LETTER

https://doi.org/10.1038/s41586-018-0289-6

Scaling up molecular pattern recognition with DNA-based winner-take-all neural networks

Kevin M. Cherry[1] & Lulu Qian[1,2]*

From bacteria following simple chemical gradients[1] to the brain distinguishing complex odour information[2], the ability to recognize molecular patterns is essential for biological organisms. This type of information-processing function has been implemented using DNA-based neural networks[3], but has been limited to the recognition of a set of no more than four patterns, each composed of four distinct DNA molecules. Winner-take-all computation[4] has been suggested[5,6] as a potential strategy for enhancing the capability of DNA-based neural networks. Compared to the linear-threshold circuits[7] and Hopfield networks[8] used previously[3], winner-take-all circuits are computationally more powerful[4], allow simpler molecular implementation and are not constrained by the number of patterns and their complexity, so both a large number of simple patterns and a small number of complex patterns can be recognized. Here we report a systematic implementation of winner-take-all neural networks based on DNA-strand-displacement[9,10] reactions. We use a previously developed seesaw DNA gate motif[3,11,12], extended to include a simple and robust component that facilitates the cooperative hybridization[13] that is involved in the process of selecting a 'winner'. We show that with this extended seesaw motif DNA-based neural networks can classify patterns into up to nine categories. Each of these patterns consists of 20 distinct DNA molecules chosen from the set of 100 that represents the 100 bits in 10 × 10 patterns, with the 20 DNA molecules selected tracing one of the handwritten digits '1' to '9'. The network successfully classified test patterns with up to 30 of the 100 bits flipped relative to the digit patterns 'remembered' during training, suggesting that molecular circuits can robustly accomplish the sophisticated task of classifying highly complex and noisy information on the basis of similarity to a memory.

Winner-take-all computation[4] is one of the simplest competitive neural-network models, inspired by the lateral inhibition and competition observed among biological neurons in the brain[14]. In this model, the output of a neuron is ON if and only if the weighted sum of all binary inputs is the largest among all neurons (Fig. 1a). Here, in a winner-take-all neural network, the weight matrix associated with each output is referred to as a 'memory'. As shown in Fig. 1b, a simple training algorithm involves using the target patterns as weights. The example network has two memories—in other words, it 'remembers' two patterns—'L' and 'T'. The network 'recognizes' a pattern by comparing it to all memories and identifying which memory the pattern is most similar to—the output associated with this memory will be ON and all other outputs will be OFF. For instance, a corrupted 'L' with the last bit flipped from 1 to 0 can be recognized as 'L', because it will result in y_1 (the output of the neuron remembering 'L') being ON and y_2 (the output of the neuron remembering 'T') being OFF.

The winner-take-all function can be broken into five subfunctions, each of which can be implemented with a simple chemical reaction (Fig. 1c): First, weight multiplication of $x_i \times w_{ij}$ (where x_i is a binary input and w_{ij} is an analogue weight) is implemented with reactions wherein an input species X_i catalytically converts a weight species W_{ij} to an intermediate product P_{ij}. If X_i is absent, then no P_{ij} will be produced; if X_i is present, then the final concentration of P_{ij} will be determined by the initial concentration of W_{ij}, thus setting the value of the weighted input. Second, summation is implemented with reactions that convert all intermediate species P_{ij} within the same neuron to a common weighted-sum species S_j. Third, comparison of weighted sums to determine which is the largest is implemented with a set of 'pairwise annihilation' reactions, wherein each weighted-sum species S_j destroys any other weighted-sum species S_k until only a single winner remains. Fourth, signal-restoration reactions bring the concentration of the winner species back to a predetermined output value—the final concentration of a winning output species Y_j corresponds to the initial concentration of a restoration-gate species RG_j. Last, reporting reactions are used to convert each output Y_j to a fluorescent signal $Fluor_j$.

All reactions except pairwise annihilation and signal restoration naturally take place sequentially, because the product of a previous reaction is a reactant of the next one. Because there are common reactants in the annihilation and restoration reactions, we used different rates to control their order: the former has a much faster rate constant than the latter, so a winner that survives all fast competitions is then converted slowly to an output signal.

Weight multiplication and signal restoration are both catalytic reactions, implemented with a pair of seesawing reactions[11] (Fig. 1e, Extended Data Fig. 1). An input X_i (or weighted sum S_j) species first interacts with a weight W_{ij} (or restoration gate RG_j) species through a reversible strand-displacement reaction[15] to release an intermediate product P_{ij} (or output Y_j) species. A fuel strand XF_i (or YF_j) then frees the input (or weighted sum) species for more catalytic cycles. As long as the fuel strand is in excess, all weight (or restoration gate) molecules will eventually be converted to intermediate (or output) molecules. Summation is implemented with a single seesawing reaction facilitated by a summation gate SG_j (Extended Data Fig. 1). The reaction is reversible by itself but drained forward by the downstream irreversible reaction of pairwise annihilation.

The annihilation reaction is implemented with cooperative hybridization[13] (Fig. 1f). One weighted-sum strand S_j can bind to a toehold on one side of an annihilator molecule Anh_{jk} and branch-migrate to the middle point of the double-stranded domain. If only S_j is present, then this process is completely reversible and no molecules will be consumed. However, if another weighted-sum strand S_k is also present, then it can bind to another toehold on the opposite side of the annihilator and also branch-migrate to the middle point of the double-stranded domain. When the S_j and S_k strands reach the middle point simultaneously, the annihilator will be split apart into two waste molecules. Because neither waste molecule has a toehold exposed, it cannot interact with any other molecules. The annihilation reaction shown in Fig. 1f is designed to be roughly 100 times faster than the signal-restoration reaction shown in Fig. 1e, owing to the two extra nucleotides in both toeholds on the annihilator—it is known that the rate of strand displacement reactions grows exponentially faster with a longer toehold[15,16].

Reporting is implemented with an irreversible strand-displacement reaction, wherein an output strand Y_j interacts with a double-stranded

[1]Bioengineering, California Institute of Technology, Pasadena, CA, USA. [2]Computer Science, California Institute of Technology, Pasadena, CA, USA. *e-mail: luluqian@caltech.edu

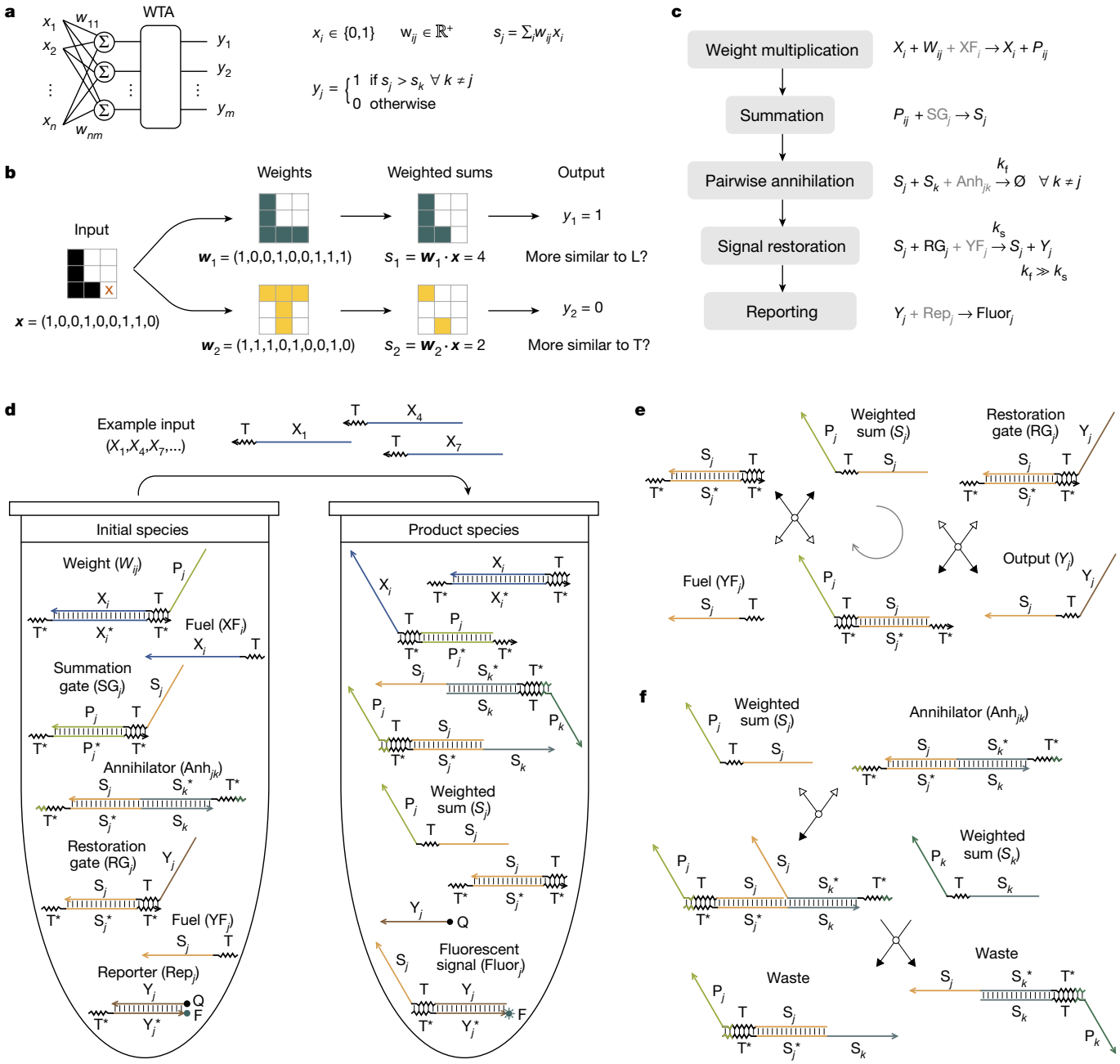


Fig. 1 | Winner-take-all neural network and its DNA implementation. **a**, A winner-take-all (WTA) neural network with m memories that each has n bits; x_1 to x_n and y_1 to y_m are binary inputs and outputs, respectively; w_{ij} ($1 \leq i \leq n$ and $1 \leq j \leq m$) are analogue weights of positive, real numbers; s_j and s_k ($1 \leq j \neq k \leq m$) are weighted sums of the inputs. **b**, Example pattern recognition using target patterns as weights. Each 9-bit pattern is shown in a 3 × 3 grid. Each black or coloured pixel indicates a 1 and each white pixel indicates a 0. The two target patterns correspond to the letters 'L' and 'T', respectively. If the input pattern is corrupted (for example, the last bit of 'L' is flipped from 1 to 0, as indicated by the orange cross), then the neural network can still recognize it as being more similar to 'L' than to 'T', because the weighted sum using 'L' as weights is still larger than the weighted sum using 'T' as weights. **c**, Chemical-reaction-network implementation. The concentrations of chemical species X_i, W_{ij}, S_j and Y_j correspond to the values of variables x_i, w_{ij}, s_j and y_j, respectively. The species in black are needed as part of the function, whereas the species in grey are needed to facilitate the reactions. The waste molecules are not shown in the reactions. k_f and k_s are the rate constants of the pairwise-annihilation and signal-restoration reactions, respectively. **d**, DNA-strand-displacement implementation. The initial test tube (left) shows all DNA species with $1 \leq i \leq n$ and $1 \leq j \neq k \leq m$. The final test tube (right) shows only the product species after a set of input strands are added, with i, j and k being a subset of all possible numbers depending on the specific input. Zigzag lines indicate short (5 or 7 nucleotide) toehold domains and straight lines indicate long (15 or 20 nucleotide) branch-migration domains in DNA strands, with arrowheads marking their 3′ ends. Each domain is labelled with a name and assigned a unique DNA sequence, with asterisks in the names indicating sequence complementarity. Strand modifications are labelled as F and Q, where F indicates a fluorophore and Q indicates a quencher. **e**, Signal-restoration reaction. The grey circle with an arrow indicates the direction of the catalytic cycle. **f**, Pairwise-annihilation reaction. Representative (not all possible) states are shown. In **e** and **f**, arrows with black-filled and white-filled arrowheads indicate the forwards and backwards directions of a reaction step, respectively. The mechanisms of weight multiplication, summation and reporting reactions are shown in Extended Data Fig. 1. DNA sequences are listed in Supplementary Table 1.

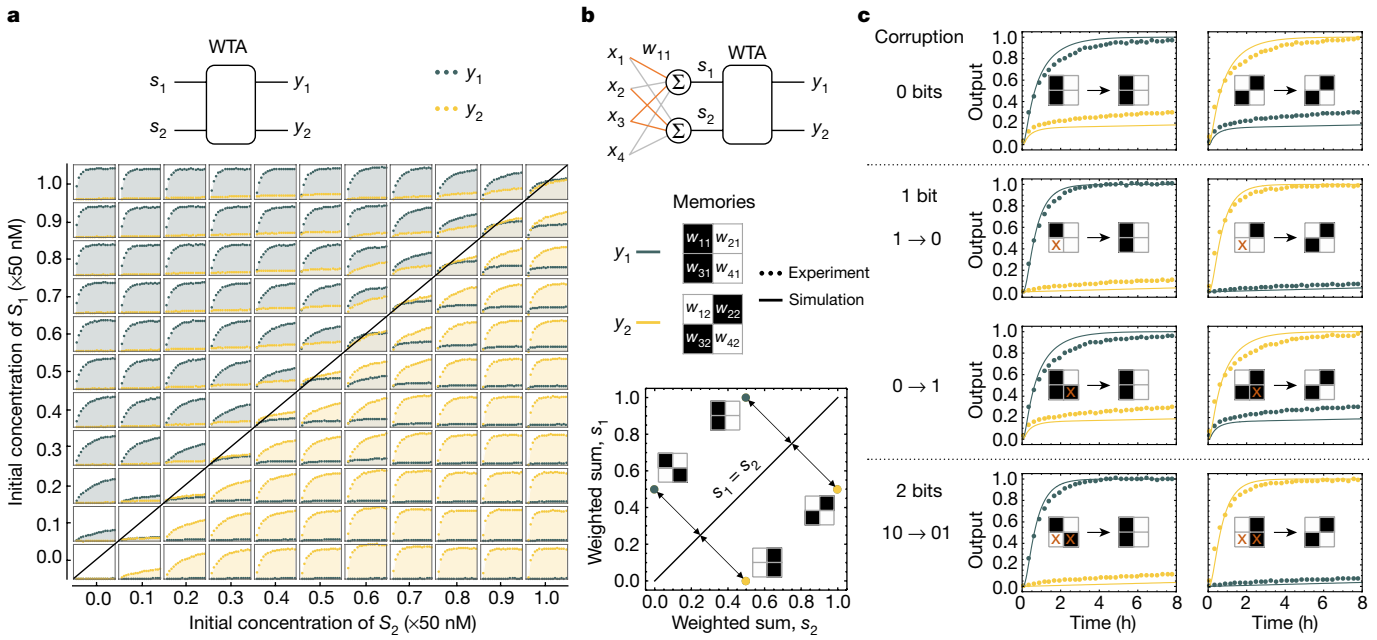


Fig. 2 | Experimental characterization of winner-take-all DNA neural networks. a, Two-species winner-take-all behaviour. The standard concentration is 50 nM (1×). The circuit is composed of two weighted-sum strands (S_1 and S_2), an annihilator molecule ([$Anh_{1,2}$] = 75 nM (1.5×)), two restoration gates ([RG_1] = [RG_2] = 50 nM (1×)), two fuel strands ([YF_1] = [YF_2] = 100 nM (2×)) and two reporters ([Rep_1] = [Rep_2] = 100 nM (2×)). Initial concentrations of S_1 and S_2 are shown as fractions of the standard concentration. The diagonal line indicates equal concentrations of both strands. Fluorescence kinetics data are shown over the course of 2.5 h, normalized using a common minimum and maximum fluorescence level (Methods section 'Data normalization'). To clearly illustrate the difference between the two output trajectories, the background below the data points are shown in the same colour (with some transparency) as the data points. **b**, A 4-bit pattern-recognition circuit. In the weighted-sum layer of the circuit diagram (top left), each wire corresponds to a weight molecule, all wires from the same input require a common fuel strand and all wires to the same output require a common summation gate. Thus, a circuit that can remember any two 4-bit patterns is composed of 25 molecules (4 inputs, 14 molecules in the weighted-sum layer and 7 molecules in the winner-take-all layer). However, a circuit that remembers two specific 4-bit patterns requires only a subset of the wires in the weighted-sum layer, each corresponding to a 1 in the memories (for example, each orange wire in the circuit diagram corresponds to a black pixel in the memories). Thus, the example circuit is composed of 20 molecules (4 fewer weight molecules and 1 fewer fuel strand). In each output-trajectory plot (right), dotted lines indicate fluorescence kinetics data and solid lines indicate simulations. The patterns to the left and right of the arrows indicate input signals and output classifications, respectively. Each orange cross indicates a bit-flip compared to the memories. The initial concentration of each input strand or weight molecule is either 0 or 50 nM; weight fuels (XF_1 and XF_2) are twice the concentration of weight molecules; the initial concentrations of the summation gates, annihilator, restoration gates, restoration fuels and reporters are 100 nM (1×), 400 nM (4×), 100 nM (1×), 200 nM (2×) and 200 nM (2×), respectively, with a standard concentration of 100 nM (details in Supplementary Table 3). In the weighted-sum space (bottom left), the two patterns with two corrupted bits are the same distance (shown as double-headed arrows) from the diagonal line as the two perfect inputs.

reporter molecule Rep_j (Extended Data Fig. 1) to separate the fluorophore- and quencher-labelled strands in the reporter, resulting in increased fluorescence. Overall, the implementation of an arbitrary winner-take-all neural network can be mapped systematically to a seesaw DNA circuit (Extended Data Fig. 2).

We started the experimental demonstration with a two-species winner-take-all function (Fig. 2a), which is similar to approximate majority[17] and consensus network[18] functions. If the initial concentration of one weighted-sum species (S_1 or S_2) is higher than that of the other, then we expect the corresponding output strand (Y_1 or Y_2) to be released catalytically and the fluorescent signal to reach an ideal ON state, while the other output signal remains at an ideal OFF state. The data agree with the expected overall circuit behaviour, and lead to two main observations. First, the circuit computed an ON state faster with a larger difference between the two species, as shown in the plots farther away from the diagonal line in Fig. 2a. This is because the signal-restoration reaction reaches completion faster with a larger amount of catalyst, which is the leftover amount of the winner after the annihilation reaction. Second, among experiments for which the differences between the two species are the same, the circuit maintained a cleaner OFF state with lower initial concentrations of the two species, as shown in the plots that are equidistant to the diagonal line but closer to the bottom left corner of the grid. This is because a small fraction of the weighted-sum strands will interact with a restoration-gate molecule before encountering an annihilator molecule—the stronger the runner-up is (that is, with a higher concentration), the more it can escape the process of being completely annihilated. These observations suggest that the DNA circuit does not yield a perfect winner-take-all behaviour, but that it does compute correctly for competitors that are not too similar to each other and are not both too strong.

Next, we added a weighted-sum layer to the winner-take-all circuit to demonstrate recognition of 4-bit patterns (Fig. 2b). Using the two target patterns as weights, the perfect input patterns each triggered the desired output trajectory to turn ON, indicating that the inputs were recognized correctly. When one or two bits of the input patterns were flipped, either from a 1 to a 0 or vice versa, the circuit still yielded the desired output for all six examples that are classifiable. The other eight possible inputs are not classifiable because they result in equal weighted sums ($s_1 = s_2$). Interestingly, the circuit behaviour was better for the inputs with 2-bit corruptions than for the perfect inputs: the ON trajectories reached completion just as fast and the OFF trajectories remained lower. This result can be understood by looking at the input patterns in the weighted-sum space (Fig. 2b, bottom left): all four inputs are equidistant to the diagonal line and the corrupted patterns are closer to the bottom left corner of the space. Because catalytic reactions are used to implement weight multiplication, together with thresholding reactions, the circuit can also handle a range of input concentration that varies from the ideal high or low concentration (Extended Data Fig. 3).

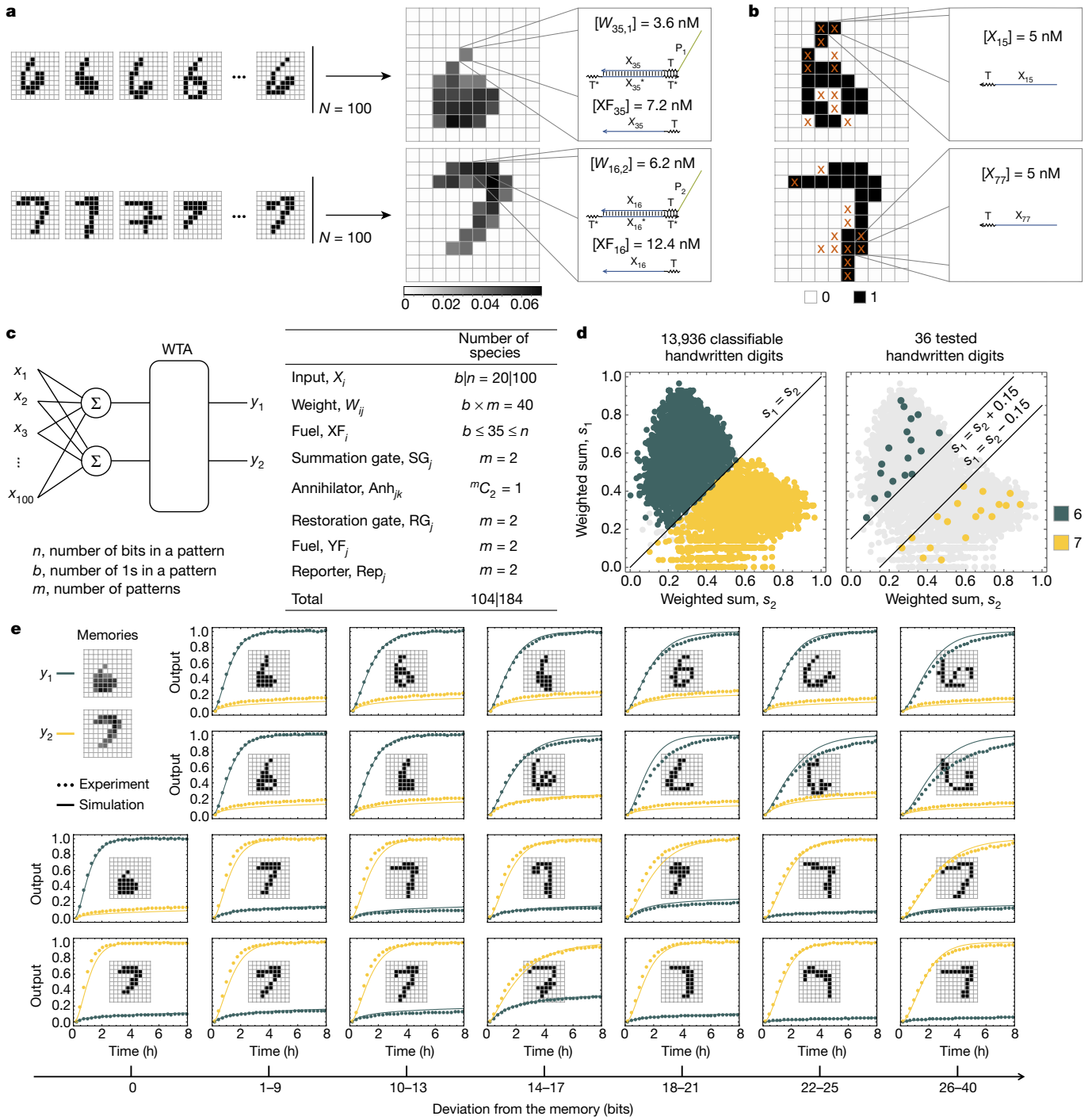


Fig. 3 | A winner-take-all DNA neural network that recognizes 100-bit patterns as one of two handwritten digits. a, Weights determined as the average of 100 '6's and '7's from the MNIST database. The value of each pixel (for example, 0.036 for the 35th pixel in '6' and 0.062 for the 16th pixel in '7') was used to determine the concentration of each weight molecule, relative to a standard concentration of 100 nM (for example, $[W_{35,1}] = 3.6$ nM and $[W_{16,2}] = 6.2$ nM). The concentrations of the fuel strands that facilitate the weight multiplication reactions were twice that of their respective weight molecules. **b**, Example binary inputs with each 1 and 0 corresponding to the presence and absence of an input strand, respectively. The concentration of each input strand present was $1/b \times 100$ nM $= 5$ nM, where $b = 20$ is the total number of 1s in each input. The orange crosses indicate bit-flips compared to the memories (that is, weight matrices) shown in **a**. There are 12 flipped bits in each example. Because the total number of 1s in each input pattern is the same as the total number of non-zero weights in the memories, it is always the case that half of the flipped bits are associated with non-zero weights. **c**, Circuit diagram and the number of distinct species in the circuit. For the total number of species, the two values correspond to the number of species for a specific number b of inputs (left) and for all n possible inputs (right). **d**, The 13,936 classifiable digits (left; large green and yellow points) correspond to 98% of all '6's and '7's in the MNIST database. Test input patterns were chosen (right; large green and yellow points; 36 in total) on the basis of their locations in the weighted-sum space. The lines labelled $s_1 = s_2 \pm 0.15$ indicate a 15% margin to the diagonal line, within which we expect the pattern recognition to be experimentally difficult. The light grey points correspond to non-classifiable (left) or non-tested (right) digits. **e**, Recognizing handwritten digits with up to 30 flipped bits compared to the 'remembered' digits. Dotted lines indicate fluorescence kinetics data and solid lines indicate simulations. The input pattern is shown in each plot. Note that 40 is the maximum number of flipped bits because all patterns have exactly 20 1s. Weights and inputs are listed in Supplementary Table 2. The initial concentrations of all species are listed in Extended Data Fig. 10 (details in Supplementary Table 3).

To understand the theoretical limits of the scalability and power of winner-take-all DNA neural networks, in the context of simply using the target patterns as weights, we now address the following three questions. The first is the number of distinct target patterns that can be remembered simultaneously. Any set of patterns that consists of the same number of 1s can be remembered (Methods, Theorem 1). For example, the largest set of 9-bit patterns that can be remembered, each consisting of five 1s, consists of ${}^{9}C_{5} = 126$ patterns. Moreover, any set of patterns can be remembered if it does not contain a pattern in which all 1s are a subset of 1s in another pattern (Methods, Theorem 2). The second question concerns which corrupted patterns can be recognized. All patterns with fewer than $b - o$ corrupted bits can be recognized, where b is the total number of 1s and o is the maximum number of overlapped 1s in all target patterns (Methods, Theorem 3). For example, all patterns with fewer than three corrupted bits can be recognized for the 9-bit target patterns 'L' and 'T' shown in Fig. 1b, because $b = 5$ and $o = 2$. Moreover, some patterns with more than $b - o$ corrupted bits can still be recognized; for example, in all possible 9-bit patterns, there are 128, 102 and 30 patterns with three, four and five corrupted bits, respectively, that can be recognized as 'L' or 'T'. We chose 28 example 9-bit patterns with an increasing number of corrupted bits from one to five, and demonstrated that the DNA neural network correctly classified all examples (Extended Data Fig. 4). The final question asks how the size of the DNA circuit scales with an increasing number of more complex patterns. In general, constructing a network that can remember m distinct n-bit patterns requires n input strands, $n \times m$ weight molecules and n fuel strands for weight multiplication, m summation gates, ${}^{m}C_{2}$ annihilators, m gates and m fuel strands for signal restoration, and m reporters, totalling $n \times m + 2n + 4m + {}^{m}C_{2}$ molecules. However, for a specific set of target patterns, only a subset of the weight molecules are required, each corresponding to a 1 in the patterns.

To demonstrate the scalability and power of winner-take-all DNA neural networks experimentally, we chose a task that is visually interesting: recognizing handwritten digits. Some aspects of this task are computationally non-trivial, such as distinguishing a sloppy '4' from a sloppy '9'. The patterns of digits were taken from the Modified National Institute of Standards and Technology (MNIST) database[19], which is commonly used to test machine learning algorithms[20]. We converted the original patterns to binary patterns with 20 1s on a 10×10 grid, averaged 100 example '6' and '7' patterns, and selected and normalized the top 20 pixels as weights (Fig. 3a, Methods section 'Neural network training and testing'). The value of each analogue weight was then implemented with the concentration of a weight molecule. The test inputs remained binary patterns, in which each 1 or 0 corresponded to the presence or absence of an input strand, respectively (Fig. 3b). The theoretical limits of the winner-take-all neural networks with analogue weights are similar to those with binary weights (Methods, Theorems 4 and 5). In total, 104 distinct molecules were used for testing any specific input pattern out of 184 distinct molecules for all possible inputs (Fig. 3c).

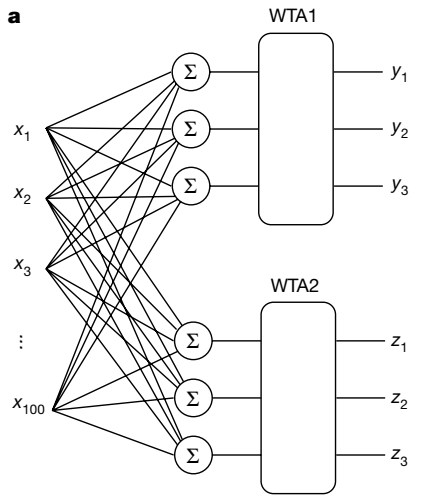
In the MNIST database, there are more than 14,000 example handwritten '6' and '7' digits. On the basis of the understanding that we have established from the experimental characterization of smaller winner-take-all circuits, we looked at all example patterns in the weighted-sum space (Fig. 3d, Extended Data Fig. 5a): 2% of the patterns are on the wrong side of the diagonal line, which means that it is impossible for the DNA circuit to recognize them correctly; 8% of the patterns are fairly close to the diagonal line (within a 15% margin), which we expect to be experimentally difficult; however, the remaining 90% of the patterns are far enough from the diagonal line that we expect correct recognition. Therefore, we chose 36 representative example patterns from the last category, ensuring both uniform distribution in the weighted-sum space and the full range of bit deviation from the memories (Methods section 'Neural network training and testing'). As shown in the experimental data (Fig. 3e, Extended Data Fig. 5d), the perfect patterns (the weights converted to binary) each yield the desired circuit output. More importantly, patterns that increasingly deviate from the memories were also recognized, with up to 30 flipped bits. Similar to observations in the smaller DNA neural networks, some of the patterns that are visually more challenging to recognize are not necessarily more difficult for the DNA circuit—a desirable property of the winner-take-all computation.

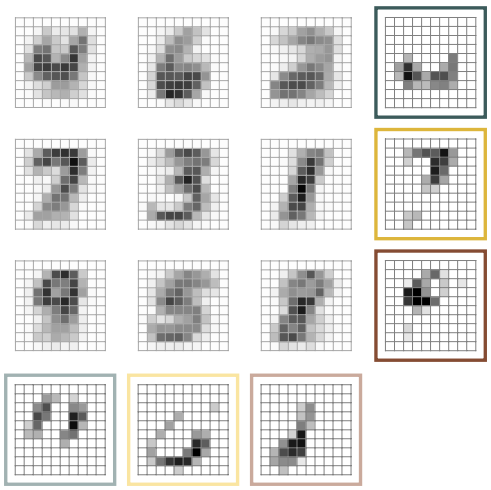
We have shown that the winner-take-all DNA neural networks scale well to more complex patterns. Next, we explore whether they could also be used to remember an increasing number of distinct patterns simultaneously. The pairwise-annihilation approach alone is not well suited for scaling up the number of patterns because the number of annihilators grows quadratically with the number of patterns. We show that the three-species winner-take-all function was still robust enough (Extended Data Fig. 6a) to allow the construction of a DNA neural network that remembers three 100-bit patterns. However, the competition became harder with more competitors: the reaction rates for multiple annihilation pathways could be matched approximately but not perfectly (Methods section 'Sequence design', Extended Data Fig. 6b, c), and it took much longer for the annihilation reactions to yield a winner and for the signal level of the winner to be fully restored (Extended Data Fig. 7). Using the same method, it would be difficult to construct networks that remember more patterns. We therefore propose an alternative approach that first divides the target patterns into groups and then uses multiple distinct group identities to classify the patterns (Fig. 4a). The nine digits '1'–'9' can be divided into three groups in two ways (shown as three rows and three columns in Fig. 4b), such that a pair of outputs corresponds uniquely to each digit (Fig. 4d). For example, a '4' is recognized if and only if $y_1 = 1$ and $z_1 = 1$ (where y_1 is the output identifying the first row and z_1 is the output identifying the first column). With this grouping approach, nine distinct patterns can be recognized using only $\sqrt{{}^{9}C_{2} \times 2} = 6$ annihilators, which would otherwise require ${}^{9}C_{2} = 36$ annihilators. In total, 225 distinct molecules were used for testing any specific input pattern out of 305 distinct molecules for all possible inputs (Fig. 4c).

We determined the weights for each group using a simple 'average then subtract' method (Fig. 4b): take the average of 100 examples per in-group digit, subtract the average of 100 examples per out-of-group digit, then select and normalize the top 20 pixels (Methods section 'Neural network training and testing'). The trade-off of the grouping approach is that fewer example patterns can be recognized. With the best grouping, 47% of the patterns can potentially be recognized, of which 48% are experimentally feasible (with a 15% margin to the diagonal line in the normalized weighted-sum space). In general, with the same circuit complexity, this alternative approach enables a larger set of distinct target patterns to be classified, but with less accuracy. Nonetheless, as shown in the experimental data, the circuit yields the desired pair of outputs for 99 representative example patterns (Fig. 4d, e).

To facilitate the design of winner-take-all DNA neural networks, we developed an online software tool. The WTA Compiler[21] (Extended Data Fig. 8) converts a user-defined set of memories and test patterns into program code that describes a DNA neural network, which can then be used to simulate the kinetics of the network. It also provides sequences of the DNA strands that are required to construct the DNA neural network experimentally.

It is interesting to compare the performance of winner-take-all neural networks with logic circuits. For example, it is possible to distinguish whether a 9-bit pattern is more similar to 'L' or 'T' using a circuit consisting of 8 logic gates, for all input patterns that we have tested experimentally. However, a more complex circuit consisting of 21 logic gates is required to correctly compute the output for all classifiable patterns (Extended Data Fig. 9a). Similarly, the 100-bit handwritten digits can be recognized by circuits with up to 23 logic gates, if only the example patterns that we have tested experimentally are considered. But these logic circuits perform poorly when tested against the entire MNIST database (Extended Data Fig. 9b). To match the theoretical limit of winner-take-all neural networks, measured by the percentage of classifiable patterns, much more complex logic circuits are needed. Importantly,

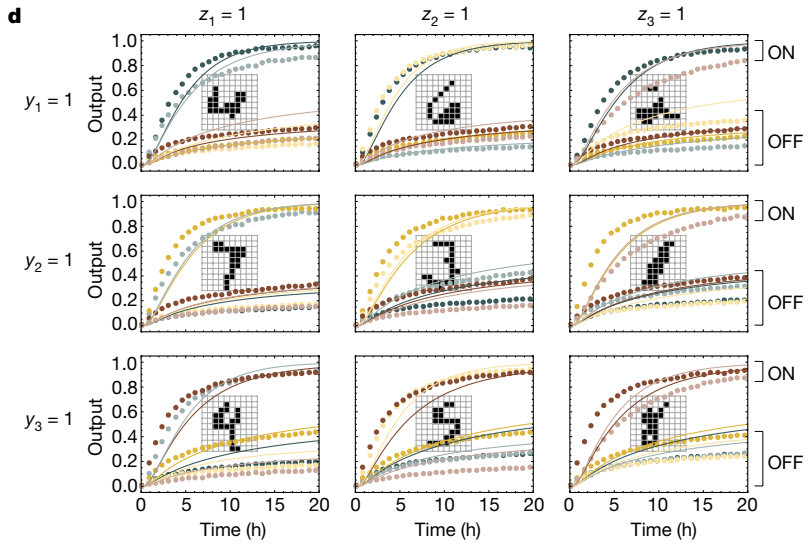




c

	Number of species
Input, X_i	$b\|n = 20\|100$
Weight, W_{ij}	$b \times 2\sqrt{m} = 120$
Fuel, XF_i	$b \le 55 \le n$
Summation gate, SG_j	$\sqrt{m} \times 2 = 6$
Annihilator, Anh_{jk}	${}^{\sqrt{m}}C_2 \times 2 = 6$
Restoration gate, RG_j	$\sqrt{m} \times 2 = 6$
Fuel, YF_j	$\sqrt{m} \times 2 = 6$
Reporter, Rep_j	$\sqrt{m} \times 2 = 6$
Total	225|305

n, number of bits in a pattern
b, number of 1s in a pattern
m, number of patterns



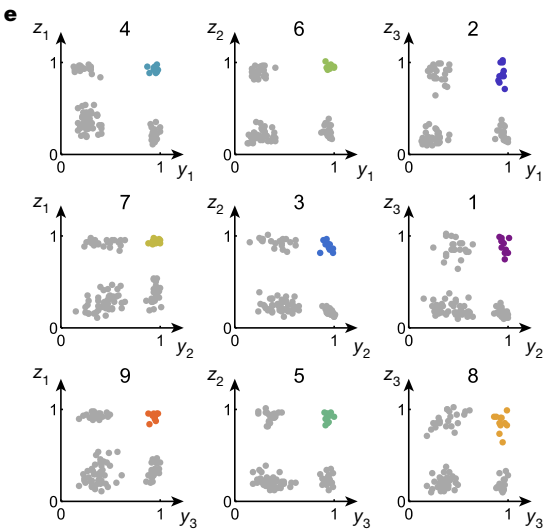


Fig. 4 | A winner-take-all DNA neural network that recognizes 100-bit patterns as one of nine handwritten digits. a, Circuit diagram for recognizing nine distinct patterns using a grouping approach. WTA1 and WTA2 are two separate winner-take-all functions that each yields a distinct set of outputs (y_j and z_j). **b**, Weights determined using an 'average then subtract' method. The average of 100 example digits from the MNIST database are shown grouped by rows (corresponding to outputs y_j) and columns (corresponding to outputs z_j). The weight matrix for each group (boxed patterns; colours correspond to the respective output trajectories in **d**) is the average of all in-group digits less the average of all out-of-group digits. Using this weight matrix, the fraction of experimentally feasible test patterns from all examples in the MNIST database was calculated for all possible ways of grouping the nine digits and the best grouping was chosen and shown here. **c**, Number of distinct species in the circuit in **a**. For the total number of species, the two values correspond to the number of species for a specific number *b* of inputs (left) and for all *n* possible inputs (right). **d**, Fluorescence kinetics data (dotted lines) and simulations (solid lines) of the circuit behaviour with nine representative input patterns (shown in the plots). **e**, Fluorescence level of each pair of outputs at 24 h or longer after the inputs were added, collected from 99 experiments with 11 example patterns per digit. Each coloured point corresponds to an example pattern from the labelled class of digit; each grey point corresponds to an out-of-class example pattern. Weights and inputs are listed in Supplementary Table 2. The initial concentrations of all species are listed in Extended Data Fig. 10 (details in Supplementary Table 3).

varying the concentrations of the weight molecules in the winner-take-all neural networks would enable the same set of DNA molecules to be used for different pattern-classification tasks. By contrast, without reconfigurable circuit architectures, a different set of DNA molecules would be required for a logic circuit that performs a different task.

The power of winner-take-all DNA neural networks could be explored further in several directions. Instead of the pairwise-annihilation approach, a winner could be selected by utilizing competing resources[5,6], which could potentially lead to more scalable and accurate pattern recognition. It could also provide the possibility of selecting several winners instead of just one, which in theory is computationally more powerful[4]. Extending the circuit construction from single-layer to multi-layer winner-take-all computation, or simply allowing the outputs of winner-take-all circuits to be connected to downstream logic circuits, could enable more sophisticated pattern recognition (such as involving translated and rotated patterns)[22]. Using a variable-gain amplifier[23,24], winner-take-all DNA circuits could be adapted to process analogue inputs, which would enable a wider range of signal-classification tasks, including applications in detecting complex disease profiles that consist of mRNA and microRNA signals. With aptamers[25,26], more diverse biomolecules could be detected.

The fact that we were able to use target patterns as weights in winner-take-all DNA neural networks opens up immediate possibilities for embedding learning within autonomous molecular systems. With one additional circuit component that actives weight molecules during a supervised training process, the DNA circuits would be capable of activating a specific set of wires in the weight-multiplication layer when exposed to a specific set of patterns. As widely discussed in experimental[27] and theoretical[28–30] studies, learning—the most desirable property

of biochemical circuits—would allow artificial molecular machines to adapt their functions on the basis of environmental signals during autonomous operations.

Online content

Any Methods, including any statements of data availability and Nature Research reporting summaries, along with any additional references and Source Data files, are available in the online version of the paper at https://doi.org/10.1038/s41586-018-0289-6.

Received: 30 October 2017; Accepted: 18 April 2018;
Published online: 04 July 2018

1. Wadhams, G. H. & Armitage, J. P. Making sense of it all: bacterial chemotaxis. *Nat. Rev. Mol. Cell Biol.* **5**, 1024–1037 (2004).
2. Mori, K., Nagao, H. & Yoshihara, Y. The olfactory bulb: coding and processing of odor molecule information. *Science* **286**, 711–715 (1999).
3. Qian, L., Winfree, E. & Bruck, J. Neural network computation with DNA strand displacement cascades. *Nature* **475**, 368–372 (2011).
4. Maass, W. On the computational power of winner-take-all. *Neural Comput.* **12**, 2519–2535 (2000).
5. Kim, J., Hopfield, J. & Winfree, E. Neural network computation by *in vitro* transcriptional circuits. *Adv. Neural Inf. Process. Syst.* **17**, 681–688 (2005).
6. Genot, A. J., Fujii, T. & Rondelez, Y. Scaling down DNA circuits with competitive neural networks. *J. R. Soc. Interface* **10**, 20130212 (2013).
7. Muroga, S. *Threshold Logic and its Applications* (Wiley Interscience, New York, 1971).
8. Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl Acad. Sci. USA* **79**, 2554–2558 (1982).
9. Yurke, B., Turberfield, A. J., Mills, A. P., Simmel, F. C. & Neumann, J. L. A DNA-fuelled molecular machine made of DNA. *Nature* **406**, 605–608 (2000).
10. Zhang, D. Y. & Seelig, G. Dynamic DNA nanotechnology using strand-displacement reactions. *Nat. Chem.* **3**, 103–113 (2011).
11. Qian, L. & Winfree, E. Scaling up digital circuit computation with DNA strand displacement cascades. *Science* **332**, 1196–1201 (2011).
12. Thubagere, A. J. et al. Compiler-aided systematic construction of large-scale DNA strand displacement circuits using unpurified components. *Nat. Commun.* **8**, 14373 (2017).
13. Zhang, D. Y. Cooperative hybridization of oligonucleotides. *J. Am. Chem. Soc.* **133**, 1077–1086 (2011).
14. Redgrave, P., Prescott, T. J. & Gurney, K. The basal ganglia: a vertebrate solution to the selection problem? *Neuroscience* **89**, 1009–1023 (1999).
15. Zhang, D. Y. & Winfree, E. Control of DNA strand displacement kinetics using toehold exchange. *J. Am. Chem. Soc.* **131**, 17303–17314 (2009).
16. Yurke, B. & Mills, A. P. Using DNA to power nanostructures. *Genet. Program. Evol. Mach.* **4**, 111–122 (2003).
17. Cardelli, L. & Csikász-Nagy, A. The cell cycle switch computes approximate majority. *Sci. Rep.* **2**, 656 (2012).
18. Chen, Y.-J. et al. Programmable chemical controllers made from DNA. *Nat. Nanotechnol.* **8**, 755–762 (2013).
19. LeCun, Y., Cortes, C. & Burges, C. J. The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/index.html.
20. Deng, L. The MNIST database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Process. Mag.* **29**, 141–142 (2012).
21. Cherry, K. M. WTA Compiler. http://www.qianlab.caltech.edu/WTAcompiler/ (2017).
22. Rojas, R. *Neural Networks: A Systematic Introduction* (Springer, Berlin, 2013).
23. Zhang, D. Y. & Seelig, G. DNA-based fixed gain amplifiers and linear classifier circuits. In *DNA 2010: DNA Computing and Molecular Programming* (eds Sakakibara, Y. & Mi, Y.) 176–186 (Springer, 2011).
24. Chen, S. X. & Seelig, G. A DNA neural network constructed from molecular variable gain amplifiers. In *DNA 2017: DNA Computing and Molecular Programming* (eds Brijder, R. & Qian, L.) 110–121 (Springer, Cham, 2017).
25. Cho, E. J., Lee, J.-W. & Ellington, A. D. Applications of aptamers as sensors. *Annu. Rev. Anal. Chem.* **2**, 241–264 (2009).
26. Li, B., Ellington, A. D. & Chen, X. Rational, modular adaptation of enzyme-free DNA circuits to multiple detection methods. *Nucleic Acids Res.* **39**, e110 (2011).
27. Pei, R., Matamoros, E., Liu, M., Stefanovic, D. & Stojanovic, M. N. Training a molecular automaton to play a game. *Nat. Nanotechnol.* **5**, 773–777 (2010).
28. Fernando, C. T. et al. Molecular circuits for associative learning in single-celled organisms. *J. R. Soc. Interface* **6**, 463–469 (2009).
29. Aubert, N. et al. Evolving cheating DNA networks: a case study with the rock–paper–scissors game. In *ECAL 2013: Advances in Artificial Life* (eds Liò, P. et al.) 1143–1150 (MIT Press, Cambridge, 2013).
30. Lakin, M. R., Minnich, A., Lane, T. & Stefanovic, D. Design of a biochemical circuit motif for learning linear functions. *J. R. Soc. Interface* **11**, 20140902 (2014).

Acknowledgements We thank R. M. Murray for sharing an acoustic liquid-handling robot. We thank C. Thachuk and E. Winfree for discussions and suggestions. K.M.C. was supported by a NSF Graduate Research Fellowship. L.Q. was supported by a Career Award at the Scientific Interface from the Burroughs Wellcome Fund (1010684), a Faculty Early Career Development Award from NSF (1351081), and the Shurl and Kay Curci Foundation.

Reviewer information *Nature* thanks R. Schulman and the other anonymous reviewer(s) for their contribution to the peer review of this work.

Author contributions K.M.C. developed the model, designed and performed the experiments, and analysed the data; K.M.C. and L.Q. wrote the manuscript; L.Q. initiated and guided the project.

Competing interests The authors declare no competing interests.

Additional information
Supplementary information is available for this paper at https://doi.org/10.1038/s41586-018-0289-6.
Reprints and permissions information is available at http://www.nature.com/reprints.
Correspondence and requests for materials should be addressed to L.Q.
Publisher's note: Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

METHODS

Sequence design. All DNA strands used in the winner-take-all neural networks were composed of long branch-migration domains and short toehold domains. Owing to the modularity of the previously developed seesaw DNA motif[3,11] and the extended new circuit component—the annihilator—the sequence design was performed at the domain level. A pool of domain sequences was generated according to a set of design heuristics that have previously been experimentally validated[12]. All domains used a three-letter code (A, T and C) to reduce secondary structure and undesired strand interactions. No domain sequences include runs of more than four consecutive As or Ts or more than three consecutive Cs, which reduces synthesis errors. All domain sequences had between 30% and 70% C-content so all double-stranded complexes would have similar melting temperatures. Finally, no pairs of domain sequences share a matching sequence longer than 35% of the domain length, and all pairs have at least 30% different nucleotides. This ensures that a strand with a mismatched branch-migration domain will not complete strand displacement initiated from either the 3′ or the 5′ end. In addition to a 15-nucleotide sequence pool used in previous work[3,11,12], a 20-nucleotide sequence pool was generated and used in the weight multiplication layers because of the large number of molecules used here. The two sequence pools were checked to ensure that the same pairwise criteria were met. All domains included the clamp design introduced previously[11], to reduce leak reactions between initial gate species.

All molecular complexes shared a 5-nucleotide universal toehold domain[3,11,12]. The annihilator complexes had 7-nucleotide toeholds composed of the 5-nucleotide universal toehold and a 2-nucleotide extension that matched the 2 nucleotides adjacent to the toehold on the upstream seesaw gate. This increased the binding energy and thus the effective strand-displacement reaction rate between the annihilator complexes and the weighted-sum strands, compared to that between the signal-restoration gates and the weighted-sum strands.

To ensure 'fair competition' between the weighted-sum species (that is, same rates for all pairwise-annihilation reactions), all annihilators within a set of winner-take-all computations had identical toehold extensions, and the weighted-sum strands had the same single-nucleotide dangle to keep the binding energies consistent within a winner-take-all computation. Here, we used up to two sets of three annihilators. The extensions and dangle sequences were chosen by estimating the binding energies using NUPACK[31], and the sequences for the second set of annihilators were chosen with similar energies to those of the first set that worked well in the three-species winner-take-all experiments (Extended Data Fig. 6a). In addition, the rate of an annihilation reaction could depend on the sequence of the branch-migration domains. We measured the rates of 15 catalytic gates, and selected two groups of three gates with the closest rates (Extended Data Fig. 6b, c). By using these gates for signal restoration, the branch-migration domains in the annihilators were determined simultaneously, because the signal-restoration gates and annihilators share the same branch-migration domains (Extended Data Fig. 1).

All DNA sequences are listed in Supplementary Table 1.

Neural-network training and testing. The winner-take-all DNA neural network was tested on patterns derived from the MNIST handwritten-digit database[19]. The training and testing sets were downloaded and merged into a single database, and all example patterns of digits '1'–'9' were retained, totalling 63,097 images. The original MNIST dataset consists of weight-centred grey-scale images on a 28×28 grid. Here, we used binary patterns on a 10×10 grid. First, the images were rescaled to a 12×12 grid using Gaussian resampling. The largest 20 bits in each image were set to 1 and the remaining bits were set to 0. Finally, the digits were re-centred on a 10×10 grid on the basis of their bounding boxes.

We made a conscious effort to train the neural networks using a simple algorithm. In the neural networks that remember two or three handwritten digits, for each digit, the weight matrices were the average of the first 100 example patterns in the database, restricted to the 20 most common bits (that is, the ones with the largest averaged values), and normalized to sum to 1. For the nine-digit network, all digits were divided into three groups in two ways. For each group, the weight matrix was the average of the first 100 examples of the three in-group digits less the average of the first 100 examples of the six out-of-group digits. The 20 most common bits were retained, and all weight matrices were normalized to sum to 1.15, to shift the test patterns into a more ideal area in the weighted-sum space. The fraction of experimentally feasible test patterns (with a 15% margin to the diagonal line in the weighted-sum space for all pairs of species) was calculated for all ways of grouping the nine digits, and the best grouping was chosen. The classification performance of the network using weights determined by non-negative least squares was only slightly better than the performance using weights from the simple 'average then subtract' method (54% versus 47%).

Experimentally tested input patterns were chosen to represent the whole weighted-sum space as well as the full range of bit deviation from the memories of the networks. To choose a set of test patterns for a digit, all correctly classified examples of that digit with at least a 15% margin in the weighted-sum space were divided into six corruption classes. The weighted sums for the digits in each class were then clustered using the k-medoids algorithm, and an example test pattern was chosen randomly from each cluster according to a uniform distribution. This ensured that the test patterns represented the whole weighted-sum space and not just the most common digits.

Weights and inputs used in all experiments are listed in Supplementary Table 2. By exporting each sheet of the Excel file to a .csv file and uploading it to the WTA Compiler[21], the weights and inputs can be visually displayed, the inputs analysed in their weighted-sum space, the kinetics behaviour of the winner-take-all DNA neural network simulated and DNA sequences generated.

DNA oligonucleotide synthesis. All DNA strands were purchased from Integrated DNA Technologies (IDT). The reporter strands with fluorophores and quenchers were purified (HPLC) and the other strands were unpurified (standard desalting). All strands were shipped lyophilized then resuspended at 100 μM in Tris-EDTA (TE) buffer, pH 8.0, and stored at 4 °C.

Annealing protocol and buffer condition. Annihilator and gate complexes were prepared for annealing at 45 μM with top and bottom strands in a 1:1 ratio. Reporters were prepared at 20 μM with top quencher strands in 20% excess of bottom strands. The buffer for all experiments and annealed complexes was TE with 12.5 mM Mg^{2+}. Complexes were annealed in a thermal cycler (Eppendorf) by heating to 90 °C for 5 min and then cooling to 20 °C at a rate of 0.1 °C per 6 s.

Purification. Annealed annihilator and gate complexes were purified using 12% polyacrylamide gel electrophoresis (PAGE). Double-stranded complex bands were cut from the gel, chopped into pieces and incubated for 24 h at room temperature in TE buffer with 12.5 mM Mg^{2+} to allow DNA to diffuse into the buffer. The solution with purified complexes was recovered and concentrations were determined with NanoDrop (Thermo Fisher). Weight matrices for the DNA neural networks that remember handwritten digits had 20 gate complexes for each neuron. These gates (weight molecules) were annealed individually and then mixed together in the appropriate ratio, on the basis of the values of the weights. This mixture was then purified via PAGE, recovered and the concentration determined by NanoDrop using the weighted-average extinction coefficient.

Fluorescence spectroscopy. Fluorescence kinetics data were collected every 2, 3 or 4 min, depending on the overall length of the experiment, using a microplate reader (Synergy H1, Biotek). Excitation (emission) wavelengths were 496 nm (525 nm) for dye ATTO488, 555 nm (582 nm) for dye ATTO550 and 598 nm (629 nm) for dye ATTO590. Experiments were performed in 96-well plates (Corning) with 160-μl reaction mixture per well for the nine-digit experiments and 200-μl reaction mixture per well for all other experiments. Experiments were performed at a standard concentration of 100 nM for all 4-bit and 100-bit pattern recognition and at a standard concentration of 50 nM for all other experiments. Initial concentrations of all species are listed in Extended Data Fig. 10. Detailed protocols for all experiments are listed in Supplementary Table 3.

In the nine-digit experiments, six distinct output trajectories were read using three distinct fluorophores. Every experiment was run twice, each having half of the outputs connected to fluorophore-labelled reporters and the other half to non-fluorophore-labelled reporters. Combining the output trajectories from each pair of experiments into a single plot allows the observation of all six outputs simultaneously.

Data normalization. All data were normalized from raw fluorescence level to standard concentration, which is the maximum concentration of an output strand Y_j released from gate RG_j and interacted with a double-stranded reporter molecule Rep_j. The fluorescence level that corresponds to standard concentration (1×) was obtained from the average of the final five measurements from the highest signal produced from gate RG_j on a plate. Negligible concentration (0×) corresponds to the background fluorescence of the reaction mixture before any reporter molecules have been triggered, which was obtained from the first measurement of the lowest signal produced from gate RG_j on a plate. All experiments on a single plate were normalized together, allowing direct comparison between the output of a network for different input patterns. In the two-species winner-take-all experiments shown in Extended Data Fig. 3, the first six columns of data were measured on one plate and the last five columns measured on another. In the 9-bit pattern-recognition experiments shown in Extended Data Fig. 4, the input patterns with 0–2 corrupted bits were measured on one plate and those with 3–5 corrupted bits were measured on another.

Model and simulations. Mass-action simulation were performed using the same set of reactions and rate constants developed in the seesaw model[11], with four additional reactions to model pairwise annihilation:

$$S_j + \mathrm{Anh}_{jk} \underset{k_r}{\overset{k_f}{\rightleftharpoons}} S_j{:}\mathrm{Anh}_{jk}$$

$$S_k + \mathrm{Anh}_{jk} \underset{k_r}{\overset{k_f}{\rightleftharpoons}} S_k{:}\mathrm{Anh}_{jk}$$

$$S_j{:}\mathrm{Anh}_{jk} + S_k \xrightarrow{k_f} \varnothing$$

$$S_k{:}\mathrm{Anh}_{jk} + S_j \xrightarrow{k_f} \varnothing$$

Here, $k_f = 2 \times 10^6\ M^{-1}s^{-1}$, which is the same as the forward rate constant of the thresholding reaction in the seesaw model[11]. The reverse rate constant $k_r = 0.4\ s^{-1}$ was determined using the experimental data shown in Extended Data Fig. 3a. This rate constant is of the same order as found in a previous study of cooperative hybridization[13]. Similar to the spurious reactions in the original seesaw model, temporary toehold binding between any single-stranded species and any annihilator (or intermediate annihilator species listed above) are also included here.

Code availability. Simulation code is available at the WTA Compiler website[21].

Theoretical limits of the power of winner-take-all neural networks. The winner-take-all function shown in Fig. 1a is defined to have:

Inputs	$\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$
Weights	$W = (\boldsymbol{w}_1^\top, \boldsymbol{w}_2^\top, \ldots, \boldsymbol{w}_m^\top)$, with $\boldsymbol{w}_j = (w_{1j}, w_{2j}, \ldots, w_{nj})$
Weighted sums	$s_j = \boldsymbol{w}_j \cdot \boldsymbol{x}$
Outputs	$y_j = \begin{cases} 1 & \text{if } s_j > s_k \ \forall\ k \neq j \\ 0 & \text{otherwise} \end{cases}$

Definition 1. Let $X = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^m\}$ be a set of m patterns, each with n bits. Let an example pattern from X be $\boldsymbol{x}^\alpha = (x_1^\alpha, x_2^\alpha, \cdots, x_n^\alpha)$, with $x_i^\alpha \in \{0, 1\}$. We say that a winner-take-all neural network with weights W remembers X if $y_\alpha = 1$ for all $1 \leq \alpha \leq m$ (and $y_j = 0$ for all $j \neq \alpha$) when $\boldsymbol{x} = \boldsymbol{x}^\alpha$.

Theorem 1. *If X is a set of m distinct n-bit patterns, each containing exactly b 1s, then the winner-take-all neural network with $W = (\boldsymbol{w}_1^\top, \boldsymbol{w}_2^\top, \ldots, \boldsymbol{w}_m^\top)$ and $\boldsymbol{w}_j = (w_{1j}, w_{2j}, \ldots, w_{nj}) = \boldsymbol{x}^j$ (that is, $w_{ij} = x_i^j$) remembers X.*

Proof. Consider this network on input $\boldsymbol{x} = \boldsymbol{x}^\alpha$. First, for $j = \alpha$, we calculate $s_\alpha = \boldsymbol{x}^\alpha \cdot \boldsymbol{x}^\alpha = b$. Second, for $j \neq \alpha$, $\boldsymbol{x}^j \neq \boldsymbol{x}^\alpha$. Because the number of 1s in both of these patterns is b, the number of indices at which the bits are both 1 is strictly less than b. Therefore, $s_j = \boldsymbol{x}^j \cdot \boldsymbol{x}^\alpha < b$. Putting the first and second calculations together, we conclude that $s_\alpha > s_j$ and thus $y_\alpha = 1$ and $y_j = 0$ for all $j \neq \alpha$.

The next theorem is a generalization of Theorem 1.

Theorem 2. *If X is a set of m distinct n-bit patterns, and the 1s in any example pattern $\boldsymbol{x}^\alpha$ is not a subset of the 1s in another pattern $\boldsymbol{x}^\beta$ (that is, no two example patterns satisfy $\boldsymbol{x}^\alpha \cdot \boldsymbol{x}^\beta = \boldsymbol{x}^\alpha \cdot \boldsymbol{x}^\alpha$), then the winner-take-all neural network with $W = (\boldsymbol{w}_1^\top, \boldsymbol{w}_2^\top, \ldots, \boldsymbol{w}_m^\top)$ and $\boldsymbol{w}_j = \boldsymbol{x}^j$ remembers X.*

Proof. Consider this network on input $\boldsymbol{x} = \boldsymbol{x}^\alpha$. First, $s_\alpha = \boldsymbol{x}^\alpha \cdot \boldsymbol{x}^\alpha$ and is equal to the total number of 1s in $\boldsymbol{x}^\alpha$. Second, for $j \neq \alpha$, $s_j = \boldsymbol{x}^j \cdot \boldsymbol{x}^\alpha \neq \boldsymbol{x}^\alpha \cdot \boldsymbol{x}^\alpha$. Third, for all j, $s_j = \boldsymbol{x}^j \cdot \boldsymbol{x}^\alpha \leq \boldsymbol{x}^\alpha \cdot \boldsymbol{x}^\alpha = s_\alpha$. Putting these three constraints together, we conclude that $s_\alpha > s_j$ and thus $y_\alpha = 1$ and $y_j = 0$ for all $j \neq \alpha$.

Definition 2. In a winner-take-all neural network with $W = (\boldsymbol{w}_1^\top, \boldsymbol{w}_2^\top, \ldots, \boldsymbol{w}_m^\top)$ and $\boldsymbol{w}_j = \boldsymbol{x}^j$, we that say each $\boldsymbol{x}^j$ is a *memory*. We say that the network *recognizes* input $\boldsymbol{x}$ as memory $\boldsymbol{x}^\alpha$ if $y_\alpha = 1$ (and $y_j = 0$ for all $j \neq \alpha$). We say that a pattern $\boldsymbol{x}$ has c *corrupted* bits compared to a memory $\boldsymbol{x}^\alpha$ (or has c-bit *deviation* from $\boldsymbol{x}^\alpha$) if the number of indices at which the bits are different (that is, one bit is 0 and the other is 1 or vice versa) in $\boldsymbol{x}$ and $\boldsymbol{x}^\alpha$ is exactly c. We say that two memories $\boldsymbol{x}^\alpha$ and $\boldsymbol{x}^\beta$ have o *overlapped* bits if the number of indices at which the bits are both 1 in these memories is exactly o.

Theorem 3. *If x is a pattern with $c < b - o$ corrupted bits compared to a memory x^α, where b is the total number of 1s in x^α and o is the maximum number of overlapped bits in x^α and x^j for all $j \neq \alpha$, then the winner-take-all neural network recognizes x as x^α.*

Proof. Let c_0 be the number of flipped 0s (that is, where 1 in $\boldsymbol{x}$ and 0 in $\boldsymbol{x}^\alpha$ appear at the same index) and c_1 be the number of flipped 1s (that is, where 0 in $\boldsymbol{x}$ and 1 in $\boldsymbol{x}^\alpha$ appear at the same index). First, $s_\alpha = \boldsymbol{x}^\alpha \cdot \boldsymbol{x} = b - c_1$. Second, for $j \neq \alpha$, $s_j = \boldsymbol{x}^j \cdot \boldsymbol{x} \leq o + c_0$ (s_j reaches its maximum when all corrupted 1s are 0s and all corrupted 0s are 1s are at the same indices in $\boldsymbol{x}^j$). Third, because $c = c_0 + c_1$ and $c < b - o$, $o + c_0 = o + c - c_1 < o + b - o - c_1 = b - c_1$. Putting the three constraints together, we conclude that $s_\alpha > s_j$ and thus $y_\alpha = 1$ and $y_j = 0$ for all $j \neq \alpha$.

Next, we consider a much larger set of n-bit patterns, $X = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^M\}$ with $M \gg m$.

Definition 3. Let each example pattern $\boldsymbol{x}^\mu = (x_1^\mu, x_2^\mu, \cdots, x_n^\mu)$ be associated with a desired output $\boldsymbol{y}^\mu = (y_1^\mu, y_2^\mu, \cdots, y_n^\mu)$, with $y_j^\mu \in \{0, 1\}$ and $\sum_{j=1}^m y_j^\mu = 1$ (that is, only one specific $y_\alpha^\mu = 1$ and $y_j^\mu = 0$ for all $j \neq \alpha$). If $y_\alpha^\mu = 1$, then we say that $\boldsymbol{x}^\mu$ is a pattern in *class* α.

Let $\tilde{\boldsymbol{x}}^\alpha = (\tilde{x}_1^\alpha, \tilde{x}_2^\alpha, \cdots, \tilde{x}_n^\alpha) = \left(\sum_\mu x_1^\mu, \sum_\mu x_2^\mu, \cdots, \sum_\mu x_n^\mu\right)$ for all μ with $y_\alpha^\mu = 1$ (that is, the sum of all patterns in class α). Let $t_\alpha = \sum_i \tilde{x}_i^\alpha$ for the b largest components of $\tilde{\boldsymbol{x}}^\alpha$. Let $\bar{\boldsymbol{x}}^\alpha = (\bar{x}_1^\alpha, \bar{x}_2^\alpha, \cdots, \bar{x}_n^\alpha)$, with $\bar{x}_i^\alpha = \tilde{x}_i^\alpha / t_\alpha$ if $\tilde{x}_i^\alpha$ is one of the b largest values and $\bar{x}_i^\alpha = 0$ otherwise (that is, the averaged pattern for class α, restricted to the b most common bits and normalized to sum to 1). Let $\hat{\boldsymbol{x}}^\alpha = (\hat{x}_1^\alpha, \hat{x}_2^\alpha, \cdots, \hat{x}_n^\alpha)$, with $\hat{x}_i^\alpha = 1$ if $\bar{x}_i^\alpha > 0$ and $\hat{x}_i^\alpha = 0$ if $\bar{x}_i^\alpha = 0$. Let $\hat{X} = \{\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^m\}$ be the set of averaged patterns converted to binary.

The next two theorems are similar to Theorems 1 and 3, but generalized to using averaged training patterns as analogue weights rather than using a single training pattern (that is, target pattern) as binary weights.

Theorem 4. *If X is a set of M distinct n-bit patterns, $\hat{\boldsymbol{x}}^j$ contains exactly b 1s for all j and $\hat{\boldsymbol{x}}^j \neq \hat{\boldsymbol{x}}^k$ for all $j \neq k$, then the winner-take-all neural network with $W = (\boldsymbol{w}_1^\top, \boldsymbol{w}_2^\top, \ldots, \boldsymbol{w}_m^\top)$ and $\boldsymbol{w}_j = \bar{\boldsymbol{x}}^j$ remembers $\hat{X}$.*

Proof. Consider this network on input $\boldsymbol{x} = \hat{\boldsymbol{x}}^\alpha$. First, we calculate $s_\alpha = \bar{\boldsymbol{x}}^\alpha \cdot \hat{\boldsymbol{x}}^\alpha = \sum_{i=1}^n \bar{x}_i^\alpha = 1$. Second, for $j \neq \alpha$, $\hat{\boldsymbol{x}}^j \neq \hat{\boldsymbol{x}}^\alpha$. Because the number of 1s in both of these patterns is b, there exist at least one index i at which $\hat{x}_i^j = 1$ (and $\bar{x}_i^j > 0$) and $\hat{x}_i^\alpha = 0$; thus $s_j = \bar{\boldsymbol{x}}^j \cdot \hat{\boldsymbol{x}}^\alpha < \sum_{i=1}^n \bar{x}_i^j = 1$. Putting the two constraints together, we conclude that $s_\alpha > s_j$ and thus $y_\alpha = 1$ and $y_\alpha = 0$ for all $j \neq \alpha$.

Definition 4. In a winner-take-all neural network with $W = (\boldsymbol{w}_1^\top, \boldsymbol{w}_2^\top, \ldots, \boldsymbol{w}_m^\top)$ and $\boldsymbol{w}_j = \bar{\boldsymbol{x}}^j$, we say that each $\bar{\boldsymbol{x}}^j$ is a *memory* and each $\boldsymbol{x} = \hat{\boldsymbol{x}}^j$ is a *perfect* input. We say that a binary pattern $\boldsymbol{x}$ has c-bit *deviation* from a memory $\bar{\boldsymbol{x}}^\alpha$ if the number of indices at which the bits are different in $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}^\alpha$ is exactly c. We say that two memories $\bar{\boldsymbol{x}}^\alpha$ and $\bar{\boldsymbol{x}}^\beta$ have *overlap* $o = \max\{\bar{\boldsymbol{x}}^\alpha \cdot \hat{\boldsymbol{x}}^\beta, \bar{\boldsymbol{x}}^\beta \cdot \hat{\boldsymbol{x}}^\alpha\}$. We say a bit i is *no more than average* in $\bar{\boldsymbol{x}}^\alpha$ if $\bar{x}_i^\alpha \leq 1/b$, where b is the total number of 1s in $\hat{\boldsymbol{x}}^\alpha$.

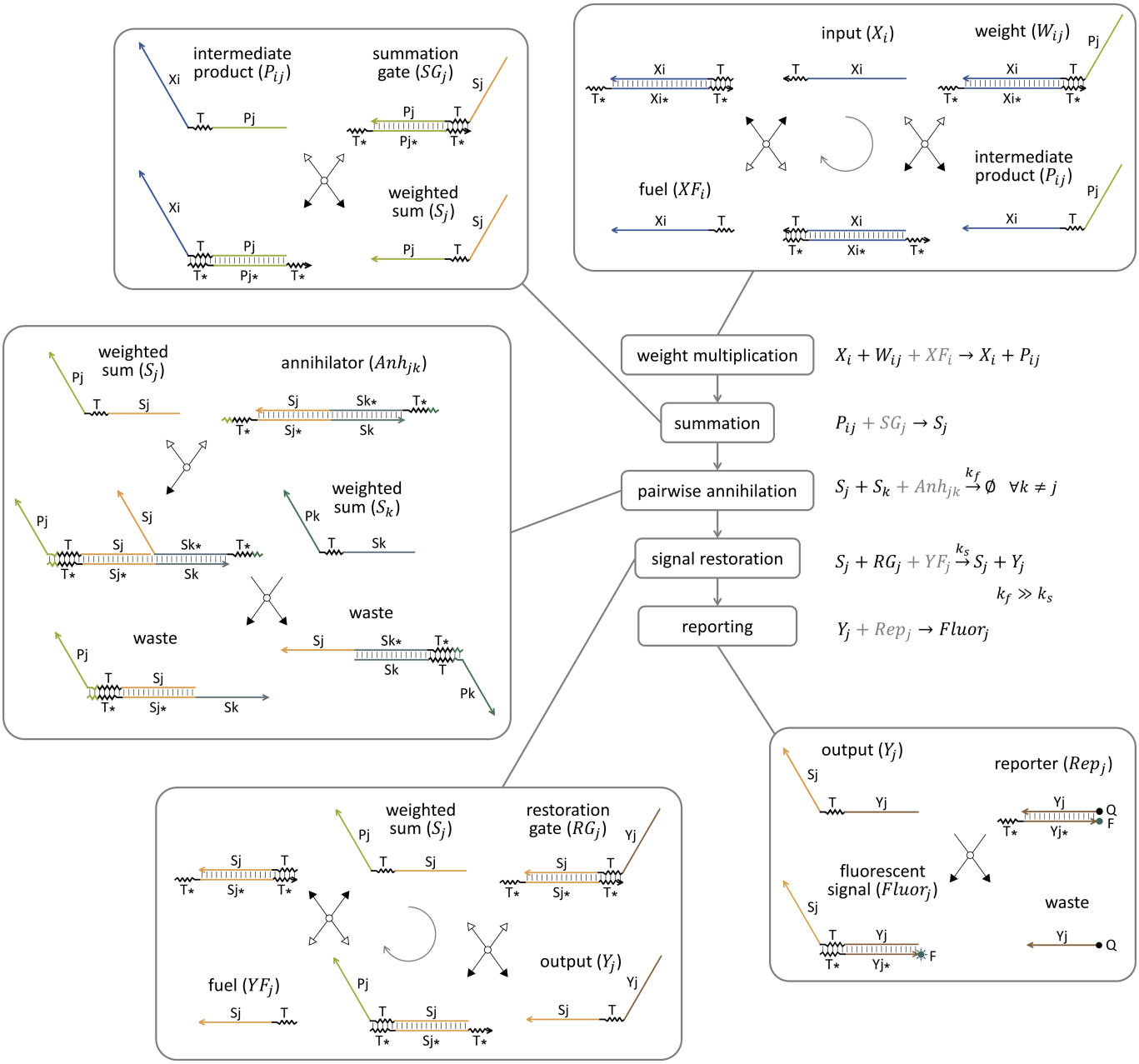
Theorem 5. *If $\boldsymbol{x}$ is a pattern with c-bit deviation from a memory $\bar{\boldsymbol{x}}^\alpha$, where $c < b(1 - o)$, b is the total number of 1s in $\hat{\boldsymbol{x}}^\alpha$ and o is the maximum overlap in $\bar{\boldsymbol{x}}^\alpha$ and $\bar{\boldsymbol{x}}^j$ for all $j \neq \alpha$, and if all flipped 1s are no more than average in $\bar{\boldsymbol{x}}^\alpha$ and all flipped 0s are no more than average in $\bar{\boldsymbol{x}}^j$ for all $j \neq \alpha$, then the winner-take-all neural network recognizes $\boldsymbol{x}$ as $\hat{\boldsymbol{x}}^\alpha$.*

Proof. Let c_0 be the number of flipped 0s (that is, where 1 in $\boldsymbol{x}$ and 0 in $\hat{\boldsymbol{x}}^\alpha$ appear at the same index) and c_1 be the number of flipped 1s (that is, where 0 in $\boldsymbol{x}$ and 1 in $\hat{\boldsymbol{x}}^\alpha$ appear at the same index). First, $s_\alpha = \bar{\boldsymbol{x}}^\alpha \cdot \boldsymbol{x} \geq 1 - c_1/b$. Second, for $j \neq \alpha$, $s_j = \bar{\boldsymbol{x}}^j \cdot \boldsymbol{x} \geq o + c_0/b$. Third, because $c = c_0 + c_1$ and $c < b(1 - o)$, $o + c_0/b = o + (c - c_1)/b < o + [b(1 - o) - c_1]/b = 1 - c_1/b$. Putting the three constraints together, we conclude that $s_\alpha > s_j$ and thus $y_\alpha = 1$ and $y_j = 0$ for all $j \neq \alpha$.

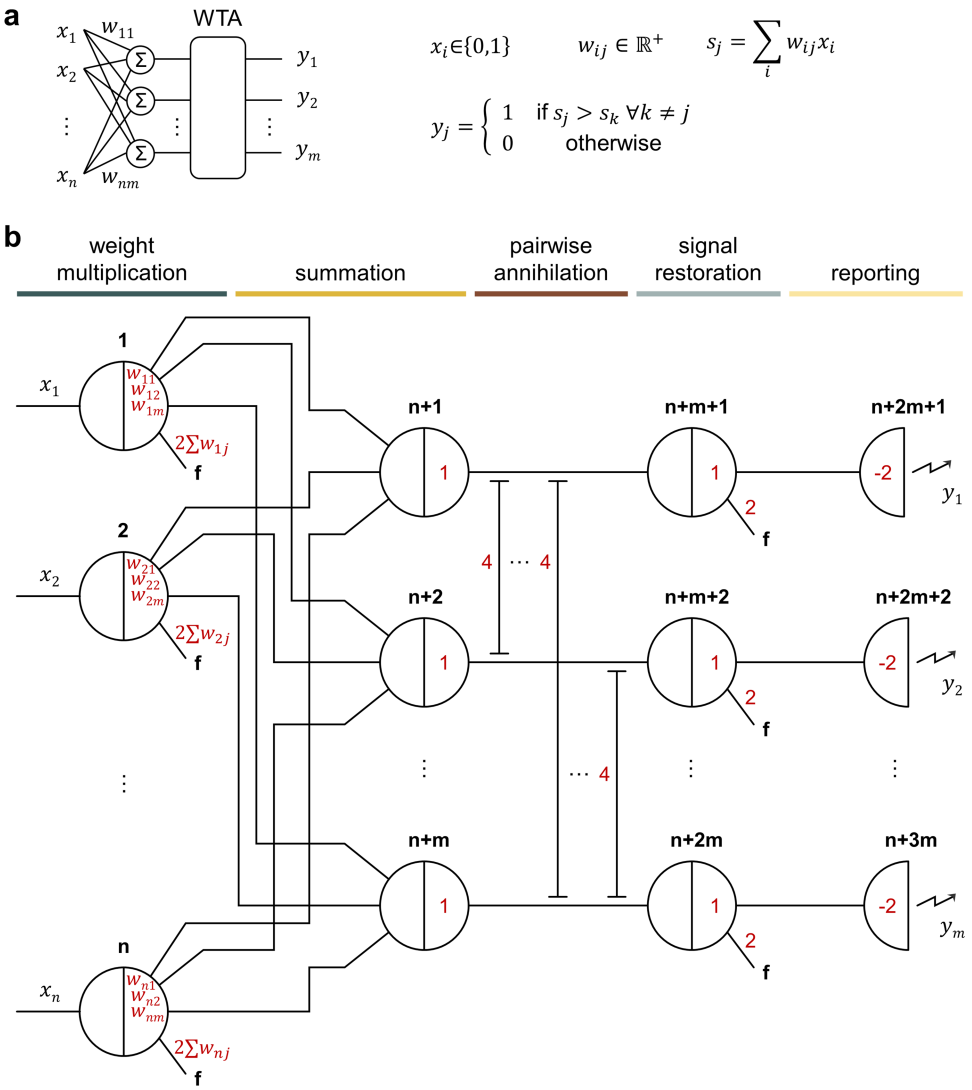
These are not the strongest results possible, but they provide intuition about how the winner-take-all neural network functions, with both binary and analogue weights, and how tolerant to errors it is.

Data availability. All data that support the findings of this study are included in the manuscript and its Extended Data. Source Data for Figs. 2–4 and Extended Data Figs. 3–7 are provided with the online version of the paper.

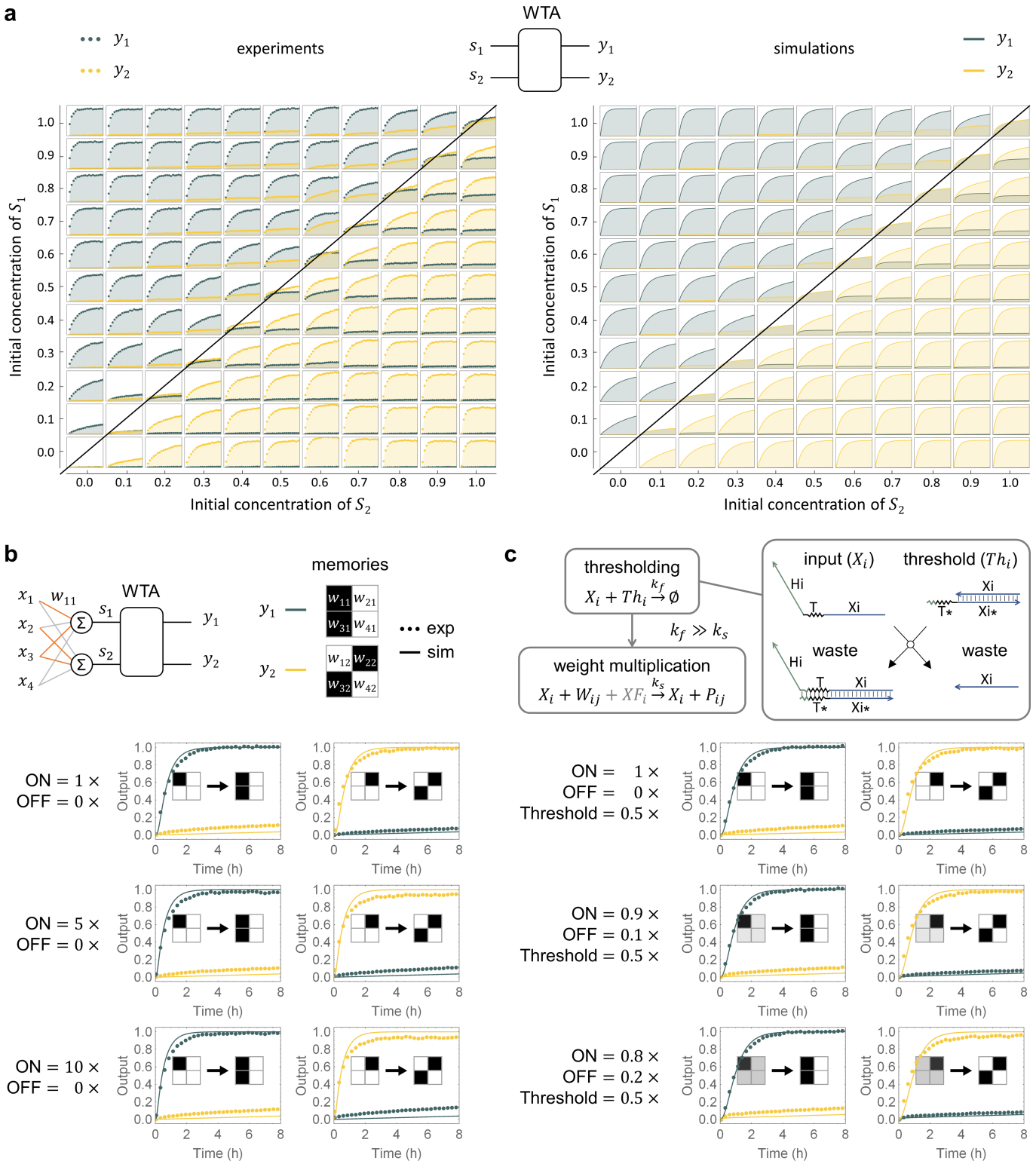
31. Zadeh, J. N. et al. NUPACK: analysis and design of nucleic acid systems. *J. Comput. Chem.* **32**, 170–173 (2011).



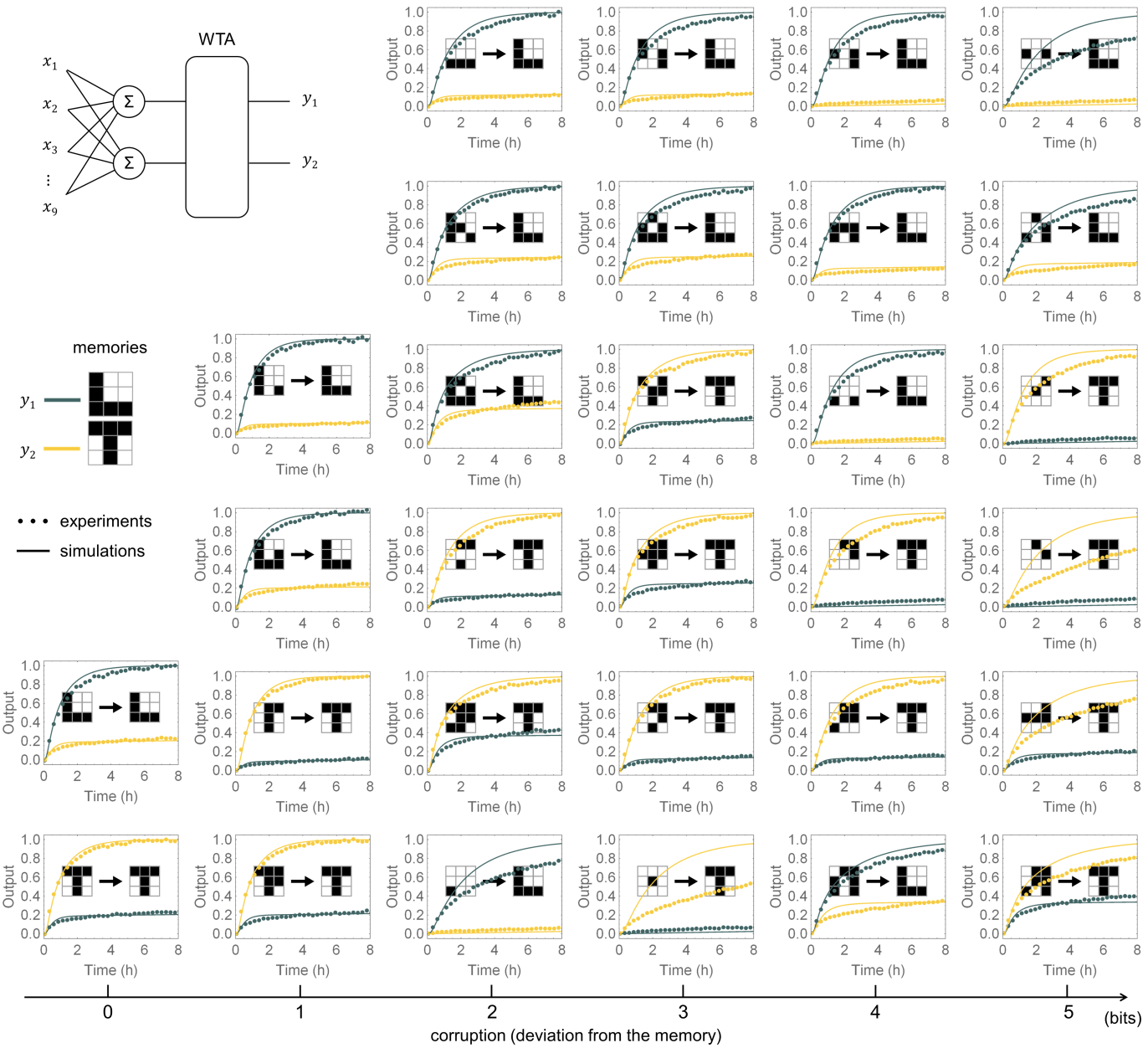
Extended Data Fig. 1 | DNA implementation of winner-take-all neural networks. The winner-take-all computation is broken into five subfunctions: weight multiplication, summation, pairwise annihilation, signal restoration and reporting. In the chemical reactions listed next to the five subfunctions, the species in black are needed as part of the function, the species in grey are needed to facilitate the reactions and the waste species are not shown. k_f and k_s are the rate constants of the pairwise-annihilation and signal-restoration reactions, respectively. In the DNA-strand-displacement implementation, weight multiplication and signal restoration are both catalytic reactions. The grey circle with an arrow indicates the direction of the catalytic cycle. Representative, but not all possible, states are shown for the pairwise-annihilation reaction. Zigzag lines indicate short (5 or 7 nucleotide) toehold domains and straight lines indicate long (15 or 20 nucleotide) branch-migration domains in DNA strands, with arrowheads marking their 3′ ends. Each domain is labelled with a name, and asterisks in the names indicate sequence complementarity. Black-filled and white-filled arrowheads indicate the forwards and backwards directions of a reaction step, respectively. All DNA sequences are listed in Supplementary Table 1.



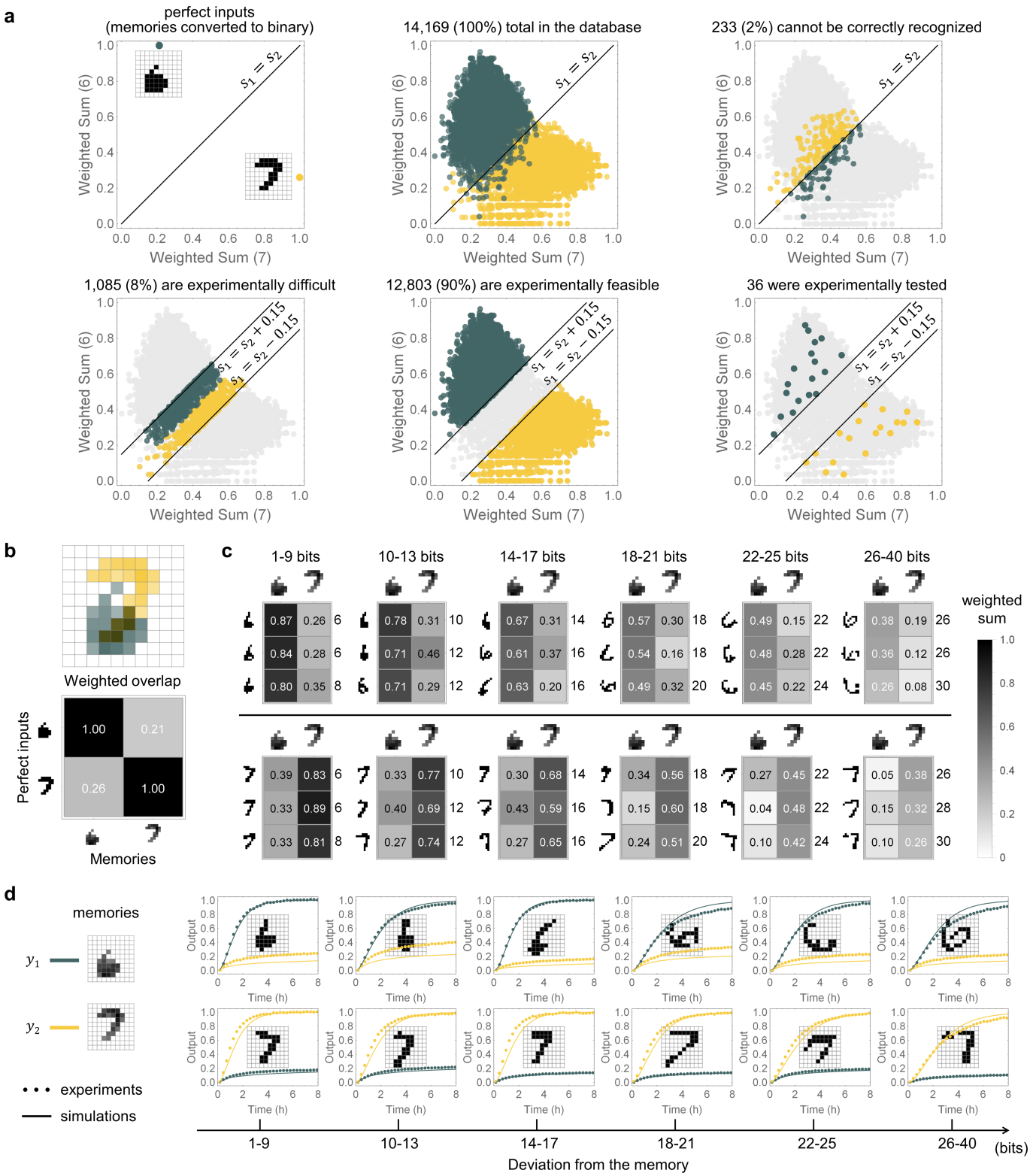
Extended Data Fig. 2 | Seesaw circuit implementation of winner-take-all neural networks. a, Same as Fig. 1a. **b**, Seesaw circuit diagram[11] for implementing the winner-take-all neural network. Each black number indicates the identity of a seesaw node. A total of $n + 3m$ nodes are required for implementing a winner-take-all neural network with m memories that each has n bits. The location and absolute value of each red number indicates the identity and relative initial concentration of a DNA species, respectively. A red number on a wire connected to a node (or between two nodes) indicates a free signal molecule, which can be an input or fuel strand. A red number inside a node indicates a gate molecule, which can be a weight, summation gate or restoration gate. A red number on a wire that stops perpendicularly at two wires indicates an annihilator molecule. A negative red number inside a half node with a zigzag arrow indicates a reporter molecule.



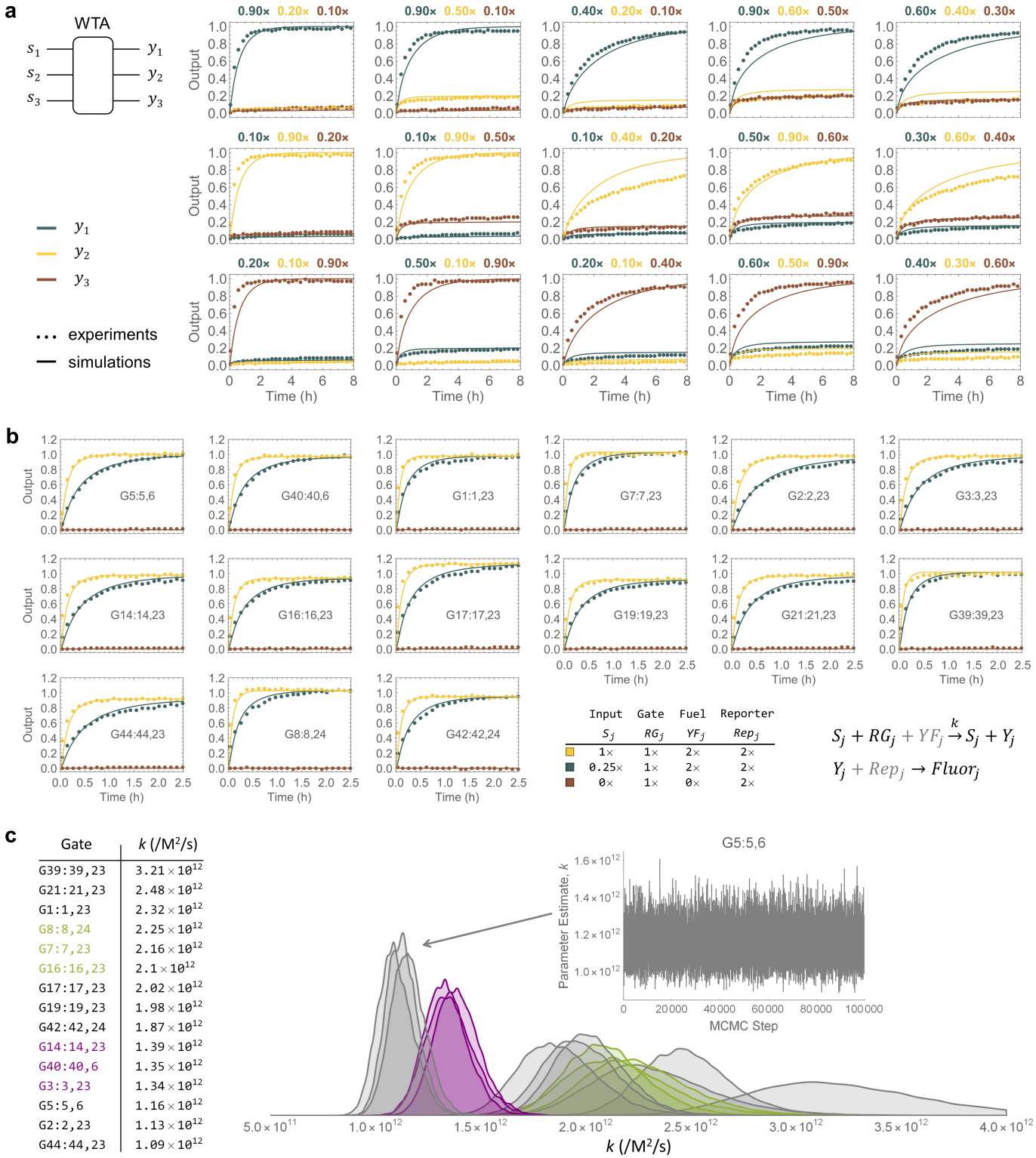
Extended Data Fig. 3 | Experimental characterization of winner-take-all DNA neural networks. a, Two-species winner-take-all behaviour. The experimental data (left, same as Fig. 2a) were used to identify the reverse rate constant $k_r = 0.4\ s^{-1}$ of the annihilation reaction in simulations (right). All fluorescence kinetics data and simulation are shown over the course of 2.5 h. The standard concentration is 50 nM (1×). Initial concentrations of the annihilator, restoration gates, fuels and reporters are 75 nM (1.5×), 50 nM (1×), 100 nM (2×) and 100 nM (2×), respectively. **b**, A 4-bit pattern recognition circuit with input concentration varying from 50 nM to 500 nM. In each output trajectory plot, dotted lines indicate fluorescence kinetics data and solid lines indicate simulation. The patterns to the left and right of the arrow indicate input signal and output classification, respectively. **c**, Applying thresholding to clean up noisy input signals. The thresholding mechanism has been reported previously in work on seesaw DNA circuits[11]. The extended toehold in threshold molecule has 7 nucleotides. In **b** and **c**, to compare the range of inputs, the concentration of each input strand is shown relative to 50 nM. The initial concentration of each weight molecule is either 0 or 50 nM; weight fuels are twice the concentration of weight molecules. The initial concentrations of the summation gates, annihilator, restoration gates, restoration fuels and reporters are 100 nM (1×), 400 nM (4×), 100 nM (1×), 200 nM (2×) and 200 nM (2×), respectively, with a standard concentration of 100 nM.



Extended Data Fig. 4 | A winner-take-all DNA neural network that recognizes 9-bit patterns as 'L' or 'T'. In each output trajectory plot, dotted lines indicate fluorescence kinetics data and solid lines indicate simulation. The standard concentration is 50 nM (1×). The initial concentration of each input strand is either 0 or 50 nM (1×). The initial concentration of each weight molecule is either 0 or 10 nM (0.2×); weight fuels are twice the concentration of weight molecules. The initial concentrations of the summation gates, annihilator, restoration gates, restoration fuels and reporters are 50 nM (1×), 75 nM (1.5×), 50 nM (1×), 100 nM (2×) and 100 nM (2×), respectively. The patterns to the left and right of the arrow indicate input signal and output classification, respectively. In addition to the perfect inputs, 28 example input patterns with 1–5 corrupted bits were tested. Note that 5 is the maximum number of corrupted bits, because an 'L' with more than 5-bit corruption will be as similar as or more similar to a 'T', and vice versa.

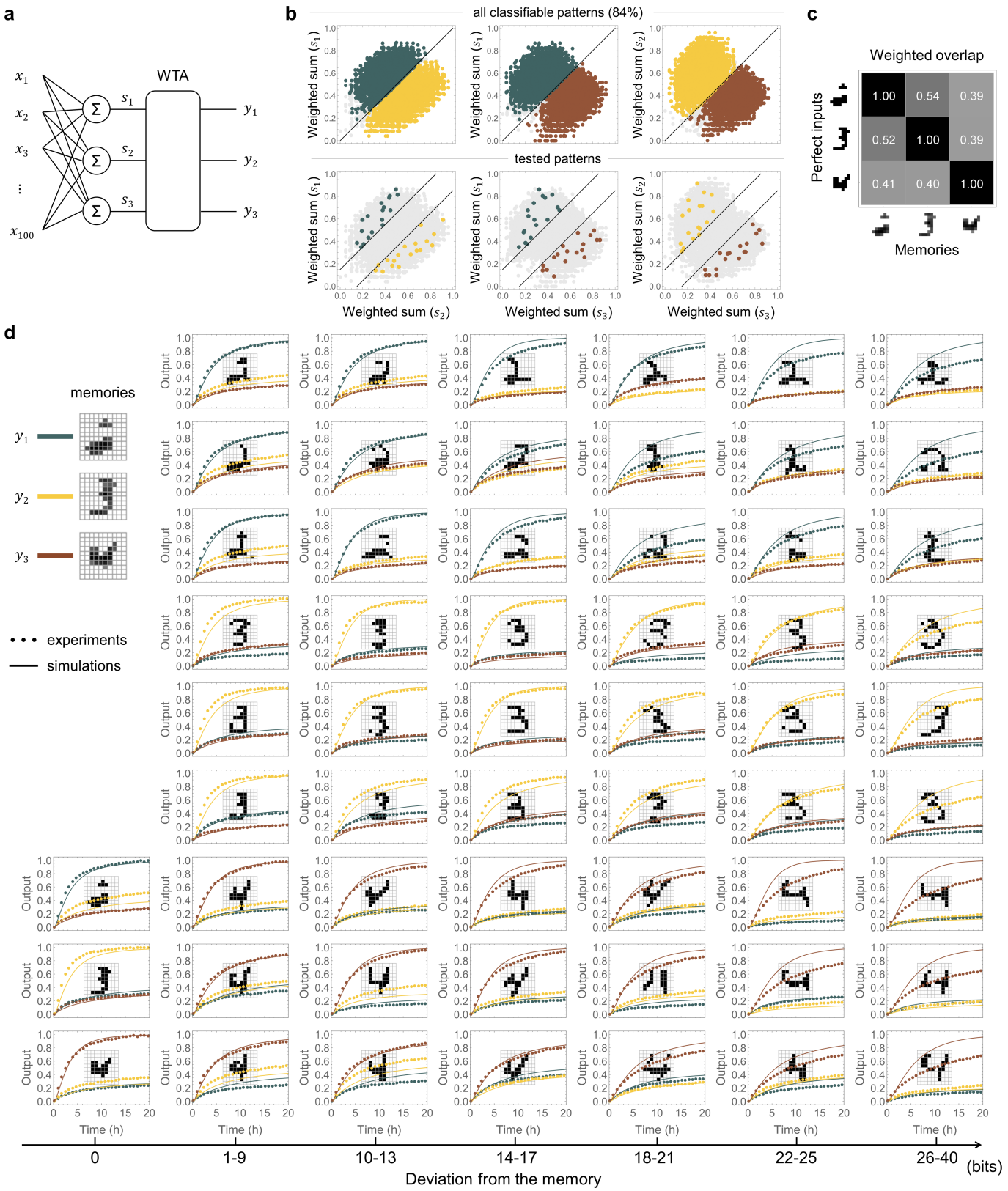


Extended Data Fig. 5 | A winner-take-all DNA neural network that recognizes 100-bit patterns as one of two handwritten digits. **a**, Choosing the test input patterns on the basis of their locations in the weighted-sum space. **b**, Overlap between the two memories: '6' and '7'. **c**, 36 test patterns with the number of flipped bits shown next to their weighted sums. **d**, Recognizing handwritten digits with up to 30 flipped bits compared to the perfect digits. Dotted lines indicate fluorescence kinetics data and solid lines indicate simulation. The standard concentration is 100 nM. Initial concentrations of all species are listed in Extended Data Fig. 10. The input pattern is shown in each plot. Note that 40 is the maximum number of flipped bits because all patterns have exactly 20 1s.

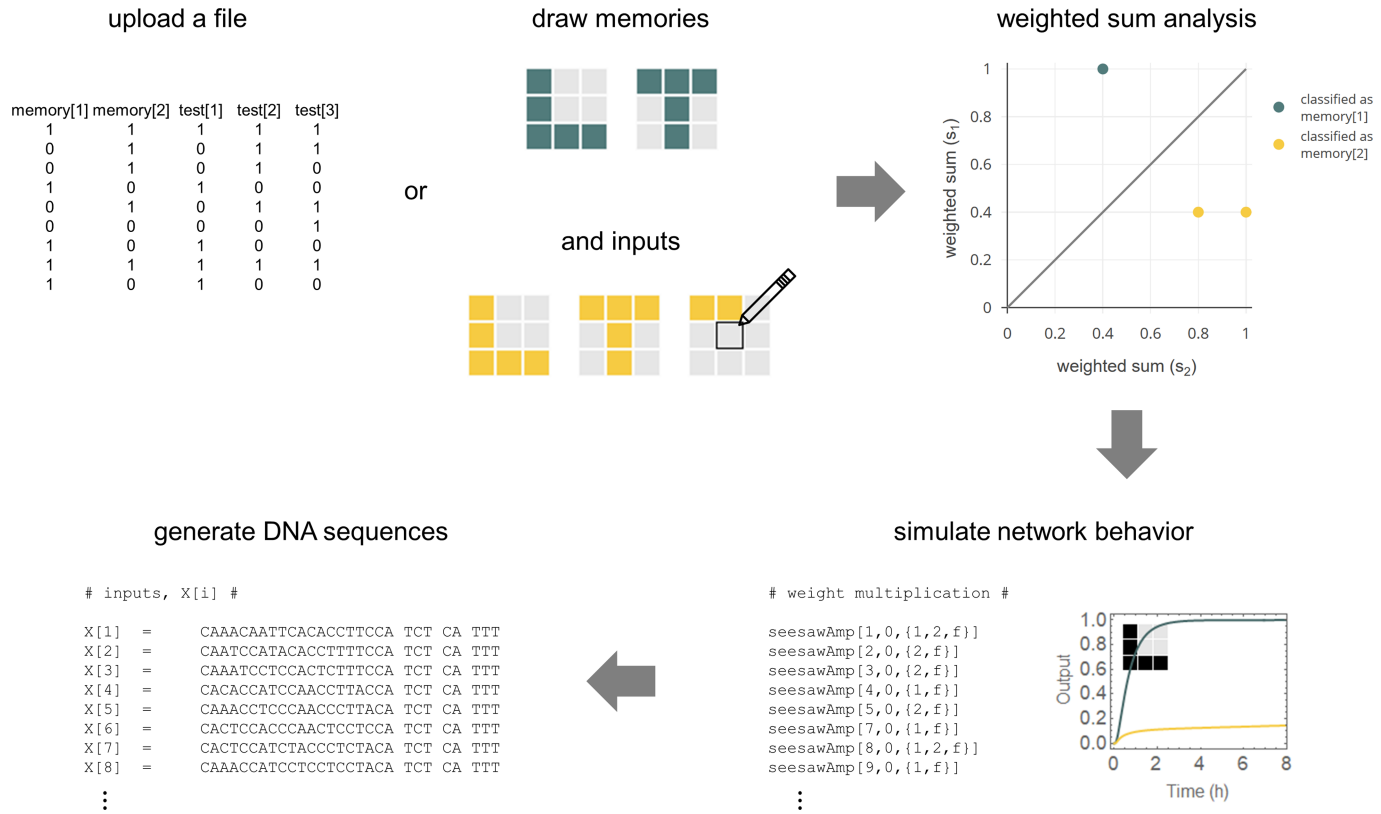


Gate	k (/M²/s)
G39:39,23	3.21×10^{12}
G21:21,23	2.48×10^{12}
G1:1,23	2.32×10^{12}
G8:8,24	2.25×10^{12}
G7:7,23	2.16×10^{12}
G16:16,23	2.1×10^{12}
G17:17,23	2.02×10^{12}
G19:19,23	1.98×10^{12}
G42:42,24	1.87×10^{12}
G14:14,23	1.39×10^{12}
G40:40,6	1.35×10^{12}
G3:3,23	1.34×10^{12}
G5:5,6	1.16×10^{12}
G2:2,23	1.13×10^{12}
G44:44,23	1.09×10^{12}

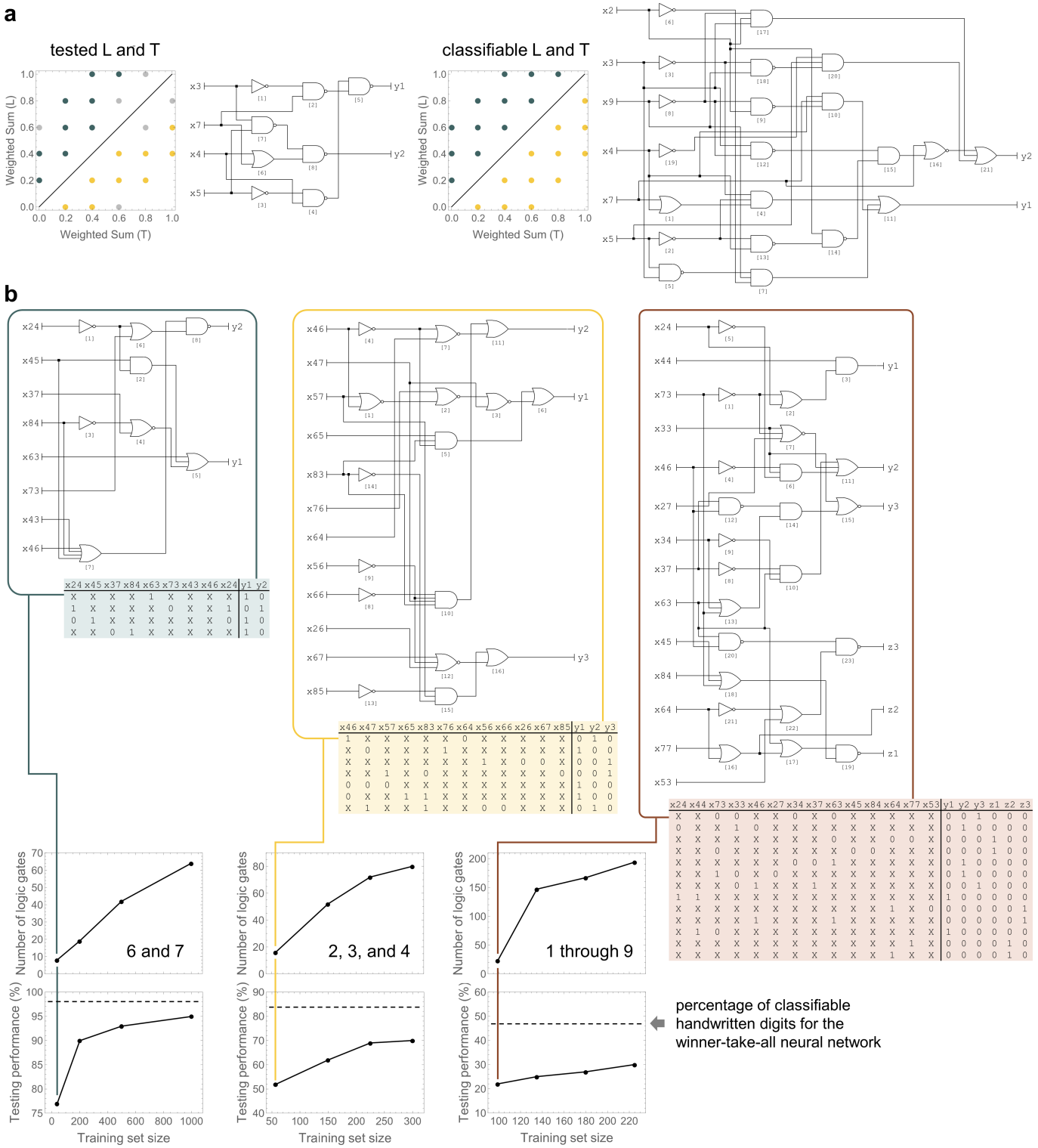
Extended Data Fig. 6 | Three-species winner-take-all behaviour and rate measurements for selecting DNA sequences in winner-take-all reaction pathways. a, Fluorescence kinetics data for a three-species winner-take-all circuit. Initial concentrations of the three weighted-sum species are shown on top of each plot as a number relative to a standard concentration of 50 nM (1×). The initial concentrations of the annihilator, restoration gates, fuels and reporters are 75 nM (1.5×), 50 nM (1×), 100 nM (2×) and 100 nM (2×), respectively. **b**, Measuring the rates of 15 catalytic gates. Fluorescence kinetics data (dotted lines) and simulations (solid lines) of the signal restoration reaction are shown, with a trimolecular rate constant (k) fitted using a Markov chain Monte Carlo package (https://github.com/joshburkart/mathematica-mcmc). The reporting reaction was needed for the fluorescence readout. Initial concentrations of all species are listed as a number relative to a standard concentration of 50 nM. **c**, The 15 catalytic gates sorted and grouped on the basis of their rate constants. All rate constants are within ±65% of the median. The two coloured groups of three rate constants are within ±5% of the median. These two groups of catalytic gates were selected for signal restoration in the winner-take-all DNA neural networks that remember two to nine 100-bit patterns (Methods section 'Sequence design').



Extended Data Fig. 7 | A winner-take-all DNA neural network that recognizes 100-bit patterns as one of three handwritten digits. a, Circuit diagram. **b**, Choosing the test input patterns on the basis of their locations in the weighted-sum space. **c**, Overlap between the three memories: '2', '3' and '4'. **d**, Recognizing handwritten digits with up to 28 flipped bits compared to the 'remembered' digits. Dotted lines indicate fluorescence kinetics data and solid lines indicate simulation. The standard concentration is 100 nM. Initial concentrations of all species are listed in Extended Data Fig. 10. The input pattern is shown in each plot. Note that 40 is the maximum number of flipped bits because all patterns have exactly 20 1s.



Extended Data Fig. 8 | Workflow of the winner-take-all compiler. The compiler[21] is a software tool for designing DNA-based winner-take-all neural networks. Users start by uploading a file that describes a winner-take-all neural network. Alternatively, the weight matrix and test patterns can be drawn graphically. Next, a plot of the weighted-sum space provides a visual representation of the classification decision boundaries. The kinetics of the system can be simulated using Mathematica code downloaded from the compiler website, and the set of reaction functions are displayed online. Finally, the compiler produces a list of DNA strands that are required to experimentally demonstrate the network as designed by the user.



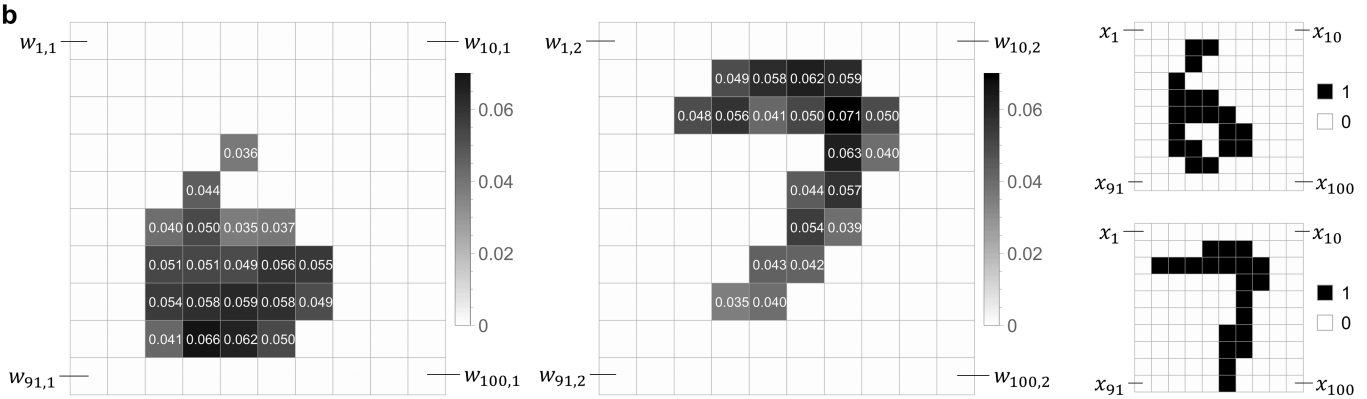
Extended Data Fig. 9 | Size and performance analysis of logic circuits for pattern recognition. a, Logic circuits that determine whether a 9-bit pattern is more similar to 'L' or 'T'. **b**, Logic circuits that recognize 100-bit handwritten digits. To find a logic circuit that produces correct outputs for a given set of inputs, with no constraint on other inputs, we first created a truth table including all experimentally tested inputs and their corresponding outputs. The outputs for all other inputs were specified as 'don't care', meaning the values could be 0 or 1. The truth table was converted to a Boolean expression and minimized in Mathematica, and then minimized again jointly for multiple outputs and mapped to a logic circuit in Logic Friday (https://download.cnet.com/Logic-Friday/3000-20415_4-75848245.html). In the minimized truth tables shown here, 'X' indicates a specific bit of the input on which the output does not depend. For comparison, minimized logic circuits were also generated from training sets with a varying total number of random examples from the MNIST database. The performance of each logic circuit, defined as the percentage of correctly classified inputs, was computed using all examples in the database. To make the minimization and mapping to logic gates computable in Logic Friday, the size of the input was restricted to the 16 most significant bits, determined on the basis of the weight matrix of the neural networks.

a

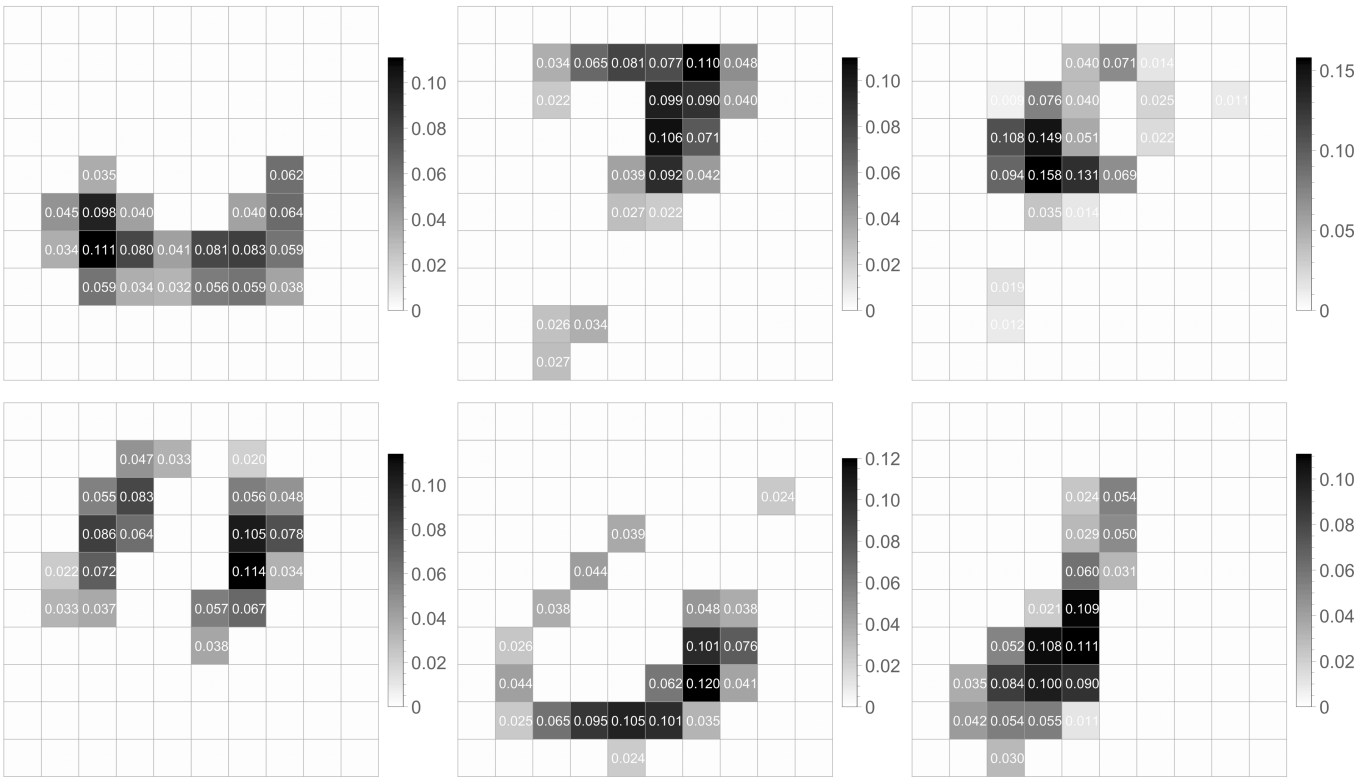
Name of species	Name of strands	Initial concentration
input (X_i)	`X[i]`	$x_i/20\times$
weight (W_{ij})	`P[i,j] + W[i]-b`	$w_{ij}\times$
fuel (XF_i)	`XF[i]`	$2\sum_{j=1}^{m} w_{ij}\times$
summation gate (SG_j)	`S[j] + SG[j]-b`	$1\times$
annihilator (Anh_{jk})	`Anh[j,k]-t + Anh[j,k]-b`	$4\times$
restoration gate (RG_j)	`Y[j] + RG[j]-b`	$1\times$
fuel (YF_j)	`YF[j]`	$2\times$
reporter (Rep_j)	`Rep[j]-t + Rep[j]-b`	$2\times$

$1\times = 100$ nM

b



c



Extended Data Fig. 10 | Species and their initial concentrations in all neural networks that recognize 100-bit patterns. a, List of species and strands. Reporters were annealed with top strands (that is, Rep[j]-t) in 20% excess. All other two-stranded complexes were annealed with a 1:1 ratio of the two strands and then PAGE-purified (Methods section 'Purification'). **b**, Weights and example inputs in the neural network that recognizes '6' and '7'. **c**, Weights in the neural network that recognizes '1'–'9'. Weights and inputs used in all experiments are listed in Supplementary Table 2. Detailed protocols for all experiments are listed in Supplementary Table 3.