Molecular Pattern Recognition and Supervised Learning in DNA-based Neural Networks

Thesis by
Kevin Matthew Cherry

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2024
Defended December 8, 2023

Kevin Matthew Cherry
ORCID: 0000-0002-2343-0754

ACKNOWLEDGEMENTS

Thank you to Lulu for believing in me at the very beginning.

I appreciate all the members of the Qian lab who created a supportive and intellectually stimulating environment.

Thank you to the roommates and friends who have been there through it all and remained steadfast during a global pandemic.

Most of all, thank you to my wonderful wife, Jeanne who has commiserated with me, encouraged me, and supported me every day through this journey. And to Julian, thank you for providing the final motivation I needed to complete this project and start the next chapter of my life. I love you both.

ABSTRACT

Adaptation in nature begins at the subcellular, molecular level with the delicate interplay of biomolecule cascades orchestrating the myriad function of cells. The intermingling activity of these cells becomes expressions of complex behavior of multi-cellular system. Nature provides a dazzling array of examples showcasing the variations of intelligent functions. However, in the realm of synthetic construction, what systems have humans managed to engineer, and what are the boundaries of our technological power? In comparison to nature's repertoire, mankind's accomplishments appear rather modest. The intricate behaviors observed in intelligent organisms emerge from the collective interactions and feedback loops among their constituent elements, resulting in the emergence of novel properties and phenomena. To develop large-scale engineered systems exhibiting ever more brain-like, intelligent behaviors, we must first devise new molecular architectures and algorithms designed for adaptation and learning at the molecular scale. My research presented here is a humble step toward those goals. I will present the design of novel molecular systems made from DNA that exhibit complex neural computation and learning behaviors.

Chapter 2 covers my contribution to scaling up the computing power of DNA circuits. From bacteria following simple chemical gradients to the brain distinguishing complex odor information, 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. Winner-take-all computation has been suggested as a potential strategy for enhancing the capability of DNA-based neural networks. Compared to the linear-threshold circuits and Hopfield networks used previously, winner-take-all circuits are computationally more powerful, allow simpler molecular implementation, and are not constrained by coupling 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 reactions. We use a previously developed seesaw DNA gate motif, extended to include a simple and robust component that facilitates the cooperative hybridization involved in 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.

Chapter 3 investigates the development of a computational neural network model inspired by biological learning mechanisms, particularly focusing on the new mechanisms for learning in a WTA neural network. The study incorporates novel molecular motifs used in inhibited activators and inhibited weights, designed specifically for training from environmental input patterns. These motifs emulate biological systems by facilitating memory storage and retrieval within DNA-based neural networks, similar to synaptic connections and signal processing observed in living organisms. We assess the function of the individual molecular motifs and characterize their specificity in up to 18-species cross-talk experiments. Furthermore, we characterize the network's performance across a wide array of training and test patterns, mirroring the adaptive responses and diverse conditions encountered by biological systems. Additionally, we analyze the computational efficiency and speed of the learning system, comparing it with both the previous non-learning DNA-based WTA model and a direct weight activation model. By exploring the principles of molecular learning, particularly within winner-take-all neural networks, this study aims to advance computational systems by emulating adaptability and resilience observed in biological organisms using robust, new molecular motifs.

PUBLISHED CONTENT AND CONTRIBUTIONS

Cherry, Kevin M. and Lulu Qian (2018). "Scaling up molecular pattern recognition with DNA-based winner-take-all neural networks". In: *Nature* 559.7714, pp. 370–376. DOI: 10.1038/s41586-018-0289-6.
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.

Cherry, Kevin M. (2017). *Winner-take-all DNA neural network compiler*. http://www.qianlab.caltech.edu/WTAcompiler/.
K.M.C. built the website, wrote the provided code, and developed the model.

Thubagere, Anupama J., Chris Thachuk, Joseph Berleant, Robert F. Johnson, Diana A. Ardelean, Kevin M. Cherry, and Lulu Qian (2017). "Compiler-aided systematic construction of large-scale DNA strand displacement circuits using unpurified components". In: *Nature Communications* 8, p. 14373. DOI: 10.1038/ncomms14373.
K.M.C. participated in collection of data.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

NOMENCLATURE

branch migration. a process in which one or two DNA helices exchange strands by the movement of a branch point or junction along a DNA molecule.

chemical reaction network. (CRN) a mathematical representation of a set of chemical reactions in a system describing how reactants interact and transform into products. It may also include kinetic rate parameters and initial species concentrations.

neural network. a computing system inspired by the brain, consisting of interconnected nodes or neurons popular for tasks like recognition and decision-making in the field of machine learning.

strand displacement. a molecular process in which one single-stranded DNA molecule displaces another from a double helix by forming more stable base pairs.

toehold. refers to a short, single-stranded DNA segment that acts as a specific binding site on a longer DNA strand.

winner-take-all. (WTA) a neural network mechanism where the neuron with the highest input value dominates and becomes the sole output, suppressing the others.

Chapter 1

INTRODUCTION

1.1 DNA computation

DNA is well known for encoding genetic information and as the heredity material in organisms. However, it can also be utilized beyond its biological role. Scientists can create synthetic DNA sequences and use them as a programmable material by exploiting the base-pairing specificity of DNA nucleotides. These synthesized sequences can self-assemble and exhibit programmable interactions through base pairing, enabling the construction of molecular circuits. Seminal work by Adleman (1994) sparked interest in computing with molecules when he reimagined a use for the large copy number of molecules as many parallel computing reactions. That insight led him to solving the popular and computationally difficult Hamiltonian path problem, known as the traveling salesman problem, using DNA molecules. Following that foundational work, concepts like branch migration, strand displacement, and toehold-mediated interactions (Yurke *et al.*, 2000; Turberfield *et al.*, 2003; Zhang and Winfree, 2009) facilitated exquisite control over DNA reactions. Molecules programmers then began building DNA-based logic gates (Saghatelian *et al.*, 2003; Okamoto *et al.*, 2004; Seelig, Yurke, *et al.*, 2005), expanding the possibilities for engineering DNA-based computational systems. These advancements enabled the design and implementation of early DNA-based molecular motors (Yurke *et al.*, 2000; Yin *et al.*, 2004; Shin and Pierce, 2004) and multi-layered molecular circuits (Seelig and Soloveichik, 2009; Qian and Winfree, 2011; Zhang and Seelig, 2011; Qian, Winfree, and Bruck, 2011), cementing DNA as more than solely an evolutionary molecule but serving as a programmable substrate for executing complex computational tasks.

1.2 Inspiration from biological learning and adaptation

Learning is a fundamental process observed across various biological scales. The brain's remarkable ability to restructure and reorganize itself, known as neuroplasticity, serves as the foundation of our understanding of learning, memory, and adaptability. This phenomenon encompasses the brain's capacity to form and strengthen neural connections and rewire pathways in response to experiences, learning, and environmental influences. At its essence, neuroplasticity embodies the brain's adapt-

ability, allowing for acquiring new skills and shaping behaviors, thoughts, and memories. The brain's dynamic and malleable nature underpins our capacity to learn, adjust to new situations, and navigate the complexities of the ever-changing world around us.

Microorganisms, like bacteria, also exhibit learning-like behaviors. While microorganisms lack a nervous system, they display adaptive responses to environmental stimuli, demonstrating a form of primitive learning through evolutionary changes and adaptive mechanisms. For instance, bacteria can respond to environmental stressors, including exposure to antibiotics or changes in environmental conditions. Bacteria become resistant to antibiotics over time by developing genetic mutations or acquiring resistance genes. This adaptation demonstrates a type of 'learning' where microorganisms respond and adjust to their surroundings, allowing them to survive and proliferate in changing environments. Furthermore, microorganisms can exhibit behaviors that indicate an ability to sense and react to environmental cues. Some single-celled microorganisms, like certain algae or bacteria, exhibit phototaxis or chemotaxis, moving towards or away from light or specific chemicals. While not traditional forms of learning, these responses illustrate a capacity for microorganisms to sense their environment and alter their behaviors in ways that enhance their survival, demonstrating a basic form of molecular adaptability.

The distributed cells of the immune system also display a mechanism of learning with remarkable parallels to machine learning. Immunological memory within the immune system is analogous to machine learning's concept of retaining information. This biological mechanism involves memory B and T cells functioning as repositories, storing precise details about confronted pathogens. Upon re-exposure to a known threat, these memory cells promptly initiate a targeted immune response, similar to how trained machine learning models apply past knowledge to respond effectively to familiar scenarios. Just as machine learning models improve predictions by learning from data, the immune system's memory cells enable a faster and more accurate response when confronted with previously encountered pathogens. The role of memory is vital in both systems, demonstrating their ability to retain and utilize past experiences to enhance future responses.

Biological adaptation, including neuroplasticity, microorganism evolution, and immunological memories, showcase life's ability to learn, adapt, and thrive. Inspired by natural biological processes, we have built an engineered system that leverages

the programmability of DNA to create artificial neural networks that can store and utilize information, adapting their responses to changing environments.

1.3 Neural network models of molecular information processing and learning

Artificial neural networks serve as a good mathematical model for biological learning due to their structure and function, which share parallels with the human brain's neural connections and learning mechanisms. These networks consist of interconnected nodes (neurons) that process information, similar to the neural pathways in the brain. Through an iterative process called "learning," neural networks edit their connections (weights) based on provided data, much like the brain adapts its neural connections through experience. Given the choice of computing models, neural networks hold greater interest compared to logic circuits primarily because of their training processes. Unlike traditional logic circuits that rely on fixed, predefined rules, neural networks can change their internal parameters, allowing them to recognize patterns, make predictions, and perform complex tasks without explicit programming. This flexible behavior and resemblance to natural learning systems make neural networks a desirable computational framework for building novel, engineered molecular learning networks. We have developed molecular neural networks that leverage DNA's programmability and self-assembly properties to create synthetic circuits capable of learning and adapting to environmental stimuli.

Although biology and machine learning have a shared inspiration in biological information processing, their development has been largely divergent. Some projects in the field of molecular computation have sought to integrate ideas from these disparate fields—creating novel engineered systems that perform advanced information processing. Qian *et al.* experimentally demonstrated a DNA-based Hopfield network (Qian, Winfree, and Bruck, 2011) that could recall four 4-bit patterns. Others have developed experimental (Pei *et al.*, 2010) and theoretical (Aubert *et al.*, 2013) molecular systems that can be programmed to play a game. Lakin *et al.* (2014) proposed a biomolecular circuit that could learn linear functions from labeled training data. Y.-J. Chen *et al.* (2013) built a consensus network that computes approximate majority using only DNA molecules. The output of the consensus network is similar to winner-take-all, but the molecular implementation is very different from our work. A mechanism for iterative updating of a biochemical network's weights was proposed by Baek *et al.* (2018). However, obtaining the weights required rounds of laborious hybridization, purification, and enzymatic amplification. More recently, Xiong *et al.* (2022) took inspiration from our work on scaling up DNA neural net-

works and devised a system that could implement convolutional neural networks by adding more orthogonal matrix multiplication operations to the input layer. This allowed them to classify test patterns with more bits into more target classes.

The works from these groups highlight the strong interest in and potential of DNA computation and molecular neural networks but also the many challenges. A molecular Hopfield network that stores memories with more than four bits would be very difficult to implement experimentally because it is a fully connected dual-rail network requiring many update steps. The other published works (Lakin *et al.*, 2014) demonstrate how difficult it is to embed learning into a molecular system. Even a small set of rules can require complex molecules, and basic learning algorithms like linear regression on a 2-bit vector need many chemical species and reactions, making them only possible in theory currently. The WTA network architecture is more suitable for the experimental implementation of large-scale pattern recognition. The simplicity of the learning algorithm we devised for the WTA network makes it an ideal candidate for experimentally building an artificial molecular neural network with learning capabilities.

1.4 WTA networks and their computational power

Winner-take-all (WTA) neural networks are a class of competitive neural models featuring a mechanism where neurons compete to become the most activated while inhibiting others. This network architecture facilitates the selection of the most relevant input stimuli, suppressing less pertinent information. WTA networks are fundamental in tasks such as pattern recognition and decision-making, enabling the focusing of attention on only the salient data. They offer valuable insights into the functioning of biological neural circuits and serve as a foundation for designing more sophisticated and intricate neural networks.

The WTA network model has one output bit for each input bit it receives. In hard-WTA, there is only one winner—the largest of the input bits. The output bit corresponding to this winner is set to 1, and the rest are set to 0. The network is capable of signal classification if a signal pattern is multiplied with a chosen weight matrix, and then the WTA layer is applied to the weighted sums. A neural network schematic of the function and the algorithm are shown in Fig. 3.1a.

WTA networks are not only inspiring because of their similarity to inhibitory biological networks in the brain, but a single WTA layer also has surprising power (Maass, 2000). For example, a k-WTA unit can compute any Boolean function, and a single

soft-WTA unit can approximate any continuous function. In fact, WTA circuits are more powerful than perceptrons (McCulloch and Pitts, 1943), also known as a linear threshold unit—one of the earliest and most common neural network algorithms. A function computed by a single n-input WTA gate requires a feed-forward circuit consisting of at least $O(n^2)$ linear threshold gates. A single k-WTA gate can compute the function of any two-layer feed-forward linear threshold circuit (Maass, 2000). Of further interest to molecular computing is that these WTA networks can accomplish this using strictly positive weights. Compared with DNA-based linear threshold circuits (Qian, Winfree, and Bruck, 2011), this eliminates the need for dual-rail representations of negative values and enables more sophisticated computation by a network of similar size. WTA networks are also more flexible than the previously demonstrated Hopfield network (Qian, Winfree, and Bruck, 2011). The number of members is not restricted to the number of bits in a pattern—networks with many small memories or several large memories can be constructed. Finally, WTA networks have the simplest training function—using the desired target patterns directly as the weight matrix. See Chapter 2 Section 2.9 for more information on the theoretical limits of WTA networks.

1.5 Summary of thesis

The work is presented here in two closely related chapters. Chapter 2 discusses our work establishing winner-take-all neural networks as a powerful model for DNA-based pattern classification. Chapter 3 builds directly on Chapter 2 and explains the newly added features of adaptable weights and supervised learning in WTA networks.

Chapter 2 covers my contribution to scaling up the computing power of DNA circuits. From bacteria following simple chemical gradients to the brain distinguishing complex odor information, 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 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 has been suggested as a potential strategy for enhancing the capability of DNA-based neural networks. Compared to the linear-threshold circuits and Hopfield networks used previously, winner-take-all circuits are computationally more powerful, 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 reactions. We use a previously developed seesaw DNA gate motif, extended to include a simple and robust component that facilitates the cooperative hybridization involved in 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.

Chapter 3 investigates the development of a computational neural network model inspired by biological learning mechanisms, particularly focusing on the new mechanisms for learning in a WTA neural network. The study incorporates novel molecular motifs used in inhibited activators and inhibited weights, designed specifically for training from environmental input patterns. These motifs emulate biological systems by facilitating memory storage and retrieval within DNA-based neural networks, conceptually similar to synaptic plasticity and signal processing observed in living organisms. We assess the function of the individual molecular motifs and characterize their specificity in up to 18-species cross-talk experiments. Furthermore, we characterize the network's performance across a wide array of training and test patterns, mirroring the adaptive responses and diverse conditions encountered by biological systems. Additionally, we analyze the computational efficiency and speed of the learning system, comparing it with both the previous non-learning DNA-based WTA model and a direct weight activation model. By exploring the principles of molecular learning, particularly within winner-take-all neural networks, this study aims to advance computational systems by emulating adaptability and resilience observed in biological organisms using robust, new molecular motifs.

Chapter 2

WINNER-TAKE-ALL NEURAL NETWORKS FOR PATTERN RECOGNITION

Cherry, Kevin M. and Lulu Qian (2018). "Scaling up molecular pattern recognition with DNA-based winner-take-all neural networks". In: *Nature* 559.7714, pp. 370–376.

2.1 WTA function and its chemical reaction network implementation

Winner-take-all computation (Maass, 2000) is one of the simplest competitive neural network models, inspired by the lateral inhibition and competition observed among biological neurons in the brain (Redgrave *et al.*, 1999). 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. 2.1a). Here, in a winner-take-all neural network, the weight matrix associated with each output is also referred to as a "memory." As shown in Fig. 2.1b, a simple training algorithm uses 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 as being most similar to one of the memories if the output associated with the memory is ON and all other outputs are OFF when the pattern is given as an input. 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 being ON and y_2 being OFF.

The winner-take-all function can be broken into five sub-functions, each of which can be implemented with a simple chemical reaction (Fig. 2.1c): First, weight multiplication of $x_i \times w_{ij}$ (x_i is a binary input and w_{ij} is an analog weight) is implemented with a reaction wherein an input species X_i catalytically converts a weight species W_{ij} to an intermediate product P_{ij}. If X_i is absent, no P_{ij} will be produced; if X_i is present, the final concentration of P_{ij} will be determined by the initial concentration of W_{ij}, thus setting the value of the weighted input. Next, summation is implemented with reactions that convert all intermediate species P_{ij} arriving at the same neuron to a common weighted sum species S_j. Comparing which weighted sum is larger than the others 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 is left. 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. Finally, reporting reactions are used to convert each output Y_j to a fluorescent signal $Fluor_j$.

All reactions except pairwise annihilation and signal restoration will naturally occur sequentially because the product of a previous reaction is a reactant of the next one. Since 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 slowly converted to an output signal.

2.2 DNA strand displacement implementation

Weight multiplication and signal restoration are both catalytic reactions that are implemented with a pair of seesawing reactions (Qian and Winfree, 2011) (Fig. 2.1e and Extended Data Fig. 2.5). 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 (Zhang and Winfree, 2009) 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. 2.5). 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 (Zhang, 2010) (Fig. 2.1f). 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, this process is completely reversible, and no molecules will be consumed. However, if another weighted sum strand S_k is also present, 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 both S_j and S_k strands simultaneously reach the middle point, the annihilator will be split apart into two waste molecules. Because neither waste molecule has a toehold exposed, they cannot interact with any other molecules. The annihilation reaction shown in Fig. 2.1f is designed to be roughly a hundred times faster than the

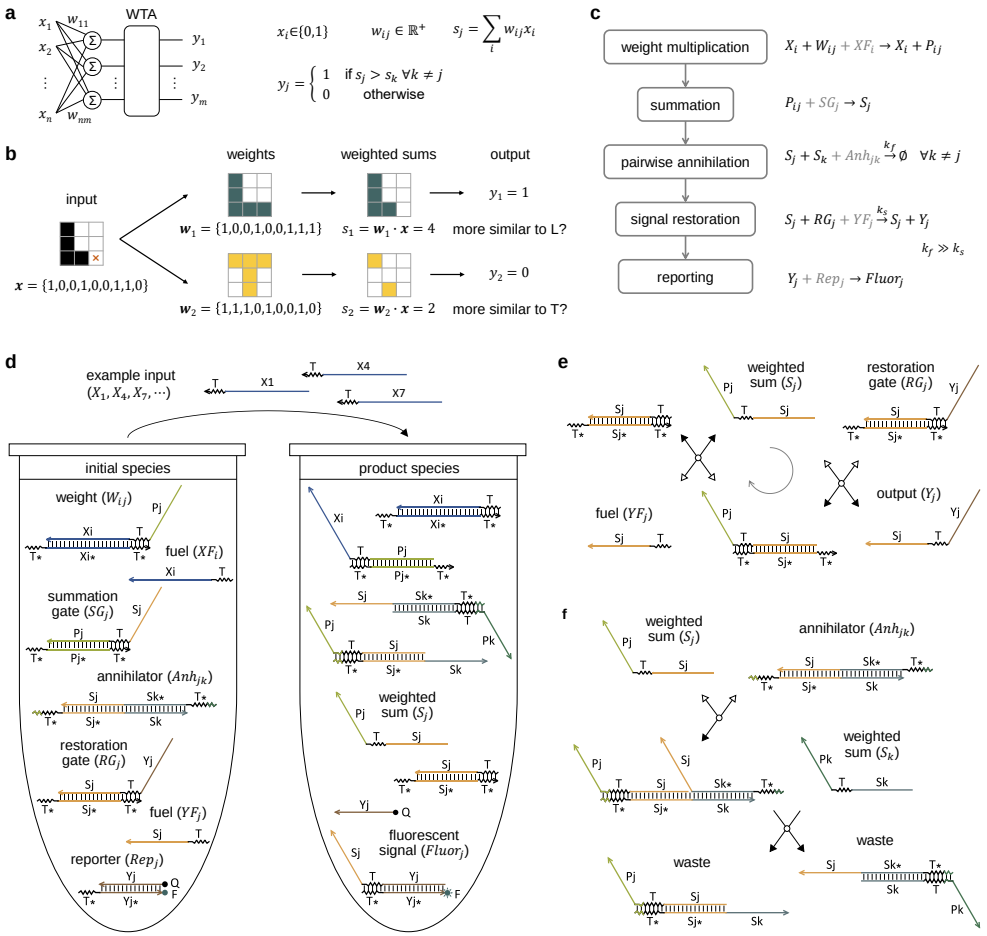


Figure 2.1 | Winner-take-all neural network and its DNA implementation. a, A winner-take-all (WTA) neural network with m memories that each has n bits. x_1 to x_n and y_1 to y_m are binary inputs and outputs, respectively. w_{ij} ($1 \le i \le n$ and $1 \le j \le m$) are analog weights of positive, real numbers. s_j ($1 \le j \le m$) are weighted sums of the inputs. **b,** Example pattern recognition using target patterns as weights. Each 9-bit pattern is shown in a 3 by 3 grid. Each black or colored pixel indicates a 1, and each white pixel indicates a 0. The two target patterns correspond to letter 'L' and 'T,' respectively. If the input pattern is corrupted (for example, the last bit of 'L' is flipped from 1 to 0, indicated by the orange '×'), 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, and the species in gray 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 $1 \le i \le n$ and $1 \le j \ne k \le m$. 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. **e,** Signal restoration reaction. The gray circle with an arrow indicates the direction of the catalytic cycle. **f,** 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 labeled with a name, and stars in the names indicate sequence complementarity. Black and white arrows indicate forward and backward directions of a reaction step, respectively. Representative but not all possible states are shown in (f). The mechanisms of weight multiplication, summation, and reporting reactions are shown in Extended Data Fig. 2.5.

signal restoration reaction shown in Fig. 2.1e because of 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 (Turberfield *et al.*, 2003; Zhang and Winfree, 2009).

Reporting is simply implemented with an irreversible strand displacement reaction, wherein an output strand Y_j interacts with a double-stranded reporter molecule Rep_j (Extended Data Fig. 2.5) to separate the fluorophore and quencher labeled strands in the reporter, resulting in increased fluorescence. Overall, the implementation of an arbitrary winner-take-all neural network can be systematically mapped to a seesaw DNA circuit (Extended Data Fig. 2.6).

2.3 Experimental characterization of the winner-take-all behavior

We started the experimental demonstration with a two-species winner-take-all function (Fig. 2.2a), which is similar to approximate majority (Cardelli and Csikász-Nagy, 2012) and consensus network (Y.-J. Chen *et al.*, 2013). If the initial concentration of one weighted sum species (S_1 or S_2) is higher than that of the other, we expect the corresponding output strand (Y_1 or Y_2) to be catalytically released and the fluorescent signal to reach an ideal ON state, while the other output signal to remain at an ideal OFF state. The data agreed with the expected overall circuit behavior, and also led 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. 2.2a. 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 where the difference 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 (i.e., 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 behavior, but 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. 2.2b). 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 correctly recognized. 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 behavior was better for the inputs with two-bit corruptions, compared with 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. 2.2b, 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 varying from the ideal high or low concentration (Extended Data Fig. 2.7).

2.4 Theoretical limits

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. First, how many distinct target patterns can be simultaneously remembered? Any set of patterns consisting of the same number of ones can be remembered (Methods, Theorem 1). For example, $\binom{9}{5} = 126$ patterns is the size of the largest set of 9-bit patterns that can be remembered, each consisting of five 1s. 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). Second, which corrupted patterns can be recognized? All patterns with less than $b - o$ corrupted bits can be recognized, where b is the total number of 1s and o is the maximum overlapped number of 1s in all target patterns (Methods, Theorem 3). For example, all patterns with less than 3 corrupted bits can be recognized for the 9-bit target patterns ‘L’ and ‘T’ shown in Fig. 2.1b, 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 3, 4, and 5 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 1 to 5, and demonstrated that the DNA

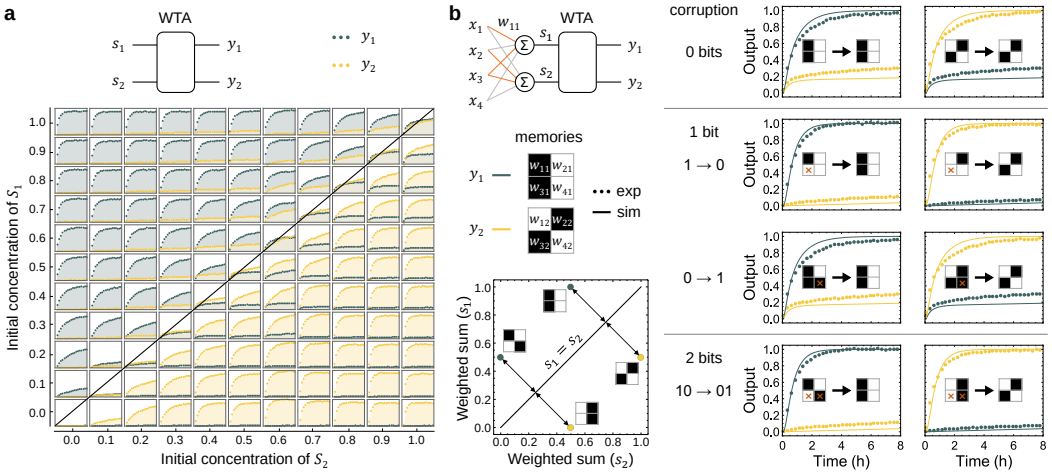


Figure 2.2 | Experimental characterization of winner-take-all DNA neural networks. a, Two-species winner-take-all behavior. Standard concentration 1× = 50 nM. The circuit is composed of two weighted sum strands (S_1 and S_2), an annihilator molecule ($Anh_{1,2}$ = 1.5×), two restoration gates (RG_1 and RG_2 = 1×), two fuel strands (YF_1 and YF_2 = 2×), and two reporters (Rep_1 and Rep_2 = 2×). Initial concentrations of S_1 and S_2 are shown as a number relative to the standard concentration. The diagonal line indicates equal concentrations of both strands. Fluorescence kinetics data are shown over the course of 2.5 hours, and normalized using a common minimum and maximum fluorescence level (Methods, Data normalization). To clearly illustrate the difference between the two output trajectories, the background below the data points are shown in the same color (with some transparency) as the data points. **b,** A 4-bit pattern recognition circuit. In the weighted sum layer of the circuit diagram, 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 total (4 inputs + 14 molecules in the weighted sum layer + 7 molecules in the winner-take-all layer). However, a circuit that remembers two specific 4-bit patterns only requires a subset of the wires in the weighted sum layer, each corresponding to a 1 in the memories (e.g., each orange wire in the circuit diagram corresponds to a black pixel in the memories). Thus, the example circuit is composed of 20 molecules total (4 fewer weight molecules and 1 fewer fuel strands). In each output trajectory plot, dotted lines indicate fluorescence kinetics data and solid lines indicate simulation. The pattern to the left and right of the arrow indicate input signals and output classification, respectively. Each orange '×' indicates a bit flip compared to the memories. Initial concentration of each input strand or weight molecule is either 0 or 50 nM. Initial concentrations of the annihilator, restoration gates, fuels, and reporters are 4×, 1×, 2×, and 2×, respectively, with 1× = 100 nM.

neural network correctly classified all examples (Extended Data Fig. 2.8). Finally, how does the size of the DNA circuit scale 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, $\binom{m}{2}$ annihilators, m gates and m fuel strands for signal restoration, and m reporters, totaling $n \times m + 2n + 4m + \binom{m}{2}$ 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.

2.5 Scalability with increasing pattern complexity

To experimentally demonstrate the scalability and power of winner-take-all DNA neural networks, we chose a task that is visually interesting: recognizing handwritten digits. Some aspects of this task are computationally nontrivial, for example, 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 (LeCun *et al.*, n.d.), which has been commonly used to test machine learning algorithms (Deng, 2012). We converted the original patterns to binary patterns with twenty 1s on a 10 by 10 grid, averaged a hundred example '6' and '7' patterns, and selected and normalized the top twenty pixels as weights (Fig. 2.3a and Methods, Neural network training and testing). The value of each analog weight was then implemented with the concentration of a weight molecule. The test inputs remained binary patterns, in which each 1 and 0 corresponded to the presence and absence of an input strand, respectively (Fig. 2.3b). The theoretical limits of the winner-take-all neural networks with analog 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. 2.3c).

In the MNIST database, there are a total of over 14,000 example handwritten '6' and '7' digits. Based on 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. 2.3d and Extended Data Fig. 2.9a): 2% of the patterns are on the wrong side of the diagonal line, which means 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, which we expect correct recognition. Thus, we

chose 36 representative example patterns from the last category, ensuring both uniform distribution in the weighted sum space and full range of bit deviation from the memories (Methods, Neural network training and testing). As shown in the experimental data (Fig. 2.3e and Extended Data Fig. 2.9d), the perfect patterns (which are the weights converted to binary) each yielded the desired circuit output. More importantly, patterns that are increasingly deviated 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—which is a desirable property of the winner-take-all computation.

2.6 Scalability with an increasing number of memories

We have now shown that the winner-take-all DNA neural networks scale well with more complex patterns. Next, we explore if 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, as the number of annihilators grows quadratically with the number of patterns. We showed that three-species winner-take-all function was still robust enough (Extended Data Fig. 2.10a) to allow the construction of a DNA neural network that remembers three 100-bit patterns, but the competition became harder with more competitors: the reaction rates for multiple annihilation pathways could be approximately but not perfectly matched (Methods, Sequence design and Extended Data Fig. 2.10bc), 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. 2.11). Using the same method, it would be difficult to construct networks that remember more patterns. Thus, we proposed an alternative approach that first divides the target patterns into groups and then uses multiple distinct group identities to classify the patterns (Fig. 2.4a). For example, the nine digits '1' through '9' can be divided into three groups in two ways, such that a pair of outputs uniquely corresponds to each digit (Fig. 2.4d). For example, a '4' is recognized if and only if $y_1 = 1$ and $z_1 = 1$. With this grouping approach, 9 distinct patterns can be recognized using only $\binom{\sqrt{9}}{2} \times 2 = 6$ annihilators, which would otherwise require $\binom{9}{2} = 36$ annihilators. In total, 225 distinct molecules were used for testing any specific input pattern out of 305 distinct molecules for all possible inputs (Fig. 2.4c).

We determined the weights for each group using a simple "average-then-subtract" method (Fig. 2.4b): take the average of a hundred examples per in-group digit,

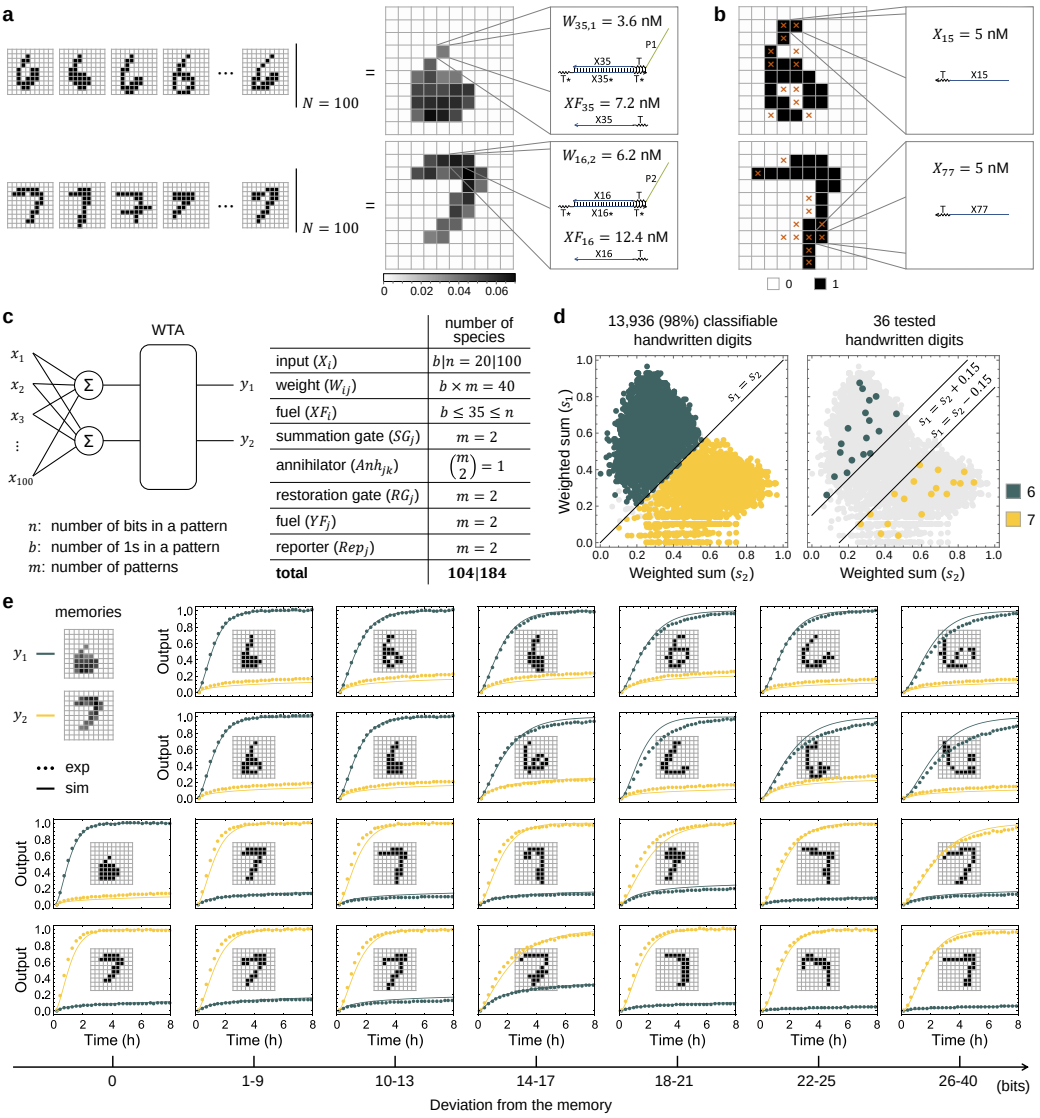


Figure 2.3 | A winner-take-all DNA neural network that recognizes 100-bit patterns as one of two handwritten digits. a, Weights determined as the average of a hundred '6's and '7's from the MNIST database. The value of each pixel (e.g., 0.036 for the 35th pixel in '6' and 0.062 for the 16th pixel in '7') was used to determine the concentration of each weight molecule, relative to a standard concentration of 1× = 100 nM (e.g., $W_{35,1}$ = 3.6 nM and $W_{16,2}$ = 6.2 nM). The fuel strands facilitating the weight multiplication reactions were twice the concentrations 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 present input strand was $1/b \times 100$ nM = 5 nM, where $b = 20$ is the total number of 1s in each input. The orange '×' indicates a bit flip compared to the memories (i.e., 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. **d,** Test input patterns chosen based on their locations in the weighted sum space. **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 simulation. The input pattern is shown on each plot. Note that 40 is the maximum number of flipped bits, because all patterns have exactly 20 1s. Initial concentrations of all species are listed in Extended Data Fig. 2.14.

subtract the average of a hundred examples per out-of-group digit, and select and normalize the top twenty pixels (Methods, 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 possibly 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 allows a larger set of distinct target patterns to be classified but with less accuracy. Nonetheless, as shown in the experimental data, the circuit yielded the desired pair of outputs for 99 representative example patterns (Fig. 2.4d and e).

To facilitate the design of winner-take-all DNA neural networks, we developed an online software tool. The WTA Compiler (Cherry, 2017) (Extended Data Fig. 2.12) converts a user-defined set of memories and test patterns into program code describing a DNA neural network, which can then be used to simulate the kinetics of the network behavior. It also provides sequences of the DNA strands required to experimentally construct the DNA neural network.

2.7 Discussion

It is interesting to compare the performance of winner-take-all neural networks with logic circuits. For example, it is possible to distinguish if 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 experimentally tested. However, a more complex circuit that consists of 21 logic gates is required to correctly compute the output for all classifiable patterns (Extended Data Fig. 2.13a). 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 experimentally tested are considered. But these logic circuits perform poorly when tested against the entire MNIST database (Extended Data Fig. 2.13b). 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, varying the concentrations of the weight molecules in the winner-take-all neural networks would allow the same set of DNA molecules to be used for different pattern classification tasks. In contrast, without reconfigurable circuit architectures, a different set of DNA molecules would be required for a logic circuit that performs a different task.

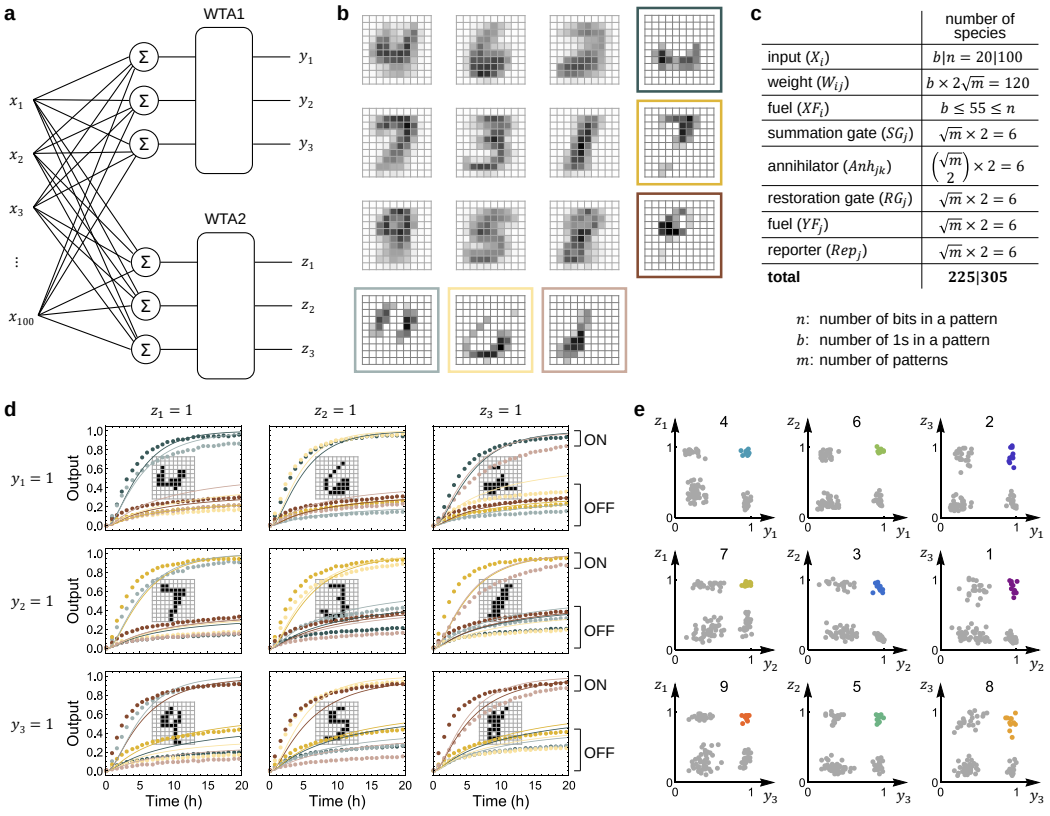


Figure 2.4 | A winner-take-all DNA neural network that recognizes 100-bit patterns as one of nine handwritten digits. a, Circuit diagram for recognizing nine distinct patterns using a grouping approach. **b,** Weights determined using an "average-then-subtract" method. Averaged digits are shown grouped by rows and columns. The weight matrix for each group is the average of all in-group digits subtracting the average of all out-of-group digits, highlighted with a distinct color that corresponds to the respective output trajectories shown in (d). **c,** Number of distinct species in the circuit. **d,** Fluorescence kinetics data (dotted lines) and simulation (solid lines) of the circuit behavior with nine representative examples of input patterns. The input is shown on each plot. **e,** Fluorescence level of each pair of outputs at 24 hours or longer after the inputs were added, collected from 99 experiments with 11 example patterns per digit. Each colored dot corresponds to one example pattern from the labeled class of digit, and each gray dot corresponds to an out-of-class example pattern. Initial concentrations of all species are listed in Extended Data Fig. 2.14.

The power of winner-take-all DNA neural networks could be further explored in several directions. Instead of the pairwise annihilation approach, a winner could be selected by competing resources (Kim *et al.*, 2004; Genot *et al.*, 2013), 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 (Maass, 2000). 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 with downstream logic circuits, could enable more sophisticated pattern recognition (for example, involving translated and rotated patterns) (Rojas, 2013). Using a variable-gain amplifier (Zhang and Seelig, 2010; S. X. Chen and Seelig, 2017), winner-take-all DNA circuits could be adapted to process analog 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 (Cho *et al.*, 2009; Li *et al.*, 2011), more diverse biomolecules could be detected.

Most excitingly, the fact that we were able to use target patterns as weights in winner-take-all DNA neural networks has opened 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 (Pei *et al.*, 2010) and theoretical (Fernando *et al.*, 2009; Aubert *et al.*, 2013; Lakin *et al.*, 2014) studies, learning—the most desirable property of biochemical circuits—would allow artificial molecular machines to adapt their functions based on environmental signals during autonomous operations.

2.8 Methods

Sequence design

All DNA strands used in the winner-take-all neural networks are composed of long branch migration domains and short toehold domains. Thanks to the modularity of the previously-developed seesaw DNA motif (Qian and Winfree, 2011; Qian, Winfree, and Bruck, 2011) 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 which have been previously experimentally validated (Thubagere *et al.*, 2017). 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 4 consecutive As or Ts or more than 3 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. And 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 mismatched branch migration domain will not complete strand displacement initiated from either the 3' or 5' end. In addition to a 15 nucleotide sequence pool used in previous work (Qian and Winfree, 2011; Qian, Winfree, and Bruck, 2011; Thubagere *et al.*, 2017), a 20 nucleotide sequence pool was generated and used in the weight multiplication layers because of the large number of molecules employed in this work. The two sequence pools were checked to ensure that the same pairwise criteria was met. All domains included the clamp design introduced in the previous work (Qian and Winfree, 2011) in order to reduce leak reactions between initial gate species.

All molecular complexes shared a 5 nucleotide universal toehold domain (Qian and Winfree, 2011; Qian, Winfree, and Bruck, 2011; Thubagere *et al.*, 2017). The annihilator complexes had 7 nucleotide toeholds composed of the 5 nucleotide universal toehold and a 2 nucleotide extension which 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 (i.e., same rates for all pairwise annihilation reactions), all annihilators within a set of winner-take-all computation 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. In this work, we used up to two sets of three annihilators. The extensions and dangle sequences were chosen by estimating the binding energies using NUPACK (Zadeh *et al.*, 2011), and the sequences for the second set of annihilators were chosen with similar energies to the first set that worked well in the three-species winner-take-all experiments (Extended Data Fig. 2.10a). Additionally, the rate of an annihilation reaction could also 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. 2.10bc). By using these gates for signal restoration, the branch migration

domains in the annihilators were simultaneously determined, as the signal restoration gates and annihilators share the same branch migration domains (Extended Data Fig. 2.5).

Neural network training and testing

The winner-take-all DNA neural network was tested on patterns derived from the classic MNIST handwritten digit database (LeCun *et al.*, n.d.). Both the training and testing sets were downloaded and merged into a single database, and all example patterns of digits '1' through '9' were retained, totaling 63,097 images. The original MNIST dataset consists of weight-centered gray-scale images on a 28 by 28 grid. In this work, we used binary patterns on a 10 by 10 grid. First, the images were rescaled to a 12 by 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-centered on a 10 by 10 grid, based on their bounding boxes.

A conscious effort was made to train the neural networks with 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 (i.e., 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 minus 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, for the purpose of shifting 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. In order 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 randomly chosen 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.

By exporting each sheet of the Excel file to a .csv file and uploading it to the WTA Compiler (Cherry, 2017), the weights and inputs can be visually displayed, the inputs analyzed in their weighted sum space, the kinetics behavior 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 TE, 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. Buffer for all experiments and annealed complexes was TE buffer with 12.5 mM Mg^{2+}. Complexes were annealed in a thermal cycler (Eppendorf) by heating to 90 °C for 5 minutes, and then cooling to 20 °C at the rate of 0.1 °C per 6 seconds.

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 hours at room temperature in TE buffer with 12.5 μM 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 (i.e., weight molecules) were annealed individually and then mixed together in the appropriate ratio, based on the values of the weights. This mixture was then purified via PAGE, recovered, and concentration determined by NanoDrop using the weighted-average extinction coefficient.

Fluorescence spectroscopy

Fluorescence kinetics data were collected every 2, 3, or 4 minutes, depending on the overall length of the experiment, using a microplate reader (Synergy H1, Biotek). Excitation and emission wavelengths were 496/525 nm for dye ATTO488, 555/582 nm for dye ATTO550, and 598/629 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 1× = 100 nM for all 4-bit and 100-bit pattern recognition and at 1× = 50 nM for all other experiments. Initial concentrations of all species are listed in Extended Data Fig. 2.14.

In the nine-digit experiments, six distinct output trajectories were read using three distinct fluorophores. Every experiment was ran twice, each having half of the outputs connected to fluorophore-labeled reporters and the other half to non-fluorophore-labeled 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 1× 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 1× was obtained from the average of the final five measurements from the highest signal produced from gate RG_j on a plate. 0× corresponds 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 a network's output for different input patterns. In the two-species winner-take-all experiments shown in Extended Data Fig. 2.7, the first 6 columns of data were measured on one plate and the last 5 columns measured on another. In the 9-bit pattern recognition experiments shown in Extended Data Fig. 2.8, the input patterns with 0 to 2 corrupted bits were measured on one plate and those with 3 to 5 corrupted bits measured on another. All other experimental data shown for the same neural network were measured on a single plate.

Model and simulations

Mass action simulation was performed using the same set of reactions and rate constants developed in the seesaw model (Qian and Winfree, 2011), with four additional reactions to model the pairwise annihilation:

$$S_j + Anh_{jk} \underset{k_r}{\overset{k_f}{\rightleftharpoons}} S_j : Anh_{jk}$$
$$S_k + Anh_{jk} \underset{k_r}{\overset{k_f}{\rightleftharpoons}} S_k : Anh_{jk}$$
$$S_j : Anh_{jk} + S_k \xrightarrow{k_f} \emptyset$$
$$S_k : Anh_{jk} + S_j \xrightarrow{k_f} \emptyset$$

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 (Qian and Winfree, 2011). The reverse rate constant $k_r = 0.4$ s^{-1} was determined using the experimental data shown in Extended Data Fig. 2.7a. This rate constant is on the same order as found in the previous study of cooperative hybridization (Zhang, 2010). 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 (Cherry, 2017).

2.9 Theoretical limits of the power of WTA neural networks

Let $\boldsymbol{X} = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^m\}$ be a set of m patterns, each having n bits. Let an example pattern from $\boldsymbol{X}$ be $\boldsymbol{x}^\alpha = [x_1^\alpha, x_2^\alpha, \cdots, x_n^\alpha]$ with $x_i^\alpha \in \{0, 1\}$. Let $s_j = \sum_{i=1}^n w_{ij} x_i$. The winner-take-all function shown in Fig. 2.1a can be rewritten as $y_j = 1$ if and only if $s_j > s_k$ for all $k \neq j$. We say that a winner-take-all neural network with weights $\boldsymbol{W}$ *remembers* $\boldsymbol{X}$ if for all α, $1 \leq \alpha \leq m$, $y_\alpha = 1$ (and $y_j = 0$ for all $j \neq \alpha$) when $\boldsymbol{x} = \boldsymbol{x}^\alpha$.

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

Proof. Consider this network on input $\boldsymbol{x} = \boldsymbol{x}^\alpha$. First, we calculate $s_\alpha = \sum_{i=1}^n x_i^\alpha x_i^\alpha = b$. Second, for $j \neq \alpha$, $\boldsymbol{x}^j \neq \boldsymbol{x}^\alpha$. Because the number of 1s in both these patterns

is b, the number of indices at which the bits are both 1 is strictly less than b. Thus $s_j = \sum_{i=1}^{n} x_i^j x_i^\alpha < b$. 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 $j \neq \alpha$. □

Theorem 2. *If $\boldsymbol{X}$ 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 $\boldsymbol{x}^\beta$ (i.e., 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 $\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_m]^\intercal$ and $\boldsymbol{w}_j = \boldsymbol{x}^j$ remembers $\boldsymbol{X}$.*

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

In a winner-take-all neural network with $\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_m]^\intercal$ and $\boldsymbol{w}_j = \boldsymbol{x}^j$, we 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 (i.e., 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 $\boldsymbol{x}$ is a pattern with $c < b - o$ corrupted bits compared to a memory $\boldsymbol{x}^\alpha$, where b is the total number of 1s in $\boldsymbol{x}^\alpha$ and o is the maximum overlapped bits in $\boldsymbol{x}^\alpha$ and $\boldsymbol{x}^j$ for all $j \neq \alpha$, then the winner-take-all neural network recognizes $\boldsymbol{x}$ as $\boldsymbol{x}^\alpha$.*

Proof. Let c_0 = the number of flipped 0s (i.e., 1 in $\boldsymbol{x}$ and 0 in $\boldsymbol{x}^\alpha$ at the same index) and c_1 = the number of flipped 1s (i.e., 0 in $\boldsymbol{x}$ and 1 in $\boldsymbol{x}^\alpha$ at the same index). First, $s_\alpha = \sum_{i=1}^{n} x_i^\alpha x_i = b - c_1$. Second, for $j \neq \alpha$, $s_j = \sum_{i=1}^{n} x_i^j x_i \leq o + c_0$ (s_j reaches its maximum when all corrupted 1s are 0s and all corrupted 0s are 1s 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 calculations 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, $\boldsymbol{X} = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \cdots, \boldsymbol{x}^M\}$, $M \gg m$. Let each example pattern $\boldsymbol{x}^\mu = [x_1^\mu, x_2^\mu, \cdots, x_n^\mu]$ be associated with a desired output $\boldsymbol{y}^\mu = [y_1^\mu, y_2^\mu, \cdots, y_m^\mu]$, $y_j^\mu \in \{0, 1\}$ and $\sum_{j=1}^{m} y_j^\mu = 1$ (i.e., only one

specific $y^\mu_\alpha = 1$, and $y^\mu_j = 0$ for all $j \neq \alpha$). If $y^\mu_\alpha = 1$, we say $\boldsymbol{x}^\mu$ is a pattern in <u>*class*</u> α.

Let $\tilde{\boldsymbol{x}}^\alpha = [\tilde{x}^\alpha_1, \tilde{x}^\alpha_2, \cdots, \tilde{x}^\alpha_n] = [\sum_\mu x^\mu_1, \sum_\mu x^\mu_2, \cdots, \sum_\mu x^\mu_n]$ for all μ with $y^\mu_\alpha = 1$ (i.e., the sum of all patterns in class α). Let $t_\alpha = \sum_i \tilde{x}^\alpha_i$ for the b largest values of $\tilde{x}^\alpha_i$. Let $\overline{\boldsymbol{x}}^\alpha = [\overline{x}^\alpha_1, \overline{x}^\alpha_2, \cdots, \overline{x}^\alpha_n]$ with $\overline{x}^\alpha_i = \frac{1}{t_\alpha} \times \tilde{x}^\alpha_i$ if $\tilde{x}^\alpha_i$ is among the b largest values and otherwise $\overline{x}^\alpha_i = 0$ (i.e., the averaged pattern for class α, restricted to the b most-common bits and normalized to sum to 1). Let $\hat{\boldsymbol{x}}^\alpha = [\hat{x}^\alpha_1, \hat{x}^\alpha_2, \cdots, \hat{x}^\alpha_n]$ with $\hat{x}^\alpha_i = 1$ if $\overline{x}^\alpha_i > 0$ and $\hat{x}^\alpha_i = 0$ if $\overline{x}^\alpha_i = 0$. Let $\hat{\boldsymbol{X}} = \{\hat{\boldsymbol{x}}^1, \hat{\boldsymbol{x}}^2, \cdots, \hat{\boldsymbol{x}}^m\}$ be the set of averaged patterns converted to binary.

Theorem 4. *If $\boldsymbol{X}$ 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 $\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_m]^\intercal$ and $\boldsymbol{w}_j = \overline{\boldsymbol{x}}^j$ remembers $\hat{\boldsymbol{X}}$.*

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

In a winner-take-all neural network with $\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_m]^\intercal$ and $\boldsymbol{w}_j = \overline{\boldsymbol{x}}^j$, we say each $\overline{\boldsymbol{x}}^j$ is a <u>*memory*</u> and each $\boldsymbol{x} = \hat{\boldsymbol{x}}^j$ is a <u>*perfect*</u> input. We say that a binary pattern $\boldsymbol{x}$ has c-bit <u>*deviation*</u> from a memory $\overline{\boldsymbol{x}}^\alpha$ 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 $\overline{\boldsymbol{x}}^\alpha$ and $\overline{\boldsymbol{x}}^\beta$ have <u>*overlap*</u> $o = \max\{\overline{\boldsymbol{x}}^\alpha \cdot \hat{\boldsymbol{x}}^\beta, \overline{\boldsymbol{x}}^\beta \cdot \hat{\boldsymbol{x}}^\alpha\}$. We say a bit i is <u>*no more than average*</u> in $\overline{\boldsymbol{x}}^\alpha$ if $\overline{x}^\alpha_i \leq \frac{1}{b}$, where b is the total number of 1s in $\hat{\boldsymbol{x}}^\alpha$.

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

Proof. Let c_0 = the number of flipped 0s (i.e., 1 in $\boldsymbol{x}$ and 0 in $\hat{\boldsymbol{x}}^\alpha$ at the same index) and c_1 = the number of flipped 1s (i.e., 0 in $\boldsymbol{x}$ and 1 in $\hat{\boldsymbol{x}}^\alpha$ at the same index). First, $s_\alpha = \sum_{i=1}^n \overline{x}^\alpha_i x_i \geq 1 - \frac{c_1}{b}$. Second, for $j \neq \alpha$, $s_j = \sum_{i=1}^n \overline{x}^j_i x_i \leq o + \frac{c_0}{b}$. Third, because $c = c_0 + c_1$ and $c < b \times (1 - o)$, $o + \frac{c_0}{b} = o + \frac{c - c_1}{b} < o + \frac{b \times (1-o) - c_1}{b} = 1 - \frac{c_1}{b}$. Putting

the three calculations together, we conclude that $s_\alpha > s_j$ and thus $y_\alpha = 1$ and $y_j = 0$ for all $j \neq \alpha$. □

Note that these are not the strongest results possible, but they give some intuition about how the winner-take-all neural network functions, with both binary and analog weights, and how tolerant to errors it is.

2.10 Extended data figures

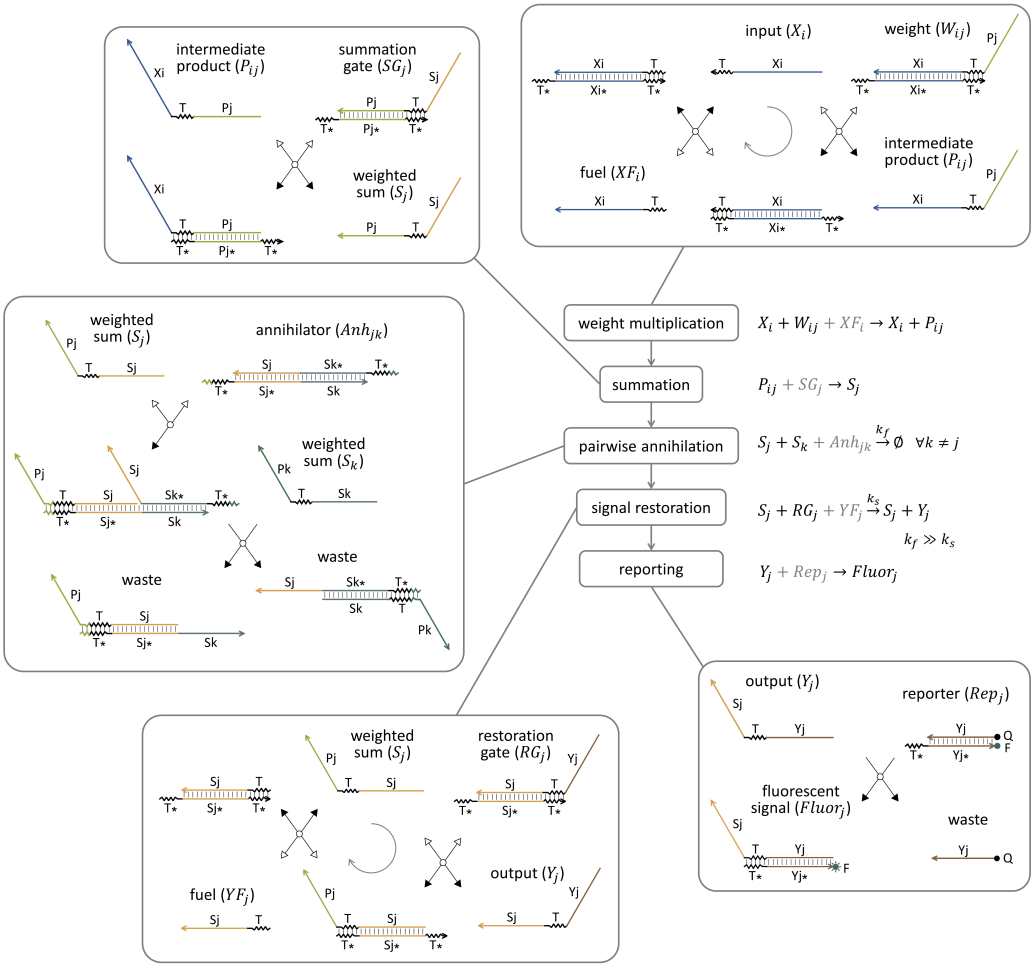


Figure 2.5 | DNA implementation of winner-take-all neural networks. The winner-take-all computation is broken into five sub-functions: weight multiplication, summation, pairwise annihilation, signal restoration, and reporting. In the chemical reactions listed next to the five sub-functions, the species in black are needed as part of the function, the species in gray 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, both weight multiplication and signal restoration are catalytic reactions. The gray 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 labeled with a name, and stars in the names indicate sequence complementarity. Black and white arrows indicate forward and backward directions of a reaction step, respectively.

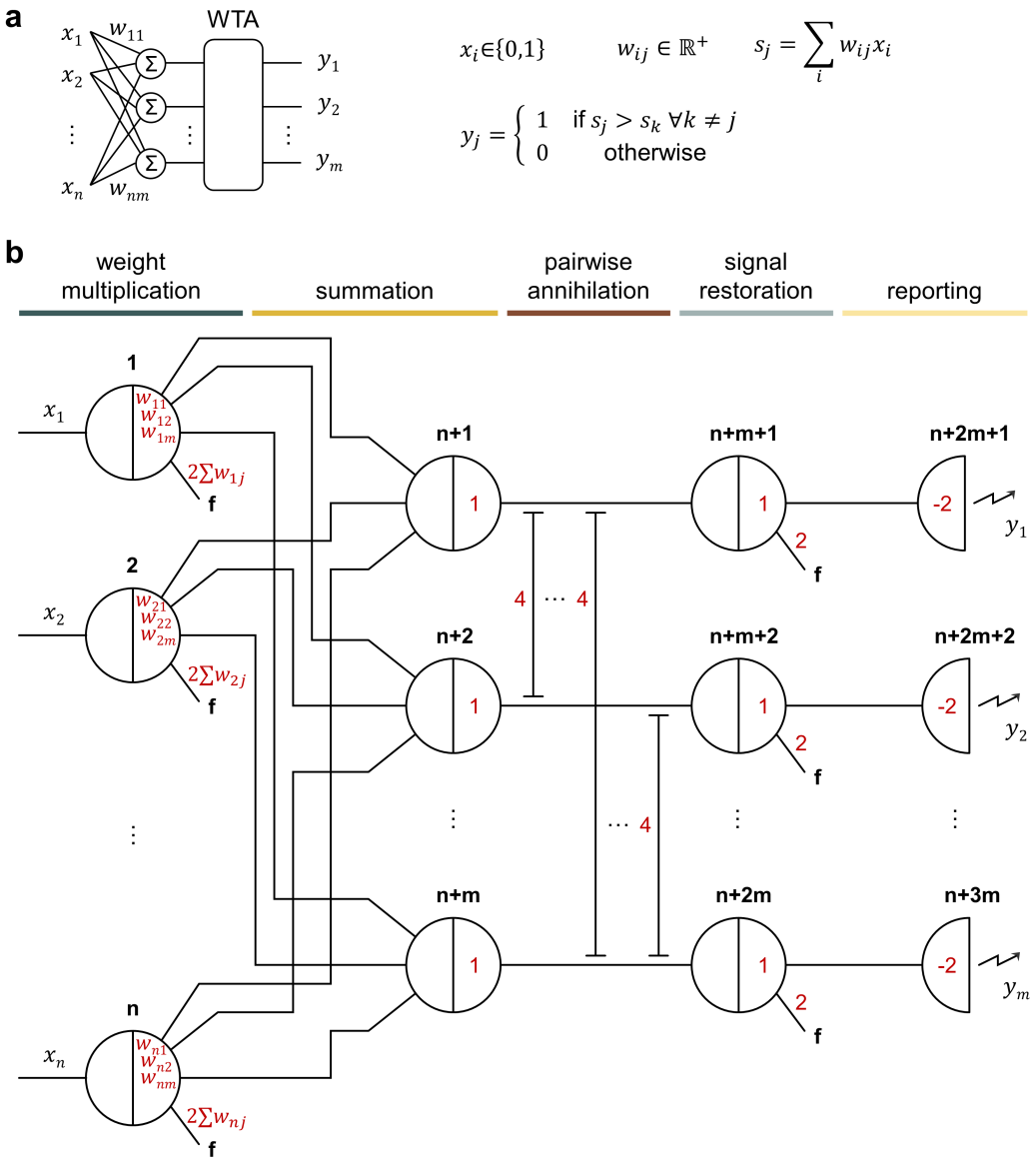


Figure 2.6 | Seesaw circuit implementation of winner-take-all neural networks. a, A winner-take-all neural network with m memories that each has n bits. x_1 to x_n and y_1 to y_m are binary inputs and outputs, respectively. w_{ij} ($1 \leq i \leq n$ and $1 \leq j \leq m$) are analog weights of positive, real numbers. s_j ($1 \leq j \leq m$) are weighted sums of the inputs. **b,** Seesaw circuit diagram 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 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.

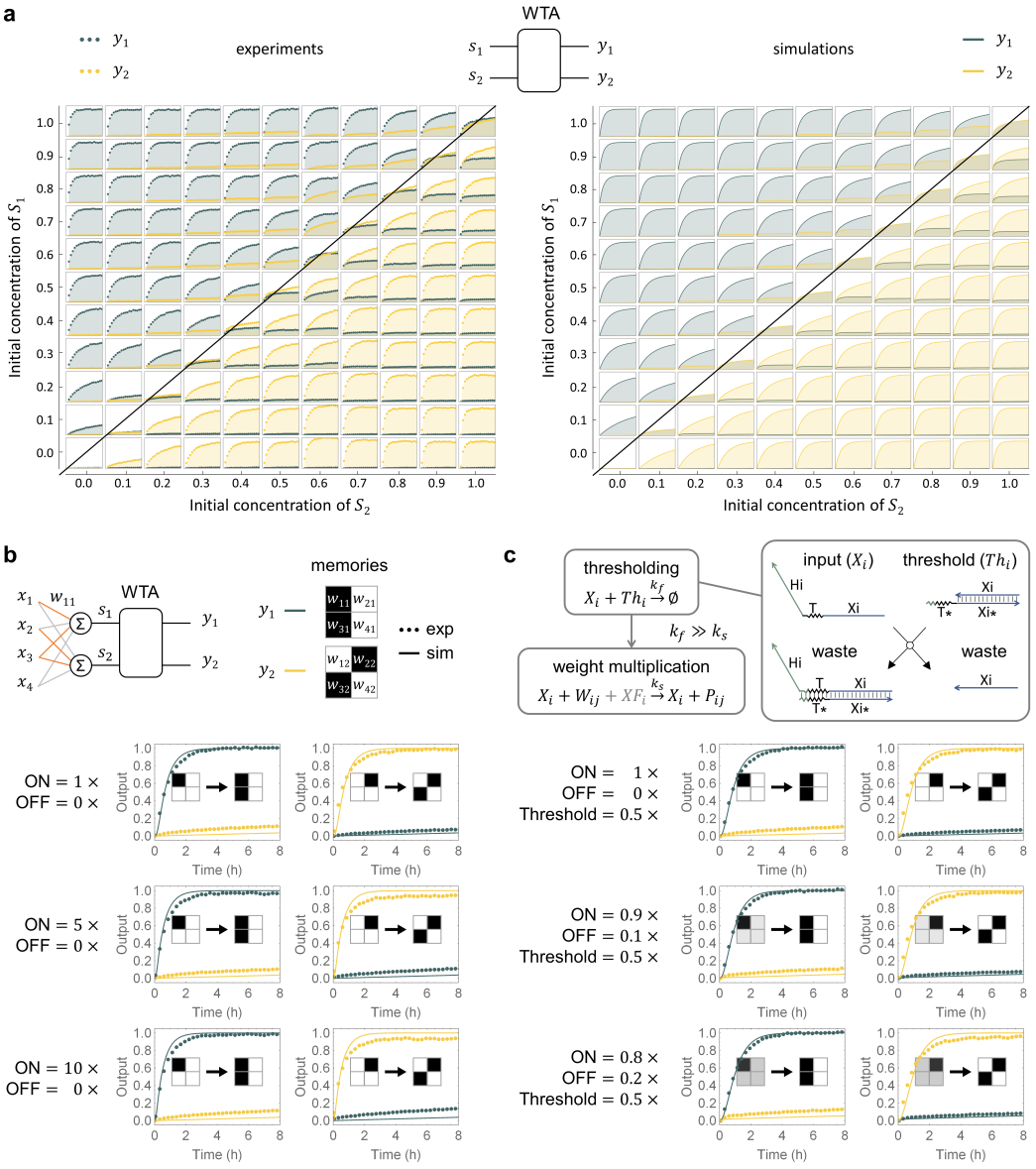


Figure 2.7 | Experimental characterization of winner-take-all DNA neural networks. a, Two-species winner-take-all behavior. The experimental data (left) were used to identify the reverse rate constant $k_r = 0.4$/s of the annihilation reaction in simulation (right). All fluorescence kinetics data and simulation are shown over the course of 2.5 hours. 1× = 50 nM. Initial concentrations of the annihilator, restoration gates, fuels, and reporters are 1.5×, 1×, 2×, and 2×, respectively. **b,** A 4-bit pattern recognition circuit with input concentration varying from 1× to 10×. In each output trajectory plot, dotted lines indicate fluorescence kinetics data and solid lines indicate simulation. The pattern 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 was reported in the previous work on seesaw DNA circuits (Qian and Winfree, 2011). The extended toehold in threshold molecule has 7 nucleotides. In both (b) and (c), for comparing the range of inputs, the concentration of each input strand is shown relative to 1× = 50 nM. Initial concentration of each weight molecule is either 0 or 50 nM. Initial concentrations of the annihilator, restoration gates, fuels, and reporters are 4×, 1×, 2×, and 2×, respectively, with 1× = 100 nM.

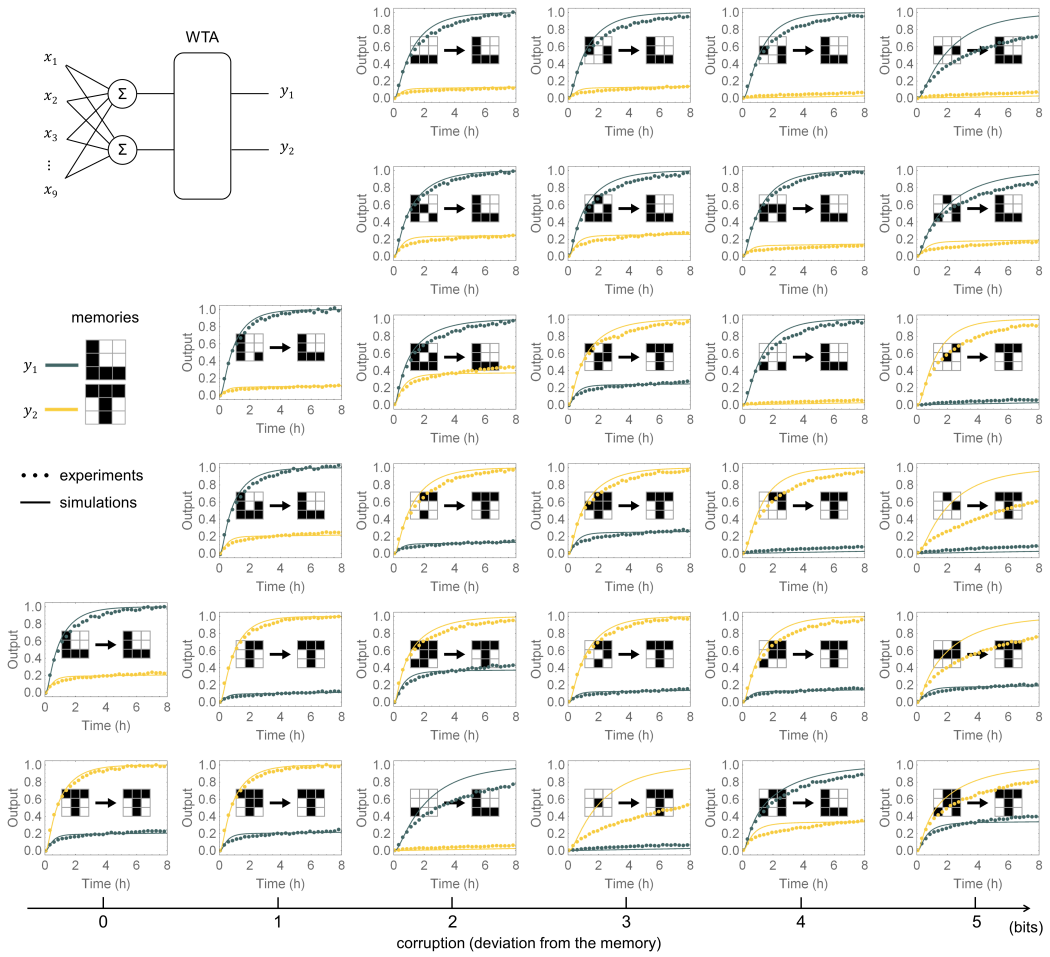


Figure 2.8 | A winner-take-all DNA neural network that recognizes 9-bit patterns as 'L' or 'T.' In each output trajectory plot, dotted lines indicate fluorescence kinetics data and solid lines indicate simulation. 1× = 50 nM. Initial concentration of each input strand is either 0× or 1×. Initial concentration of each weight molecule is either 0× or 0.2×. Initial concentrations of the annihilator, restoration gates, fuels, and reporters are 1.5×, 1×, 2×, and 2×, respectively. The pattern 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 to 5 corrupted bits were tested. Note that 5 is the maximum number of corrupted bits, because a 'L' with more than 5-bit corruption will be as similar or more similar to a 'T,' and vice versa.

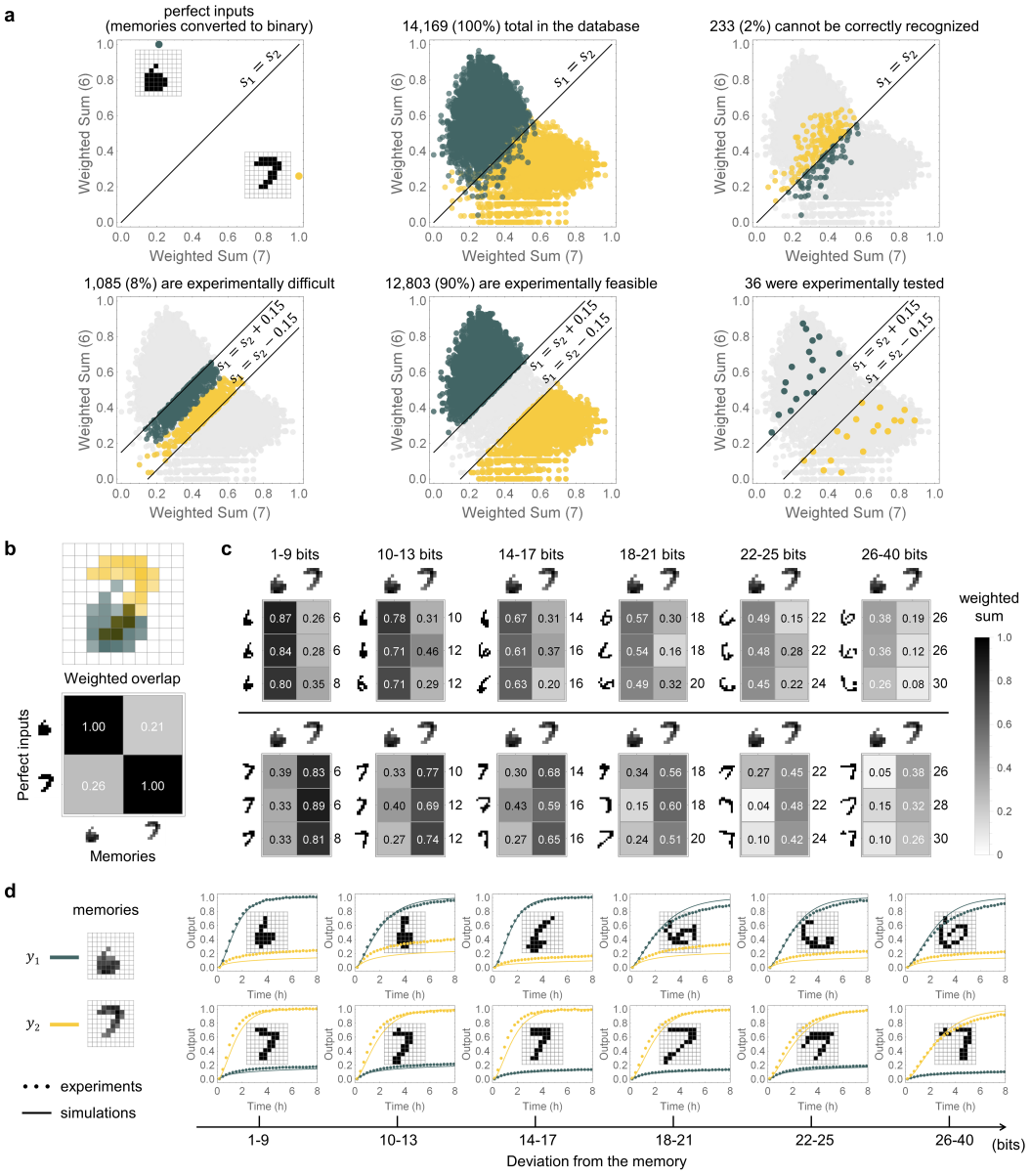
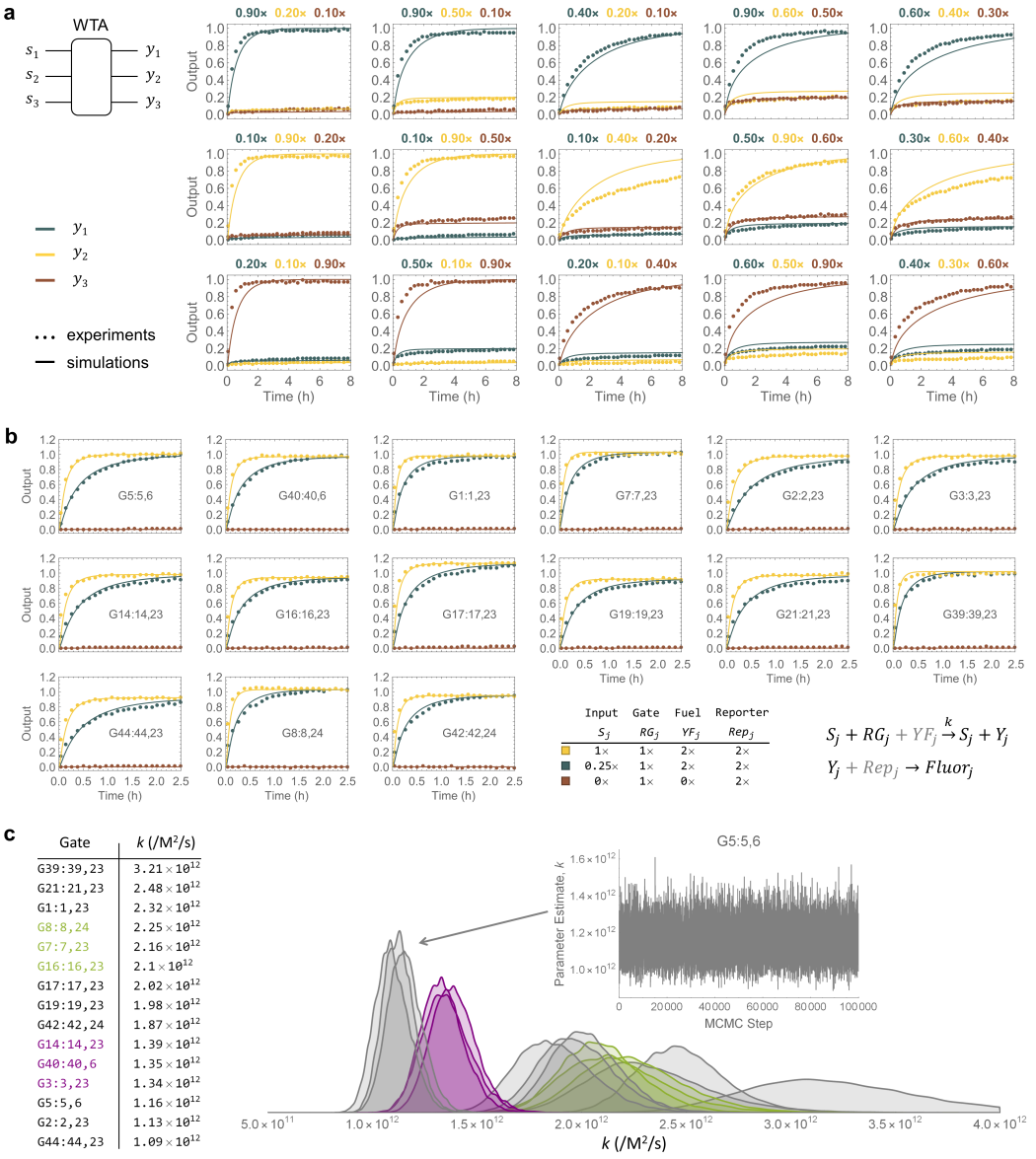


Figure 2.9 | A winner-take-all DNA neural network that recognizes 100-bit patterns as one of two handwritten digits. a, Choosing the test input patterns based on their locations in the weighted sum space. **b,** Overlap between the two memories: '6' and '7.' **c,** 36 test patterns with the number of 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. 1× = 100 nM. Initial concentrations of all species are listed in Extended Data Fig. 2.14. The input pattern is shown on 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}

Figure 2.10 | Three-species winner-take-all behavior and rate measurements for selecting DNA sequences in winner-take-all reaction pathways. a, Fluorescence kinetics data of 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 1× = 50 nM. Initial concentrations of the annihilators, restoration gates, fuels, and reporters are 1.5×, 1×, 2×, and 2×, respectively. **b,** Measuring the rates of 15 catalytic gates. Fluorescence kinetics data (dotted lines) and simulations (solid lines) of the signal restoration reaction, with a trimolecular rate constant (k) fitted using a Mathematica Markov Chain Monte Carlo (MCMC) package (Burkart, 2015). 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 1× = 50 nM. **c,** The 15 catalytic gates sorted and grouped based on their rate constants. All rate constants are within ±65% of the median. The two colored 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, Sequence design).

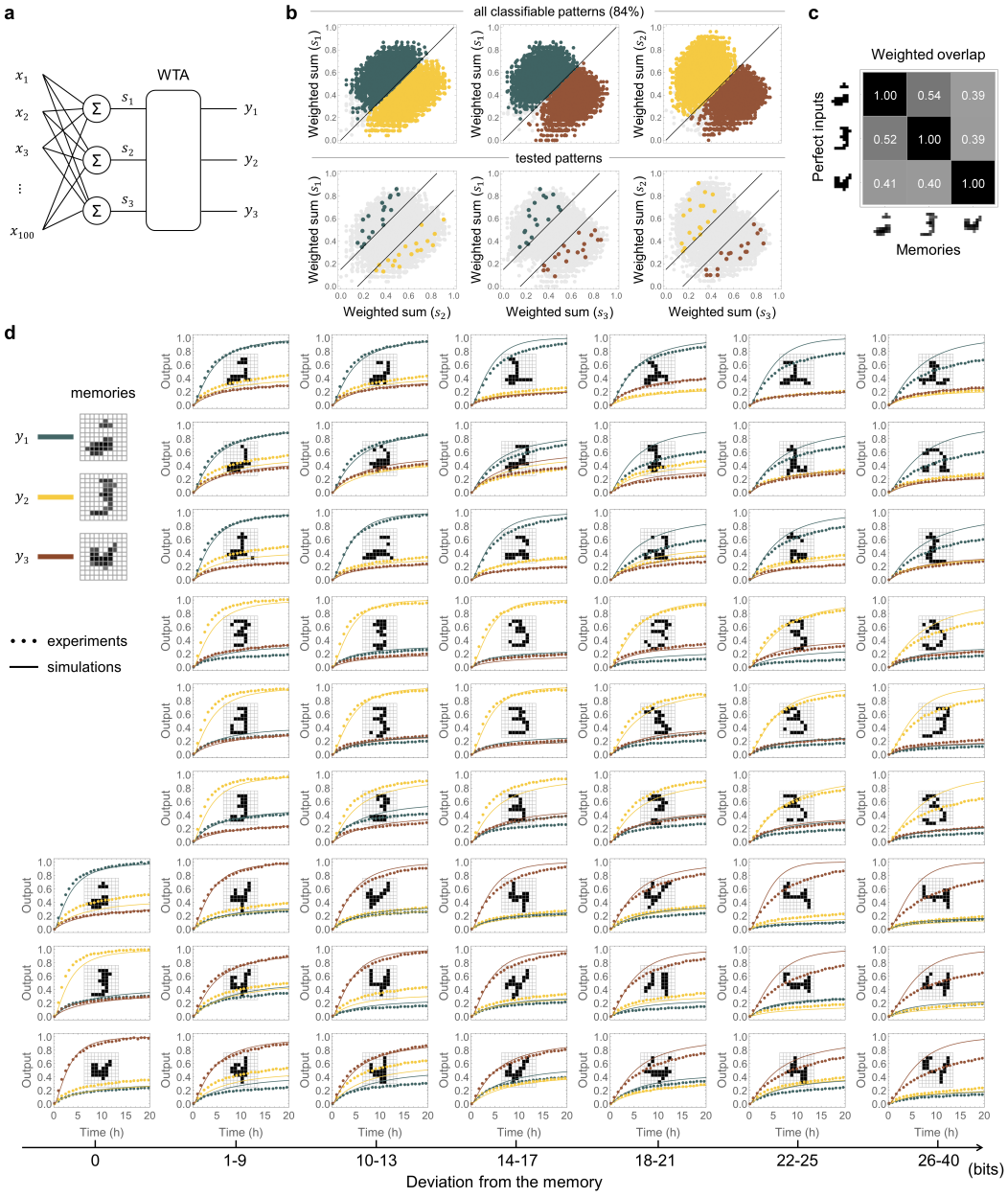


Figure 2.11 | A winner-take-all DNA neural network that recognizes 100-bit patterns as one of three handwritten digits. a, Circuit diagram. **b,** Choosing the test input patterns based on their locations in the weighted sum space. **c,** Overlap between the three memories: '2,' '3,' and '4.' **d,** 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. 1× = 100 nM. Initial concentrations of all species are listed in Extended Data Fig. 2.14. The input pattern is shown on each plot. Note that 40 is the maximum number of flipped bits, because all patterns have exactly 20 1s.

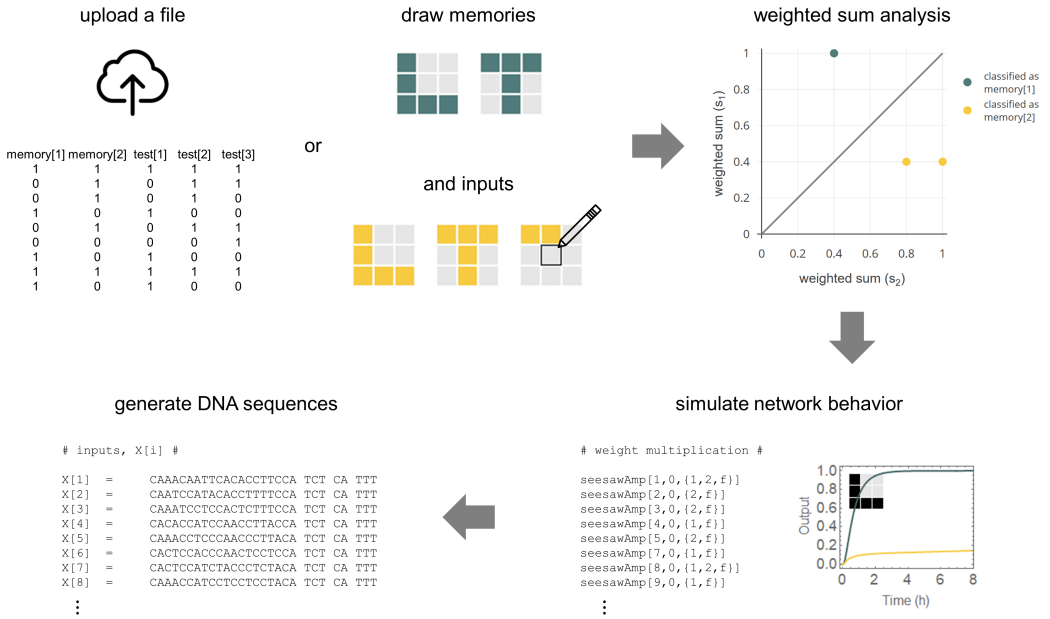


Figure 2.12 | Workflow of the winner-take-all compiler. The compiler (Cherry, 2017) is a software tool for designing DNA-based winner-take-all neural networks. Users can start by uploading a file describing a WTA neural network. Alternatively, the weight matrix and test patterns can be graphically drawn. 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 with Mathematica code downloaded from the compiler website, and the set of the reaction functions are displayed online. Finally, the compiler produces a list of DNA strands required to experimentally demonstrate the network as designed by the user.

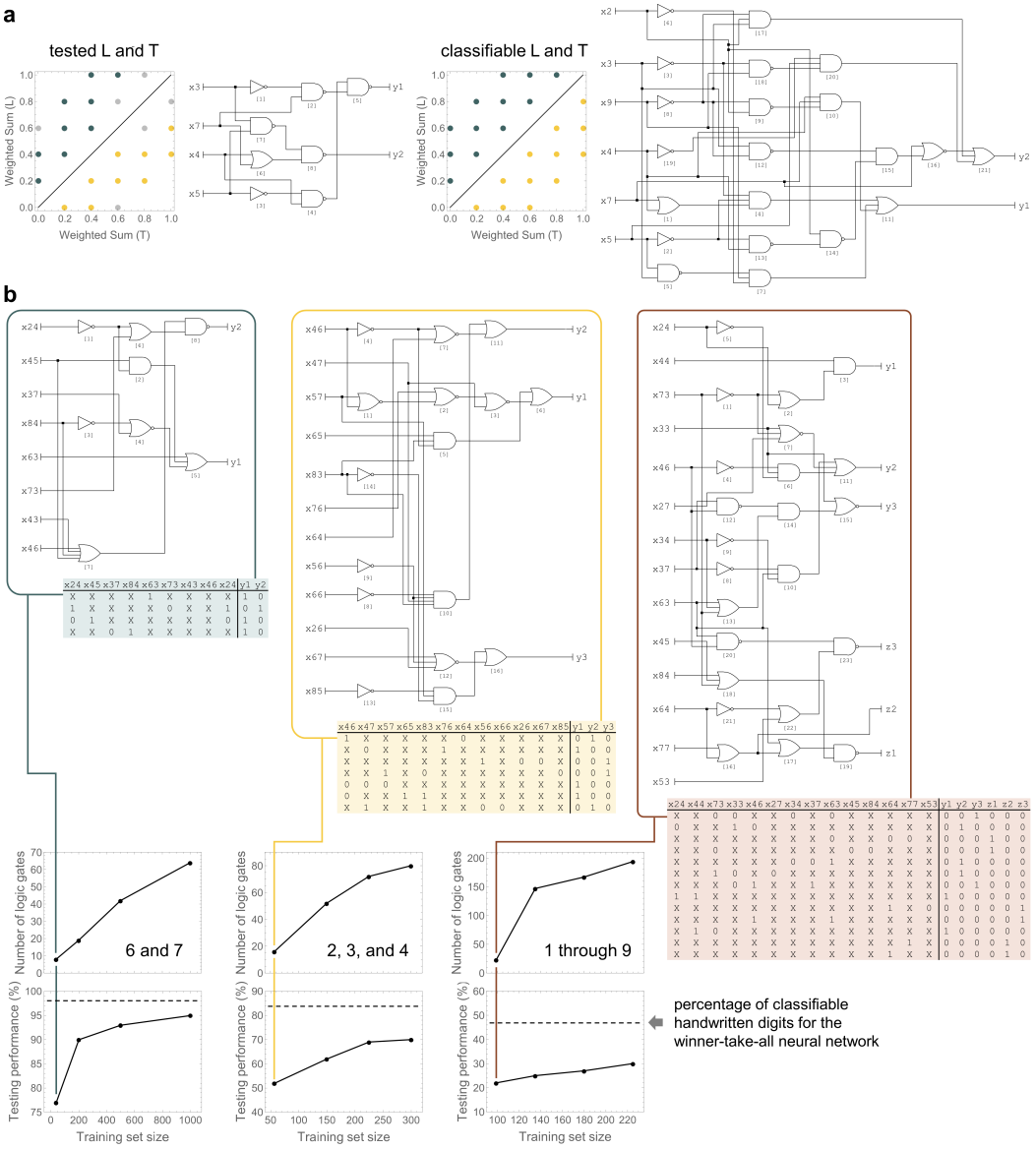


Figure 2.13 | Size and performance analysis of logic circuits for pattern recognition. a, Logic circuits that distinguish if a 9-bit pattern is more similar to L or T. **b,** Logic circuits that recognize 100-bit handwritten digits. To find a logic circuit that produces correct outputs for a given set 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. 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 input was restricted to the 16 most significant bits based on 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

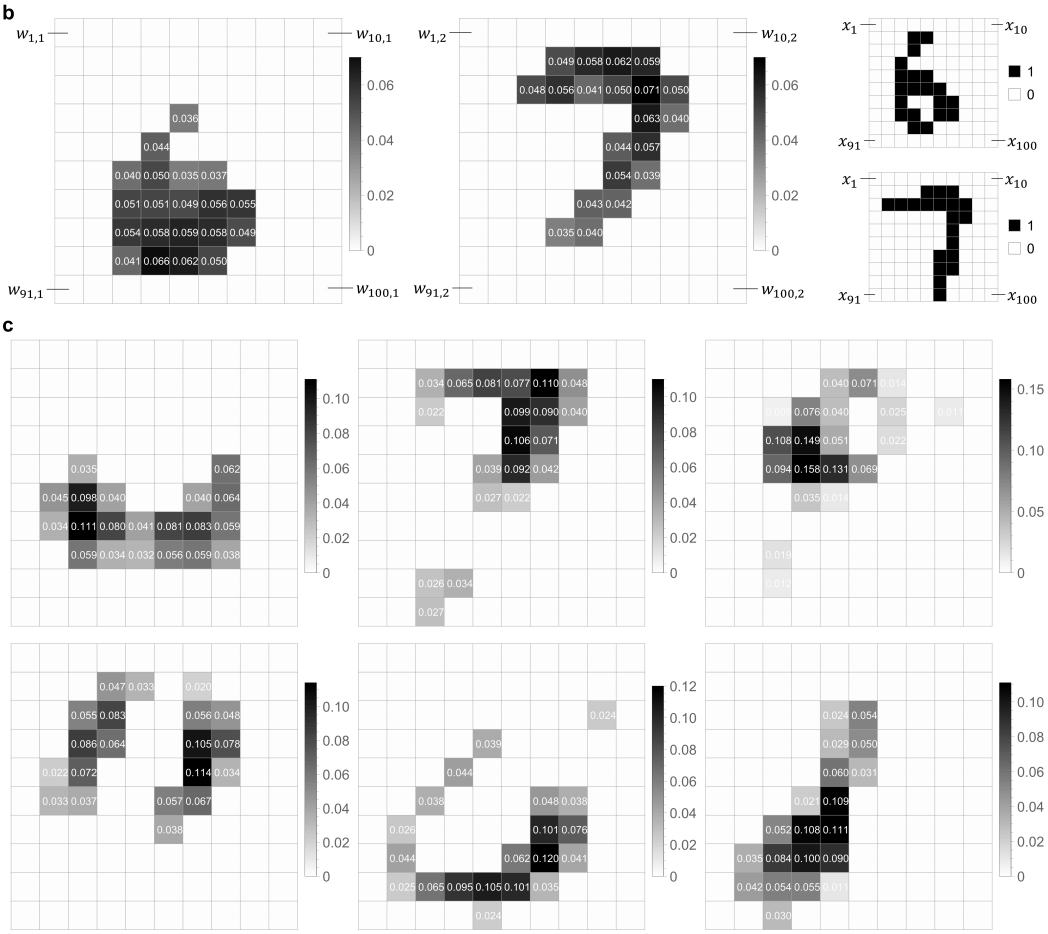


Figure 2.14 | Species and their initial concentrations in all neural networks that recognize 100-bit patterns. a, List of species and strands. Reporters were annealed with top strands (i.e., `Rep[j]-t`) in 20% excess. All other two-stranded complexes were annealed with a 1:1 ratio of the two strands and then PAGE purified (Methods, Purification). **b,** Weights and example inputs in the neural network that recognizes '6' and '7.' **c,** Weights in the neural network that recognizes '1' through '9.'

Chapter 3

SUPERVISED LEARNING IN DNA-BASED WINNER-TAKE-ALL NEURAL NETWORKS

[pages 37-65 temporarily redacted]

Chapter 4

CONCLUSION

A DNA-based neural network implementing the winner-take-all function is an effective architecture for classifying very large, complex patterns. Because it requires only positive weights, it is an efficient network architecture for molecular computation. The simple, robust DNA molecules along with the pair-wise annihilation approach to computing the nonlinear operation results in a scalable molecular network much larger than previous experimental demonstrations. The digital input and output mean that the network can be composed with other types of molecular computation. One limitation is the scaling of the number of molecules required for pair-wise input annihilation as more memories are added, so it is best suited for recognizing large input patterns and classifying them into a few target classes; however, a carefully determined weight matrix and orthogonal WTA neurons provides a suitable solution for adding more memories. Finally, the ability to use the target patterns as the weight matrix allowed us to construct a molecular network capable of dynamically learning new weights from the environment. The development of new, activatable molecular motifs provided the means for implementing supervised learning in a DNA-based artificial neural network. The network learns from labeled training patterns and then classifies unseen molecular test patterns.

4.1 A Look into the future

Here, I discuss just a few of the tantalizing next steps and open questions related to WTA networks and molecular design. These topics frequently occupied my thoughts, but I could never devote enough time to finding satisfying answers. I encourage others who find this work interesting to consider these intriguing inquiries.

Potential of k-WTA networks

The WTA network architecture implemented in this work utilizes a destructive WTA layer as the non-linear function where the signal strands annihilate each other until only one signal species remains. However, another mechanism for computing WTA with global feedback was demonstrated by Kim *et al.* (2004) and Genot *et al.* (2013). This mechanism employs strands competing for a shared, limited resource. The concentration of a signal strand will correlate with its success in acquiring

the shared resource which boosts its concentration. All strands are also subject to degradation. If tuned carefully, the balance of production and degradation will select the winning species. The challenge of this mechanism is needed to carefully balance the supply of shared resource and the rate of degradation. These parameters limit the range of signal species concentration in which this mechanism will be successful. However, this method also has some major benefits. The network is much less limited in the number of competing signal species, as the number of required molecules scales linearly instead of quadratic with my pairwise annihilation approach. What is equally exciting is careful choice of parameters will allow implementation of k-WTA, which is a more general, powerful version of WTA, compared with the 1-WTA network implemented in my work here.

The theory papers by Maass (1999) and Maass (2000) provide insights on the power of WTA neural networks. It is known that logic circuits are capable of universal computation, and linear threshold unit (LTU) circuits can implement any logic circuit. The author shows a method for converting any feed forward LTU neural network into a k-WTA circuit. In a k-WTA network, the ith output has value 1 only if the ith input was among the k largest inputs. My work only implemented k-WTA, where $k = 1$. However, using the competitive WTA approach mentioned above, one may be able to implement a k-WTA network in DNA with k much larger than 1. I have not explored which features of a linear threshold circuit result in which required values for k. My cursory explorations of a handful of networks only produced converted k-WTA networks with $k \geq 2$. Which classes of logic circuits and LTU networks could be simplified, reduced, and implemented as DNA k-WTA network remains an intriguing outstanding question. It is possible, through this conversion to k-WTA, that logic circuits much larger than any experimentally demonstrated to date may be constructed. This could be a break-through in circumventing scaling limitations currently encounter in the largest, state-of-the-art, well-mixed chemical reaction networks.

Providing structure to WTA weights

My networks presented here often have stored memories and classify test patterns that are visually interesting, meaningful shapes, such as the MNIST digits. Our human visual systems recognize handwritten digits due to the proximity of the bit information. This belies a sense of structure to the information not present in the well-mixed chemical system. I show the digits as a 10x10 grid, but the chemical system would be equally well shown as a 1x100 vector of bits. Truly, the

WTA system has much less information than the human brain is receiving when viewing an image of a digit—the WTA completely lacks structural information. This realization often disappoints the audience when they understand the network's weakness in classifying patterns with affine transformations like shifts or rotations. Although, much biological information does not inherently contain structure like diagnostic tests for identification and classification of patterns of small molecules. A motivated student could add more weight layers which account for pixel location, where structure is important like with images. These additional layers could be logical tests of subsets of the patterns, or more generally, they could be convolutions across the pattern. These convolutions could blur or sharpen, for example, and also serve to reduce the large size of input data. However, one should be aware that adding weight layers could strongly affect the time at which competing summation species are produced possibly resulting in erroneous computation. Inspired by my work, Xiong *et al.* (2022) devised a system that could implement convolutional neural networks by adding more orthogonal matrix multiplication operations to the input layer in a multi-step process. This allowed them to classify test patterns with more bits into more target classes.

I spent some time considering the layout and structure of the weight matrix, but it was in the service of minimizing leak reactions that would adversely affect network performance. In the early design of the learning network, each bit was uniquely identified by a toehold-length domain that underwent branch migration. The first few initiating nucleotides would necessarily be shared with other bit sequences due to the limited supply of unique domains, and therefore some cross-talk between bits would occur. If these cross-talking bits were placed near each other, the resulting weighted input from a leaky test pattern would be similar to that produced by a blurry test image. This works for MNIST digits because an ON bit is more likely to be adjacent to other ON bits. The handwritten digit is a connected image. This strategy would not work on highly non-contiguous patterns resembling checkerboards. In pursuit of optimizing this layout, I built a fully connected spring model with spring constants proportional to the similarity between each pair of sequences and then simulated a 2D thermal annealing before assigning each bit to a grid location. This layout strategy worked well, but ultimately the cross-talk between was too great to continue pursuing this DNA molecule design.

Activatable seesaw gates as transistors

The activatable seesaw gates utilized in the weight layer of the learning WTA network bear a strong resemblance to electronic relays or transistors. Transistors are among the most significant inventions in history and serve as the foundation of modern computing technology. A transistor is a fundamental electronic component employed to amplify or switch electronic signals. It accomplishes this by regulating the flow of an electrical signal between the source and the drain through control of the voltage on the gate terminal. I dedicated a substantial amount of time to studying the molecular design of the activatable seesaw gate. When an activator is present, the molecular gate allows signals to pass from an input to an output representation. Like modern central processors, these molecular gates could allow construction of general network circuits with nodes selectively activated to control the creation of many alternate functions. There is only a minor, though important, difference between the molecule and electronic transistor. In a transistor the voltage on the gate terminal can be much smaller than the controlled voltage passing between the source and drain. In my molecular design the source is catalytic, but it only produces output up to the limit of the concentration of the activator. A small change making the activator catalytic would remove this limitation. The transistor-like molecular gate could have many applications in diagnostic and molecular detection devices that require signal amplification with low leak.

4.2 Philosophy on project selection

It is a most unfortunate truth about science that someone who wants to know everything, learn everything, and build everything, only has time to wade into one minuscule area of knowledge. To make a splash, a scientist must immerse themselves in one specialized area, leaving countless other fields untouched. Graciously, there is great satisfaction in probing one idea to its vast depths—becoming an expert, albeit in a singular domain. They get to explore the boundaries of an area entirely of their choosing, so they should endeavour to make it count.

I want my contribution to science to be more than just the completion of a small engineering project contained neatly within the blip of time of one person's graduate school tenure. The impact should be more than the project's direct purpose—its stated goals. Beyond the data and the graphs, a project should inspire. It should spawn more projects and encourage more exploration. A project's greatest achievement is not the publication's final figure with the most complex graphic. The loftiest achievement of a scientific work nudges a field in a wholly new direction.

Not to the detriment of other directions, but a new, additional direction—stretching the domain of a research discipline ever larger.

In this sense, academic fields are like water droplets on a surface. Often they are distinct and non-overlapping. Occasionally they touch; a connection is made between them and new ideas quickly infuse throughout previously disconnected fields. A truly successful project can pull a droplet into a new direction, and if there is enough interest, akin to strong surface tension, a quantity of the droplet will follow and the field expands—sometimes onto a wholly new, previously dry, uncharted surface and sometimes nearer to other droplets.

I hope that I was able to achieve this to some degree with my research project. I have undoubtedly expanded the field of molecular computing. By implementing a DNA-based winner-take-all neural network architecture that is much larger than any previously demonstrated network, I showed that a molecular network could classify larger, more interesting patterns into a larger number of distinct memories, and it could handle corrupted input patterns. Additionally, by adding adaptive behavior to that neural network—where the molecules learn from environmental examples—I also brought that allegorical water droplet of the field of molecular computing closer to another interesting droplet: machine learning. New students have been inspired to join our scientific community after hearing about my research, and I have seen more molecular computing research projects focused on building neural networks. And interest in molecular learning has piqued. Time will tell the extend of this impact, but germinated seeds are taking hold throughout our small community—even if my work was only able to wet the surface.

BIBLIOGRAPHY

Adleman, Leonard M. (1994). "Molecular computation of solutions to combinatorial problems". In: *Science* 266.5187, pp. 1021–1024.

Aubert, Nathanaël, Quang Huy Dinh, Masami Hagiya, Teruo Fujii, Hitoshi Iba, Nicolas Bredeche, and Yannick Rondelez (2013). "Evolving cheating DNA networks: a case study with the rock-paperscissors game". In: *Advances in Artificial Life* 12, pp. 1143–1150.

Baek, Christina, Sang-Woo Lee, Ra Yoo, Beomjin Lee, and Byoung-Tak Zhang (2018). "Intelligent DNA learning system with multiple iterations of weight update". In: *Korean Information Society Academic Presentation*, pp. 811–813.

Burkart, Josh (2015). *Mathematica Markov Chain Monte Carlo package*. `https://github.com/joshburkart/mathematica-mcmc`.

Cardelli, Luca and Attila Csikász-Nagy (2012). "The cell cycle switch computes approximate majority". In: *Scientific Reports* 2, p. 656.

Chen, Sherry Xi and Georg Seelig (2017). "A DNA neural network constructed from molecular variable gain amplifiers". In: *Lecture Notes in Computer Science* 10467, pp. 110–121.

Chen, Yuan-Jyue, Neil Dalchau, Niranjan Srinivas, Andrew Phillips, Luca Cardelli, David Soloveichik, and Georg Seelig (2013). "Programmable chemical controllers made from DNA". In: *Nature Nanotechnology* 8.10, pp. 755–762.

Cherry, Kevin M. (2017). *Winner-take-all DNA neural network compiler*. `http://www.qianlab.caltech.edu/WTAcompiler/`.

Cherry, Kevin M. and Lulu Qian (2018). "Scaling up molecular pattern recognition with DNA-based winner-take-all neural networks". In: *Nature* 559.7714, pp. 370–376.

Cho, Eun Jeong, Joo-Woon Lee, and Andrew D. Ellington (2009). "Applications of aptamers as sensors". In: *Annual Review of Analytical Chemistry* 2, pp. 241–264.

Deng, Li (2012). "The MNIST database of handwritten digit images for machine learning research [best of the web]". In: *IEEE Signal Processing Magazine* 29.6, pp. 141–142.

Fernando, Chrisantha T., Anthony M.L. Liekens, Lewis E.H. Bingle, Christian Beck, Thorsten Lenser, Dov J. Stekel, and Jonathan E. Rowe (2009). "Molecular circuits for associative learning in single-celled organisms". In: *Journal of the Royal Society Interface* 6.34, pp. 463–469.

Genot, Anthony J., Teruo Fujii, and Yannick Rondelez (2013). "Scaling down DNA circuits with competitive neural networks". In: *Journal of The Royal Society Interface* 10.85, p. 20130212.

Kim, Jongmin, John Hopfield, and Erik Winfree (2004). "Neural network computation by in vitro transcriptional circuits". In: *Advances in neural information processing systems* 17.

Lakin, Matthew R., Amanda Minnich, Terran Lane, and Darko Stefanovic (2014). "Design of a biochemical circuit motif for learning linear functions". In: *Journal of the Royal Society Interface* 11.101, p. 20140902.

LeCun, Yann, Corinna Cortes, and Christopher J.C. Burges (n.d.). *The MNIST database of handwritten digits*. `http://yann.lecun.com/exdb/mnist/index.html`.

Li, Bingling, Andrew D. Ellington, and Xi Chen (2011). "Rational, modular adaptation of enzyme-free DNA circuits to multiple detection methods". In: *Nucleic Acids Research* 39.16, e110–e110.

Maass, Wolfgang (1999). "Neural computation with winner-take-all as the only nonlinear operation". In: *Advances in neural information processing systems* 12, pp. 293–299.

Maass, Wolfgang (2000). "On the computational power of winner-take-all". In: *Neural Computation* 12.11, pp. 2519–2535.

McCulloch, Warren S. and Walter Pitts (1943). "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4, pp. 115–133.

Okamoto, Akimitsu, Kazuo Tanaka, and Isao Saito (2004). "DNA logic gates". In: *Journal of the American Chemical Society* 126.30, pp. 9458–9463.

Pei, Renjun, Elizabeth Matamoros, Manhong Liu, Darko Stefanovic, and Milan N. Stojanovic (2010). "Training a molecular automaton to play a game". In: *Nature Nanotechnology* 5.11, pp. 773–777.

Qian, Lulu and Erik Winfree (2011). "Scaling up digital circuit computation with DNA strand displacement cascades". In: *Science* 332.6034, pp. 1196–1201.

Qian, Lulu, Erik Winfree, and Jehoshua Bruck (2011). "Neural network computation with DNA strand displacement cascades". In: *Nature* 475.7356, pp. 368–372.

Redgrave, Peter, Tony J. Prescott, and Kevin Gurney (1999). "The basal ganglia: a vertebrate solution to the selection problem?" In: *Neuroscience* 89.4, pp. 1009–1023.

Rojas, Raúl (2013). *Neural networks: a systematic introduction*. Springer Science & Business Media.

Saghatelian, Alan, Nicolas H. Völcker, Kevin M. Guckian, Victor S.-Y. Lin, and M. Reza Ghadiri (2003). "DNA-based photonic logic gates: AND, NAND, and INHIBIT". In: *Journal of the American Chemical Society* 125.2, pp. 346–347.

Seelig, Georg and David Soloveichik (2009). "Time-complexity of multilayered DNA strand displacement circuits". In: *International Workshop on DNA-Based Computers*. Springer, pp. 144–153.

Seelig, Georg, Bernard Yurke, and Erik Winfree (2005). "DNA hybridization catalysts and catalyst circuits". In: *DNA Computing: 10th International Workshop on DNA Computing, DNA10, Milan, Italy, June 7-10, 2004, Revised Selected Papers 10*. Springer, pp. 329–343.

Shin, Jong-Shik and Niles A. Pierce (2004). "A synthetic DNA walker for molecular transport". In: *Journal of the American Chemical Society* 126.35, pp. 10834–10835.

Thubagere, Anupama J., Chris Thachuk, Joseph Berleant, Robert F. Johnson, Diana A. Ardelean, Kevin M. Cherry, and Lulu Qian (2017). "Compiler-aided systematic construction of large-scale DNA strand displacement circuits using unpurified components". In: *Nature Communications* 8, p. 14373.

Turberfield, Andrew J., J.C. Mitchell, Bernard Yurke, Allen P. Mills Jr, M.I. Blakey, and Friedrich C. Simmel (2003). "DNA fuel for free-running nanomachines". In: *Physical Review Letters* 90.11, p. 118102.

Xiong, Xiewei, Tong Zhu, Yun Zhu, Mengyao Cao, Jin Xiao, Li Li, Fei Wang, Chunhai Fan, and Hao Pei (2022). "Molecular convolutional neural networks with DNA regulatory circuits". In: *Nature Machine Intelligence* 4.7, pp. 625–635.

Yang, Xiaolong, Yanan Tang, Sarah M. Traynor, and Feng Li (2016). "Regulation of DNA strand displacement using an allosteric DNA toehold". In: *Journal of the American Chemical Society* 138.42, pp. 14076–14082.

Yin, Peng, Hao Yan, Xiaoju G. Daniell, Andrew J. Turberfield, and John H. Reif (2004). "A unidirectional DNA walker that moves autonomously along a track". In: *Angewandte Chemie International Edition* 43.37, pp. 4906–4911.

Yurke, Bernard, Andrew J. Turberfield, Allen P. Mills, Friedrich C. Simmel, and Jennifer L. Neumann (2000). "A DNA-fuelled molecular machine made of DNA". In: *Nature* 406.6796, pp. 605–608.

Zadeh, Joseph N., Conrad D. Steenberg, Justin S. Bois, Brian R. Wolfe, Marshall B. Pierce, Asif R. Khan, Robert M. Dirks, and Niles A. Pierce (2011). "NUPACK: analysis and design of nucleic acid systems". In: *Journal of Computational Chemistry* 32.1, pp. 170–173.

Zhang, David Yu (2010). "Cooperative hybridization of oligonucleotides". In: *Journal of the American Chemical Society* 133.4, pp. 1077–1086.

Zhang, David Yu and Georg Seelig (2010). "DNA-based fixed gain amplifiers and linear classifier circuits". In: *Lecture Notes in Computer Science* 6518, pp. 176–186.

Zhang, David Yu and Georg Seelig (2011). “Dynamic DNA nanotechnology using strand-displacement reactions”. In: *Nature Chemistry* 3.2, pp. 103–113.

Zhang, David Yu and Erik Winfree (2009). “Control of DNA strand displacement kinetics using toehold exchange”. In: *Journal of the American Chemical Society* 131.47, pp. 17303–17314.