight of the output than the distribution $\mathbf{p}$ of the output itself. The exchange rate between the two is just the binomial coefficients. Whereas $\mathbf{q}(n)$ starts out with a binomial distribution and ends up essentially flat (apart from the endpoints), $\mathbf{p}(n)$ starts out flat and ends up as an inverted binomial distribution (apart from the endpoints).

While all this is happening, another effect is taking place as n increases: the probability of the bulk is leaking into the output functions true and false (the endpoints). In the infinite n limit, all of the bulk probability has leaked away, and true and false each occur with probability 1/2. Mozeika *et al.* [12] observed a similar phenomena when they considered deep-layered machines with rectified linear unit functions: "random deep ReLU networks compute only *constant* Boolean functions in the infinite depth limit". But their macroscopic approach misses the key interim behavior for finite n.

The analytic form of the transition matrix $\mathbf{A}$ gives no a priori reason to distinguish between, on the one hand, $w = \ell$ and $w = 0$ (true and false) and, on the other, $0 < w < \ell$ (everything else). But in fact $\mathbf{A}$ describes two distinct dynamical processes—one governed by the outer edge of $\mathbf{A}$, and the other by the interior of $\mathbf{A}$. This is because the left and right columns of $\mathbf{A}$ are all zeroes, apart from corners, the boundary and interior of $\mathbf{A}$ don't talk to each other.

A deep-layered machine can be viewed as an archetypal input-output map. The inputs can be thought of as instructions, and the outputs can be thought of as functionalities. Input-output maps are prevalent in biology, physics, mathematics and technology, and mounting empirical evidence suggests that they are biased towards simple outputs [1, 2]. We study the simplicity bias of deep-layered machines in a follow-up paper.

Taking the inputs and outputs to be genotypes and phenotypes suggests a number of questions about the evolutionary properties of deep-layered machines. In particular, what is their robustness and evolvability? The robustness is the fraction of mutations that change the phenotype, that is, the output function F. The evolvability is the number of distinct phenotypes in the adjacent possible of the phenotype. Both quantities depend on the geometry of the neutral network of the phenotype F: the set of all genotypes (configurations) that produce the same phenotype (output F).

Deep-layered machines provide a model framework for studying these properties, and their symmetry suggests that they might be analytically tractable.

[1] K. Dingle, C. Q. Camargo, A. A. Louis, Input-output maps are strongly biased towards simple outputs, Nat Commun **9**, 761 (2018).

[2] I. G. Johnston et al., Symmetry and simplicity spontaneously emerge from the algorithmic nature of evolution, P Natl Acad Sci USA **119**, e2113883119 (2022).

[3] C. Mingard et al., Deep neural networks have an inbuilt Occam's razor, Nat Commun **16**, 220 (2025).

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

[5] J. L. England, E. I. Shakhnovich, Structural determinant of protein designability, Phys Rev Lett **90**, 218101 (2003).

[6] S. E. Ahnert, T. M. A. Fink, Form and function in gene regulatory networks, J Roy Soc Interface **13**, 20160179 (2016).
[7] T. Fink, F. Sheldon. Number of cycles in the critical Kauffman model is exponential, Phys Rev Lett, **131**, 267402 (2023).
[8] T. Fink, Exact behavior of the critical Kauffman model with connectivity one, Phys Rev Res, **6**, 43315 (2024).
[9] T. Fink and R. Hannam, Biological logics are restricted, arxiv.org/abs/2109.12551.
[10] N. J. A. Sloane, editor, The On-Line Encyclopedia of Integer Sequences, published electronically at https://oeis.org, 2021.
[11] K. Raman, A. Wagner, The evolvability of programmable hardware, J Roy Soc Interface **8**, 269 (2011).
[12] A. Mozeika, B. Li, D. Saad, The space of functions computed by deep layered machines, Phys Rev Lett **125**, 168301 (2020).