PNAS RESEARCH ARTICLE | COMPUTER SCIENCES
BIOPHYSICS AND COMPUTATIONAL BIOLOGY OPEN ACCESS

# Programming and training rate-independent chemical reaction networks

Marko Vasić[a,1,2], Cameron Chalk[a,1,2], Austin Luchsinger[a], Sarfraz Khurshid[a], and David Soloveichik[a,2]

Edited by Darko Stefanovic, The University of New Mexico; received June 29, 2021; accepted March 29, 2022 by Editorial Board Member William H. Press

**Embedding computation in biochemical environments incompatible with traditional electronics is expected to have a wide-ranging impact in synthetic biology, medicine, nanofabrication, and other fields. Natural biochemical systems are typically modeled by chemical reaction networks (CRNs) which can also be used as a specification language for synthetic chemical computation. In this paper, we identify a syntactically checkable class of CRNs called noncompetitive (NC) whose equilibria are absolutely robust to reaction rates and kinetic rate law, because their behavior is captured solely by their stoichiometric structure. In spite of the inherently parallel nature of chemistry, the robustness property allows for programming as if each reaction applies sequentially. We also present a technique to program NC-CRNs using well-founded deep learning methods, showing a translation procedure from rectified linear unit (ReLU) neural networks to NC-CRNs. In the case of binary weight ReLU networks, our translation procedure is surprisingly tight in the sense that a single bimolecular reaction corresponds to a single ReLU node and vice versa. This compactness argues that neural networks may be a fitting paradigm for programming rate-independent chemical computation. As proof of principle, we demonstrate our scheme with numerical simulations of CRNs translated from neural networks trained on traditional machine learning datasets, as well as tasks better aligned with potential biological applications including virus detection and spatial pattern formation.**

chemical computation | neural networks | molecular programming

## Significance

To program complex behavior in environments incompatible with electronic controllers such as within bioreactors or engineered cells, we turn to chemical information processors. While chemical reactions can perform computation in the stoichiometric exchange of reactants for products (how many molecules of which reactants result in how many molecules of which products), the control of reaction rates is usually thought to allow more complex computation. Motivated by the fact that correct stoichiometry is easier to ensure than reaction rates, we provide a method for programming and training chemical computation by stoichiometry. We show that such computation can be straightforwardly programmed in a manner analogous to sequential programming, and demonstrate the execution of neural networks capable of complex machine learning tasks.

Compared to our remarkable capacity to build complex electronic circuits, we lack in our ability to engineer sophisticated reaction networks like the regulatory networks prevalent in biology. Molecular programming aims to engineer synthetic chemical information processors of increasing complexity from first principles. This approach yields control modules compatible with the chemical environments within natural or synthetic cells, bioreactors, and in-the-field diagnostics. Such computation could, for example, recognize a disease state based on chemical inputs and actuate drug delivery to the affected cell.

Chemical reaction networks (CRNs) are key objects of molecular programming. CRNs formally model chemical concentrations changing due to coupled chemical reactions in a well-mixed solution. Biological CRNs are often hard to analyze because, in general, they require working with systems of coupled nonlinear differential equations capable of highly complex dynamical systems behavior such as multistability, oscillation, and chaos (1). However, in engineering, we may aim at specific classes of CRNs that are easier to reason about. One such class has recently emerged in which information processing occurs solely due to the stoichiometric exchange of the reactants for products rather than the reaction rate (2, 3). An example of such computation is the single irreversible reaction $A + B \rightarrow C$ which computes the minimum function in the sense that the concentration of $C$ converges to the minimum of the initial concentrations of $A$ and $B$. By coupling multiple reactions, more-complex functions can be computed. Although stoichiometric computation is effectively limited to continuous piecewise linear (affine) functions, these functions are computationally powerful, as evidenced by their ability to approximate arbitrary functions and their widespread use in machine learning (e.g., neural networks with the rectified linear unit [ReLU] activation function; see below).

Besides ease of analysis, such stoichiometrically computing CRNs are absolutely robust to variations in kinetics (rate independence). Computation carried out by stoichiometry alone is correct whether the system obeys standard mass-action kinetics, Hill function, or Michaelis–Menten kinetics, or any other kinetic laws, and does not err if the system is not well mixed. Engineering may also be aided by the fact that, unlike factors contributing to reaction rates, the stoichiometry of reactants and products is inherently digital and can be set exactly by the nature of the reaction. For example, if realized with DNA strand displacement cascades, the identity and stoichiometry of reactants and products can be programmed by synthesizing DNA strands with specific parts that are identical or

This work was presented in part at the virtual 37th International Conference on Machine Learning, 12–18 July 2020.

Author contributions: M.V., C.C., A.L., S.K., and D.S. designed research; M.V., C.C., A.L., and D.S. performed research; M.V., C.C., A.L., S.K., and D.S. analyzed data; and M.V., C.C., A.L., and D.S. wrote the paper.

The authors declare no competing interest.

This article is a PNAS Direct Submission. D.S. is a guest editor invited by the Editorial Board.



[1] M.V. and C.C. contributed equally to this work.

[2] To whom correspondence may be addressed. Email: vasic@utexas.edu, ctchalk@utexas.edu, or david.soloveichik@utexas.edu.

This article contains supporting information online at https://www.pnas.org/lookup/suppl/doi:10.1073/pnas.2111552119/-/DCSupplemental.

Published June 9, 2022.

complementary (4–6). Note that such reactions can be made effectively irreversible, as they are strongly driven by the formation of new base pairs. Although we are motivated mostly by engineering concerns, some biological CRNs may exhibit similar stoichiometric, rate-independent behavior as identified in searches of the Biomodels repository (7).

In the first part of the paper, we develop a new technique for proving that a class of CRNs stoichiometrically computes the desired function. We identify the noncompetitive property, which means that a species is consumed in, at most, one reaction (see later for a formal definition). We show that, for noncompetitive CRNs, rate independence can be verified and the function computed can be determined by simple reasoning analogous to sequential programming: Although all reactions occur simultaneously with continuously varying rates, we can imagine, counterfactually, that reactions happen sequentially in a series of straight line segments. Noncompetition is easy to check, and, further, fully captures the computational power of stoichiometric computation. Thus, noncompetitive CRNs are a powerful class of CRNs for rationally programming chemical behavior. All subsequent constructions in this paper are noncompetitive, and their correctness is proven via the above technique.

In the second part of this paper, motivated by the widespread use of neural networks to generate behavior that is not easily specified programmatically, we show a natural way to specify rate-independent chemical input–output behavior through training. Specifically, we show how (feed-forward) ReLU neural networks can be directly implemented by noncompetitive CRNs. ReLU neural networks are one of the most successful types of neural networks for deep learning, prevalent in all areas of machine learning. Thus we provide a powerful paradigm for creating chemical systems with complex computational functionality not easily obtained by other means.

The key elements of our general (rational weight) ReLU neural network implementation are the ReLU and the weight multiplication modules. Our ReLU module consists of a single unimolecular and a single bimolecular reaction. Our weight multiplication module uses a number of unimolecular and bimolecular reactions that is proportional to the number of bits of precision in the weight. (Although weight multiplication can be performed with two high-order reactions, such reactions cannot easily be implemented and are slow.)

To simplify the construction even further, we consider restricting the class of ReLU neural networks to have $\{-1, 0, 1\}$ weights. Despite the restriction on the values of the weights, such binary weight ReLU (BReLU) neural networks are known to be powerful in solving machine learning tasks and are well researched in the deep learning community (8). Applying an optimized version of our construction to BReLU networks yields a surprisingly compact CRN with only a single bimolecular reaction per ReLU node (plus additional unimolecular reactions at the input layer of the network).

Showing how two models of computing can simulate each other elucidates the computational power of one model in terms of the other. In the case of stoichiometrically computing CRNs and ReLU neural networks, they are both capable of computing arbitrary continuous piecewise linear (affine) functions. However, since the size of the CRN depends on the digits of precision of the weights, making a quantitative connection between the computational power of the two models (e.g., comparing the number of reactions versus number of ReLU nodes to achieve the same functionality) is difficult. Nonetheless, in the case of BReLU networks, we can make a tight connection to the subclass of noncompetitive CRNs (named CheLU) in which a reaction involves any species at most once and with unit stoichiometry. We show that such CheLU CRNs and BReLU networks can be considered to be equivalent models of computing, as they can simulate each other with the number of ReLU nodes equaling the number of bimolecular reactions.

In the last part of the paper, we demonstrate, through examples, our procedure of using BReLU neural networks to embed functionality in CRNs. For each of the following datasets, we trained the neural network classifier, generated the resulting CRN, and numerically simulated the CRN under the usual mass-action kinetics. The kinetic simulations confirm convergence to the expected output and provide additional information about convergence time. The first datasets, Iris and MNIST (Modified National Institute of Standards and Technology), are widely used in machine learning. The next dataset, motivated by the envisioned application of molecular computation in medical diagnostics, differentiates between viral infections, using chemical information as input (gene expression levels). Finally, an important direction of chemical computation in synthetic biology lies in spatial pattern formation with applications in tissue and organ engineering (9). As an example of spatial pattern formation, we used a neural network to generate a two-dimensional (2D) pattern (heart shape).

## CRNs and Nondeterministic Kinetics

CRNs formally model the time evolution of molecules in a solution undergoing chemical interactions. Besides the use of CRNs to capture the behavior of naturally existing chemical systems, synthetic biologists and molecular programmers often use CRNs as a programming language for rationally designed synthetic chemical networks such as DNA strand displacement cascades (5, 6) and DNA–enzyme networks (10). Related models of distributed computation include population protocols (11), Petri nets (12), and vector addition systems (13).

A CRN consists of a set of species $\Lambda$ and a set of reactions. Reactions are written generally in the following form:

$$r_1 R_1 + \cdots + r_n R_n \longrightarrow p_1 P_1 + \cdots + p_m P_m,$$

where $R_i, P_j \in \Lambda$ are the reactant and product species, respectively, and the $r_i, p_j \in \mathbb{N}$ are stoichiometric coefficients quantifying how much of each species is produced and how much is consumed. The reactions written this way are irreversible—the products cannot react to form the reactants—and any reverse reaction must be explicitly included (e.g., $R_1 + R_2 \longrightarrow P$ and $P \longrightarrow R_1 + R_2$). While reactions always have some degree of reversibility, synthetic chemical reactions, such as DNA strand displacement cascades, can be made highly irreversible. While the results of *Programming CRN Computation by Stoichiometry* apply to reactions with arbitrarily many reactants, the constructions in *Rational Weight ReLU Neural Networks* and *Binary Weight ReLU Neural Networks* consist of reactions with, at most, two reactants. Reactions with more than two reactants are slow, in practice, as they require the colocalization of more than two molecules before reactions can occur. Further, while simulation of high-order reactions by bimolecular ones is possible, the typical method disturbs kinetics and does not fit in the noncompetitive class (defined later) we are focusing on.*

A state of a CRN is an assignment of nonnegative real-valued concentrations (amount per volume) to each species. It helps to pick an arbitrary ordering on the species so that we can view states

*The typical method for simulating, for example, the reaction $3X \longrightarrow Y$ is to use the reactions $X + X \rightleftharpoons X_1$ and $X + X_1 \longrightarrow Y$.

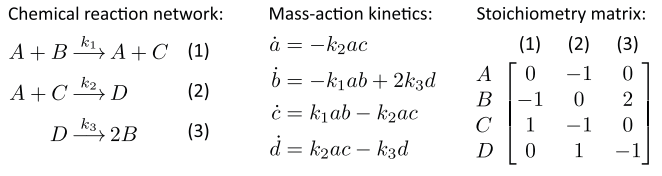


**Fig. 1.** Representations of CRNs. The law of mass action induces the differential equations describing the CRN's change in concentrations over time, where, for example, *a* represents the concentration of species *A*. The stoichiometry matrix captures the net change in species by each reaction, where entry *i*, *j* corresponds to the change in species *i* by applying reaction *j*.

as column vectors from $\mathbb{R}^{\Lambda}_{\geq 0}$ for compatibility with linear algebra techniques used later. We use $\mathbf{a}(S)$ to denote the concentration of species $S$ in state $\mathbf{a}$. We write column vectors as row vectors for notational convenience.

The state of a CRN changes over time as prescribed by a rate law. For example, the most commonly applicable rate law is the mass-action kinetics model (example in Fig. 1) which prescribes differential equations from reaction rates proportional to the product of the reactants' concentrations. While we do show that our results hold for mass-action kinetics, we further prove them for any rate laws which satisfy basic fairness (allowing reactions to eventually happen if reactants are present) and stoichiometric constraints. We give this class of reasonable rate laws a formal definition at the end of this section, after providing a useful formalization of stoichiometric constraints.

Next, we present a nondeterministic kinetic model, first proposed by Chen et al. (2), designed to isolate the effect of stoichiometry from the effect of rates. This model does not intend to capture real-world chemical kinetics directly. Instead, it is a simplified model that aids analysis of CRNs: As we show, for the class of CRNs of interest, convergence in this simplified model implies convergence under mass-action kinetics or any other reasonable rate law. Intuitively, the model explores the set of states reachable by the CRN assuming nothing about the kinetics besides that stoichiometry is obeyed.

To capture the stoichiometry of a CRN, we focus on the stoichiometry matrix $\mathbf{M}$ (example in Fig. 1). Each column corresponds to a reaction, and each row corresponds to a species: $\mathbf{M}_{ij}$ corresponds to the net increase/decrease of species $i$ by applying reaction $j$.

In a state $\mathbf{a}$, we say a reaction is applicable if all of its reactants have positive concentration. Flux vectors, which are column vectors $\mathbf{u} \in \mathbb{R}^{\Lambda}_{\geq 0}$, describe arbitrary, simultaneous applications of reactions, which, when multiplied by the stoichiometry matrix $\mathbf{M}$, yield the change in concentrations caused by applying those reactions. Since $\mathbf{u}$ is a vector whose positive entries describe which reactions apply, we say $\mathbf{u}$ is (initially) applicable at a state $\mathbf{a}$ if all species which are reactants of reactions with positive entries in $\mathbf{u}$ have positive concentration in $\mathbf{a}$; formally, flux vector $\mathbf{u}$ is applicable at state $\mathbf{a}$ if, for all reactions $R$, $\mathbf{u}(R) > 0$ implies that all reactants $S$ of reaction $R$ have $\mathbf{a}(S) > 0$.

For states $\mathbf{a}$ and $\mathbf{b}$, we say $\mathbf{a} \to^1_{\mathbf{u}} \mathbf{b}$ if there is a flux vector $\mathbf{u}$ applicable at $\mathbf{a}$ such that $\mathbf{b} = \mathbf{M}\mathbf{u} + \mathbf{a}$; this is straight-line reachability.[†] Given $\mathbf{a} \to^1_{\mathbf{u}} \mathbf{b}$, we say reaction $R$ is being applied if $\mathbf{u}(R) > 0$. We say $\mathbf{a} \to \mathbf{b}$ if there is a finite length sequence $\mathbf{a} \to^1 \cdots \to^1 \mathbf{b}$, that is, $\to$ is the transitive reflexive closure of $\to^1$; this is called line segment reachability. If no flux vectors $\mathbf{u}$ besides the zero vector are applicable at state $\mathbf{b}$, then we call $\mathbf{b}$ a static state. In other words, in a static state, at least one reactant of every reaction has zero concentration.

Intuitively, the nondeterministic model and the reachability relation $\to$ describe what could happen in any (reasonable) rate law. Or, put another way, if $\mathbf{a} \not\to \mathbf{b}$, then this trajectory is prevented by stoichiometry.

For our proofs, the most practical general characterization of rate laws is a function mapping time to the total flux through each reaction. Formally, given a CRN with $r$ reactions and an initial state $\mathbf{a}$, any rate law induces a continuous, monotonic function $\mathcal{F}_{\mathbf{a}} : \mathbb{R}_{\geq 0} \to \mathbb{R}^r_{\geq 0}$ such that $\mathcal{F}_{\mathbf{a}}(0) = \mathbf{0}$ (the zero vector). Using this function $\mathcal{F}_{\mathbf{a}}$ and the stoichiometry matrix $\mathbf{M}$, the state $\mathbf{b}$ of the CRN at time $t$ is given by $\mathbf{b} = \mathbf{M}\mathcal{F}_{\mathbf{a}}(t) + \mathbf{a}$. We say $\mathbf{a}$ converges to $\mathbf{b}$ under the rate law if $\lim_{t\to\infty} \mathcal{F}_{\mathbf{a}}(t)$ is finite and $\mathbf{M} \lim_{t\to\infty} \mathcal{F}_{\mathbf{a}}(t) + \mathbf{a} = \mathbf{b}$.

Using this formalization of rate law, we can define the general class of reasonable rate laws for which our main *Theorem* holds, as follows:

***Definition 1:*** Reasonable rate law. A rate law is reasonable if it satisfies the fairness and stoichiometric constraints below.

***Definition 2:*** Fairness constraint. If $\lim_{t\to\infty} \mathcal{F}_{\mathbf{a}}(t)$ exists and is finite, then $\mathbf{b} = \mathbf{M} \lim_{t\to\infty} \mathcal{F}_{\mathbf{a}}(t) + \mathbf{a}$ must be a static state (i.e., the rate law cannot imply a limit state which stops applying reactions but has applicable reactions).

***Definition 3:*** Stoichiometric constraint. For times $t_1, t_2$ with $t_1 < t_2$, if $\mathbf{b} = \mathbf{M}\mathcal{F}_{\mathbf{a}}(t_1) + \mathbf{a}$ and $\mathbf{c} = \mathbf{M}\mathcal{F}_{\mathbf{a}}(t_2) + \mathbf{a}$, then it must be that $\mathbf{b} \to^1_{\mathbf{u}_1} \cdots \to^1_{\mathbf{u}_n} \mathbf{c}$ with total flux $\sum_{i=1}^{n} \mathbf{u}_i = \mathcal{F}_a(t_2) - \mathcal{F}_a(t_1)$.

If *Definition 2* were not satisfied, a rate law could induce an $\mathcal{F}$ that simply applies no reactions at all. *Definition 3* ensures that the rate law is consistent with the nondeterministic model. In particular, we avoid rate laws in which reactions happen when some of their reactants are absent.[‡]

Chen et al. (2) proved that mass-action kinetics is reasonable. One only needs to prove that a relevant kinetic model is reasonable in order to apply our main *Theorem*.

## Programming CRN Computation by Stoichiometry

The computational power of CRNs typically arises from both kinetics and stoichiometry. However, the equilibrium of certain CRNs can be understood entirely by the stoichiometric exchange of reactants for products (Fig. 2). Such systems have been used as an alternate paradigm for programming complex chemical behavior (2, 3, 14), inspired by similar notions in distributed computing (11). We call such CRNs stoichiometrically defined.[§]

To view CRNs as a method of computation (or a programming language), given a function $f : \mathbb{R}^n_{\geq 0} \to \mathbb{R}^m_{\geq 0}$, we assign some species to be input species and others to be output species. The input species $X_1, \ldots, X_n$ and an initial concentration assignment to each represents an input vector $\mathbf{x}$. For a fixed rate law, for example, mass-action kinetics with particular rate constants, we say

[†] Removing the applicability constraint would trivialize finding the set of reachable states of the CRN but would lead to erroneous analysis. For example, given the CRN $X_1 + X_2 \longrightarrow Y + Z$, $Z \longrightarrow X_2$, given the ordering on species $X_1, X_2, Y, Z$ and an initial state $\mathbf{a} = [10, 0, 0, 0]$, state $\mathbf{b} = [0, 0, 10, 0]$, and flux vector $\mathbf{u} = [10, 10]$, we would have that $\mathbf{b} = \mathbf{M}\mathbf{u} + \mathbf{a}$, although, from $\mathbf{a}$, no reactions should be applicable, because there is initially zero concentration of $X_2$ and $Z$.

[‡] Note that requiring that $d\mathcal{F}_{\mathbf{a}}(t)(R)/dt > 0$ if reaction $R$ is applicable in $\mathbf{M}\mathcal{F}_{\mathbf{a}}(t) + \mathbf{a}$ is not enough: Consider the CRN with only one reaction $R : X \longrightarrow 2X$ and a rate law which induces $\mathcal{F}_{\mathbf{0}}(t)(R) = t^2$. We have $d\mathcal{F}_{\mathbf{0}}(0)(R)/dt = 0$, and the reaction is not applicable at $\mathbf{0}$, but this rate law is pathological—the reaction makes a positive amount of $X$ from zero but has $X$ as a reactant. Such pathological examples exist even if one enforces the rate of change of $\mathcal{F}$ to depend solely on the concentrations of the reactant species (3; footnote 8).

[§] Previous work calls this notion stable computation. We use the term stoichiometrically defined to avoid confusion with other notions of stability in chemistry.

| Rate-dependent: | | Stoichiometric: |
|---|---|---|
| $X \longrightarrow Y$ | $X + X \longrightarrow \text{waste}$ | $2X \longrightarrow Y$ |
| $Y \longrightarrow X$ | $X \longrightarrow Y$ | |

**Fig. 2.** Two rate-dependent CRNs and one stoichiometrically defined CRN computing $y = x/2$. For certain choices of rate laws, each CRN converges to a state with concentration of $Y$ equal to half the initial concentration of $X$, but *Left* and *Middle* CRNs require strong assumptions about the rate law. *Left* CRN converges correctly under mass-action kinetics with equal rate constants on both reactions. *Middle* CRN converges correctly if the second reaction happens at twice the rate of the first, but, interestingly, cannot converge correctly under mass-action kinetics for any rate constants. *Right* CRN converges correctly with only a mild assumption that the rate is not zero while $X$ has a positive concentration. Intuitively, the correctness of *Right* CRN is solely due to the stoichiometry of two reactants to one product.

the CRN computes $f$ if the concentrations of the output species $Y_1, \ldots, Y_m$ as time goes to infinity are the output vector $\mathbf{y}$ such that $f(\mathbf{x}) = \mathbf{y}$. For our computation to be stoichiometrically defined, we want to prove that a CRN computes $f$ with respect to any rate law, which is achieved by the *Theorem* below. Note that a dynamic equilibrium results from the precise balance in rates between conflicting reactions; therefore, stoichiometrically defined CRNs must reach static equilibria where no reactions are applicable.

A small example is the reaction $X_1 + X_2 \longrightarrow Y$ which computes $f(x_1, x_2) = \min(x_1, x_2)$, since the reaction converges to a state where either $X_1$ or $X_2$, whichever has initially lower concentration, is depleted. Another example appears in Fig. 3 which computes $f(x) = (2/3)x$. More examples and discussion of stoichiometrically defined CRN computation can be found in *SI Appendix*, section E.

**Noncompetitive CRNs.** Here we identify a class of CRNs which we will show are easy to analyze and yet do not lose any computational power if we are interested in stoichiometrically defined, rate-independent computation. Consider the following CRN:

$$X \longrightarrow Y_1$$
$$X \longrightarrow Y_2.$$

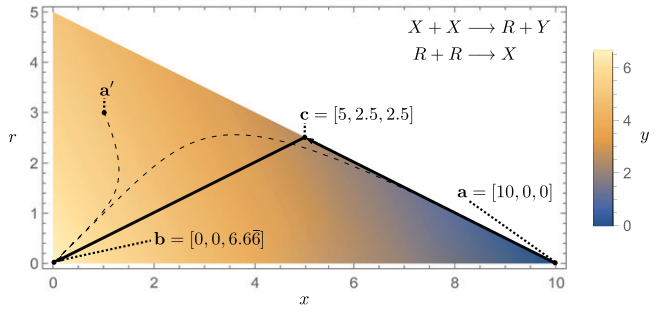


**Fig. 3.** Example application of the *Theorem* on the noncompetitive CRN $X + X \longrightarrow R + Y$, $R + R \longrightarrow X$ with initial state $\mathbf{a} = [10, 0, 0]$. The shaded region shows all line segment reachable states from $\mathbf{a}$, that is, all states $\mathbf{d}$ such that $\mathbf{a} \rightarrow \mathbf{d}$. Solid lines are straight-line reachable paths (specifically, $\mathbf{a} \rightarrow^1 \mathbf{c}$ and $\mathbf{c} \rightarrow^1 \mathbf{b}$), and dashed lines are mass-action trajectories (assuming both reactions have rate constant one, although the theorem applies to any rate constants). Since there is a path $\mathbf{a} \rightarrow \mathbf{b}$ and $\mathbf{b}$ is static, the *Theorem* implies that $\mathbf{a}$ also converges to $\mathbf{b}$ under mass action or any other reasonable rate law. Further, as shown by the state $\mathbf{a}'$, any state line segment reachable from $\mathbf{a}$ will also converge to $\mathbf{b}$ under mass action or any other reasonable rate law, showing that the convergence is robust to any initial perturbations that do not leave the line segment reachable space of states. Letting the concentration of $X$ in $\mathbf{a}$ be a variable $x(0)$, generalizing the shown path gives $\mathbf{a} \rightarrow \mathbf{b}$ with $\mathbf{b} = [0, 0, (2/3)x(0)]$.

Intuitively, the two reactions compete for the species $X$. The faster reaction will yield more output than the other, so the output state depends on the rates of the reactions. Formalizing this intuition yields the following definition:

***Definition 4:*** Noncompetitive CRNs. A CRN is noncompetitive if every species which is decreased in a reaction is a reactant in only that reaction.

Note that, by the definition above, a reactant may appear in any number of reactions if it is not decreased (e.g., if it acts as a catalyst). Also note that noncompetition is not necessary to compute rate independently (see an example in *SI Appendix*, section G).

In *SI Appendix*, section C, we prove the following about noncompetitive CRNs:

**Theorem.** *For noncompetitive CRNs, if* $\mathbf{a} \rightarrow \mathbf{b}$ *and* $\mathbf{b}$ *is a static state, then, for any state* $\mathbf{a}'$ *such that* $\mathbf{a} \rightarrow \mathbf{a}'$, $\mathbf{a}'$ *converges to* $\mathbf{b}$ *under any reasonable rate law.*

Fig. 3 illustrates a small application of this theorem. The precondition of this theorem, that $\mathbf{a} \rightarrow \mathbf{b}$ with $\mathbf{b}$ static, is the same as providing a line segment path from the input state to a static state with the correct output. Thus, this theorem greatly simplifies the analysis of equilibrium for noncompetitive CRNs. Further, the theorem states that any state that is line segment reachable from the initial state still converges correctly under any reasonable rate law. The path $\mathbf{a} \rightarrow \mathbf{a}'$ captures a wide class of perturbations, allowing any adversarial conditions to be applied to the system initially, such as non–well mixedness or withholding of certain reactions, as long as stoichiometry is still obeyed. Then, as long as any reasonable rate laws are allowed to take over, the system converges to the output state $\mathbf{b}$. (Note that $\mathbf{a}'$ can be equal to $\mathbf{a}$, since $\mathbf{a} \rightarrow \mathbf{a}$, meaning that this theorem also implies convergence from the initial state.)

**Composition and Dual-Rail Logic.** To translate ReLU neural networks to CRNs, and generally to construct scalable computation, we desire networks that are composable, meaning a network computing $g(x)$ should be straightforwardly concatenated to a network computing $f(x)$ to compute $g(f(x))$. It turns out that this is equivalent to the following condition (15): We say that a CRN is composable if its output species $Y_1, \ldots, Y_n$ do not appear as reactants in the network. (See *SI Appendix*, section E.1 for more details on composability.)

There is a tight connection between composability and noncompetition. Note that, if a CRN is not composable according to the above definition, then one reaction has an output species $Y$ as a reactant. Then, attempting to compose a downstream CRN adds another reaction with $Y$ as a reactant, resulting in a competitive network.

The composability constraint weakens computational power unless the dual-rail representation is used. (See *SI Appendix*, section E for more details.) The dual-rail representation utilizes two species, for example, $X^+$ and $X^-$, to represent one real-valued signal $x$, where $x = x^+ - x^-$, that is, the difference between concentrations of species $X^+$ and $X^-$. Note that dual rail allows representation of negative values by concentrations which must be nonnegative. We use the dual-rail representation in our ReLU network constructions.

**Computational Power.** Dual-rail stoichiometrically defined CRNs compute exactly the continuous piecewise rational affine functions (*SI Appendix*, section E; see Fig. S5, *Right* for an example). Importantly, restricting to the noncompetitive subclass does not limit this computational power; this argument is made in *SI Appendix*, section E. The expressiveness of continuous piecewise affine functions is underwritten by the empirical power

of ReLU neural networks, which compute exactly the same functions. Thus we motivate the connection between CRNs and ReLU neural networks, and explore this connection in more detail in *Rational Weight ReLU Neural Networks* and *Binary Weight ReLU Neural Networks*.

## Rational Weight ReLU Neural Networks

In this and the subsequent section, we develop constructions for implementing ReLU neural networks with noncompetitive CRNs. We start with broadly allowing arbitrary rational weights in this section, and focus on binary weights in *Binary Weight ReLU Neural Networks*.

Rational weight ReLU neural networks (RReLU) are neural networks with rational weights and ReLU activation function. Fig. 4*A* shows an example RReLU neural network, consisting of an input layer, a single hidden layer, and an output layer. The output of the network is defined by $y = ReLU(\mathbf{W_2} \cdot ReLU(\mathbf{W_1} \cdot \mathbf{x} + \mathbf{r_1}) + r_2)$, where $\mathbf{x} \in \mathbb{R}^2$ is an input vector, $\mathbf{W_1} \in \mathbb{Q}^{2\times 2}$ is a weight matrix into the hidden layer, $\mathbf{r_1} \in \mathbb{R}^2$ is a vector of bias terms, $\mathbf{W_2} \in \mathbb{Q}^{1\times 2}$ is a weight vector into the output layer with $r_2$ as the corresponding bias term, and $y \in \mathbb{R}$ is the output. In our example, $\mathbf{r_1} = [-3/2, 1/2]$ (column vector), $\mathbf{W_2} = [4, 4]$ (row vector), and

$$\mathbf{W_1} = \begin{bmatrix} 1 & 1 \\ -1/2 & -1/2 \end{bmatrix}.$$

Fig. 4*C* shows an implementation of such RReLU networks with composable, noncompetitive CRN modules (fan out, weighted sum, and ReLU). For each module separately, we describe a line segment path to a static state with the correct output, and thus, by the *Theorem*, each module performs the desired computation. We finish this section by discussing how the modules are composed to realize the entire RReLU network. The paths described below assume zero initial concentration of all species other than the inputs. To implement bias terms, we set the initial concentrations of the corresponding species to the dual-rail value of the bias terms.

The fan-out module passes a single value to $n$ downstream neurons. Applying the two reactions of the module until completion results in a static state with the dual-rail value $y_i = y_i^+ - y_i^- = x^+(0) - x^-(0) = x(0)$ for every $i$. The weighted sum module combines the outputs of multiple predecessor neurons by multiplying them with a weight (rational number) and summing up the values. Applying reactions $q_j X_j^+ \longrightarrow p_j Y^+$ and $q_j X_j^- \longrightarrow p_j Y^-$ until completion changes the dual-rail value $y$ of the output by $(p_j/q_j)(x_j^+(0) - x_j^-(0)) = (p_j/q_j)x_j(0)$. Similar reactions are included for the other input species of the weighted sum (note that positive and negative species are flipped in the case of a negative-signed weight). Thus, after applying all the reactions, we reach a static state where the total dual-rail value of $y$ is equal to the weighted sum of the inputs.

While rational weight multiplication is easily computable through stoichiometry as above (e.g., $qX \longrightarrow pY$ computes $y = (p/q)x$), the use of many reactants is undesirable, as discussed in *CRNs and Nondeterministic Kinetics*. We can use the scheme shown in Fig. 5 for rational weight multiplication using only noncompetitive unimolecular and bimolecular reactions. Using reactions of the form $L_0 \longrightarrow L_1 + L_1$ and $R_0 + R_0 \longrightarrow R_1$, we can double and halve the concentration of a species, respectively. In this way, a set of reactions may mimic the binary expansion of a given rational $\frac{p}{q}$, generating an output species $Y$ for each one bit in the binary representation. If the rational number has an infinitely repeating portion in its binary expansion, our CRN uses a final reaction which "loops" back to a previous reaction. Fig. 5*C* shows a concrete example of this. A detailed proof of correctness for this construction may be found in *SI Appendix*, section F. The proof shows a path from a state with concentration $x$ of the input species to a state at static equilibrium with concentration $(p/q)x$ of the output species. By the *Theorem* (and the fact that this CRN is noncompetitive), this is sufficient to show that the construction computes $(p/q)x$. To satisfy the dual-rail representation, the construction is repeated for both the positive $X^+$ and negative $X^-$ species. Since this CRN is composable, it may be used for the weighted sum by creating similar reaction chains for all input species.

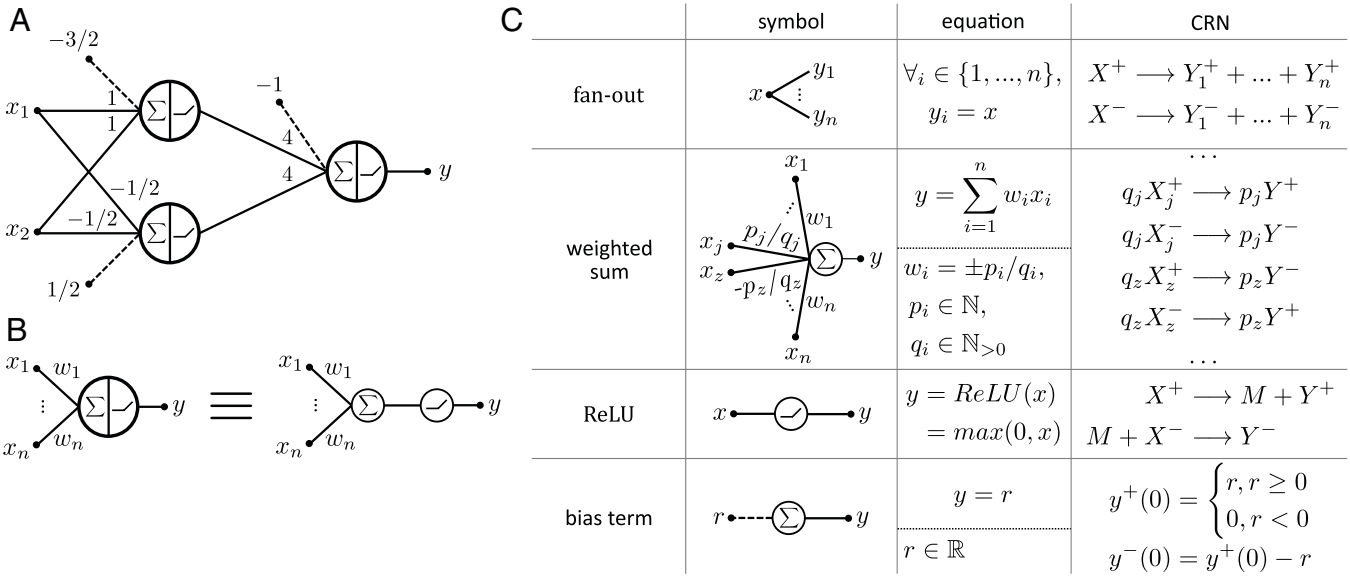


**Fig. 4.** CRN implementation of RReLU neural networks. (*A*) An example RReLU network. (Although the inputs and outputs are real-valued quantities, this network can also be thought to compute the XNOR function: $y = \overline{x_1 \oplus x_2}$ if zero and one values represent logical false and true.) (*B*) Decomposition of a neuron into weighted summation and nonlinearity. (*C*) CRN implementation for each RReLU network component.

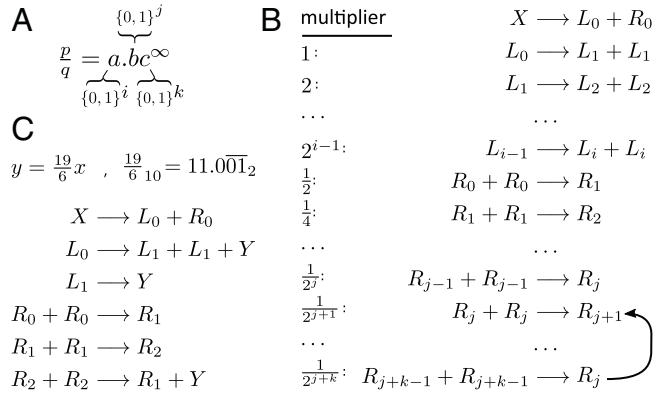


**Fig. 5.** Noncompetitive bimolecular rational multiplication. (*A*) Binary representation of rational $p/q$, where *a*, *b*, and *c* are binary strings. Strings *a* and *b* are lengths *i* and *j*, respectively, while *c* is an infinitely repeated string of length *k*. (*B*) A scheme for constructing a noncompetitive bimolecular CRN for rational multiplication. The reaction chain uses $i + j + k + 1$ reactions to "implement" the binary expansion. The last reaction creates a "loop" in the reaction chain which corresponds to string *c*. The number of times each reaction will be applied (before looping) is some multiple of the initial count of *X*, as indicated by the multiplier column. An output species *Y* will appear as a product for each reaction where a "1" appears in the binary expansion. (*C*) An example CRN which computes $y = (19/6)x$. Note that, to achieve a dual-rail representation, we can repeat this construction twice, for both positive ($X^+$, $Y^+$) and negative ($X^-$, $Y^-$) input and output species.

ReLU is implemented with two reactions shown in Fig. 4.¶ Consider the following line segment path. Apply the first reaction ($X^+ \longrightarrow M + Y^+$) to completion. This results in $y^+ = m = x^+(0)$. Then apply the second reaction ($M + X^- \longrightarrow Y^-$) until completion, resulting in $y^- = \min(m, x^-(0)) = \min(x^+(0), x^-(0))$. In that state, $y = y^+ - y^- = x^+(0) - \min(x^+(0), x^-(0)) = \max(x^+(0) - x^-(0), 0) = \mathrm{ReLU}(x(0))$. Also, $x^+ = 0$ and either $m$ or $x^-$ is zero. Since at least one reactant of both reactions is zero, the state is static, and the *Theorem* applies.

We concatenate the above CRN modules into a single CRN by appropriately renaming species to make the outputs of upstream modules be inputs to the downstream modules (and avoid any other overlap in species). Since each module is noncompetitive and composable, the entire network is noncompetitive. Therefore, applying the line segment paths as described above module by module, layer by layer gives a straightforward path in the nondeterministic kinetic model from the initial state to a static state with the output equal to the output of the neural network. The *Theorem* then argues that the CRN converges correctly under mass-action kinetics or any reasonable rate law. We show an example RReLU neural network and its complete CRN implementation in *SI Appendix*, section A.

Note that, in the dual-rail representation, cancellation reactions, like $X^+ + X^- \longrightarrow$waste, can be added to decrease the concentrations of $X^+$ and $X^-$ without changing the represented value of *x*. Such reactions may help to keep concentrations in check, which may otherwise become exponentially large in the number of layers (17), and may achieve faster convergence. In *SI Appendix*, section I, we prove that adding cancellation does not affect the dual-rail values of the outputs of noncompetitive CRNs.

## Binary Weight ReLU Neural Networks

BReLU neural networks are neural networks with binary weights ($\pm 1$) and ReLU activation function. Since they are a subclass of RReLU networks, the same translation procedure as illustrated for RReLU applies. BReLU networks were popularized in the machine learning community due to the computational speedups they bring (they eliminate the need for a large portion of multipliers which are the most space- and power-hungry components of specialized deep learning hardware), while, at the same time, preserving the performance (8). From the angle of CRNs, computing rational weights $(p/q)\mathbf{x}$ in dual rail requires either two reactions with many reactants or many reactions with at most two reactants, neither of which is desirable. Thus, BReLU networks are a better-suited class of neural networks for CRNs than RReLU, producing CRNs that are easier to implement in a wet lab. In sum, the restriction to binary weights simplifies the implementation of neural networks in both silicon and chemical hardware while maintaining performance.

Note that the fan-out and weighted sum can be merged into a single step, since BReLU networks have $\pm 1$ weights. Thus, by default, the fan-out and weighted sum of BReLU networks is implemented using a reaction set similar to the fan-out module in Fig. 4, with the difference that the $\pm$ signs of the output species are flipped in the case of negative weight.

**Translation Optimization.** We find that unimolecular reactions of noncompetitive CRNs, such as the first reactions of ReLU modules, can be eliminated from the CRN, reducing the total number of reactions. This is achieved by altering the bimolecular reactions and the initial concentrations of the CRN species, a process which we describe next. Unimolecular reactions are those with exactly one reactant like $A \longrightarrow B + C$. Whenever $A$ is produced in another reaction, we can replace it with $B + C$. For example, if there is another reaction $X \longrightarrow A + B$, we replace the reaction with $X \longrightarrow 2B + C$. Further, we adjust the initial concentrations of the product species ($B$ and $C$) by increasing them by the initial concentrations of the reactant ($A$). Importantly, this transformation works only if $A$ is not a reactant in any other reaction; for example, if there were another reaction like $X + A \longrightarrow Y$, it is not clear what to replace instances of $A$ with, and, indeed, it is not possible to remove the unimolecular reaction in that case. Luckily, our constructions are noncompetitive, and we are able to show that, for noncompetitive CRNs, the optimization does not affect the state of convergence (*SI Appendix*, section D). The optimization procedure is illustrated in Fig. 6.

RReLU networks allow for the optimization of fan-out modules and partial optimization of ReLU modules (only the unimolecular reactions). Weighted sum modules may be optimized only in case of integer weights. Note that the unimolecular reactions corresponding to the input species are not optimized, in order not to alter the input to the system. Following the above optimization procedure for BReLU networks results in a CRN which consists of 1) a single bimolecular reaction per RReLU node and 2) two unimolecular reactions per input of the neural network.

Optimization of some adversarial ReLU networks results in reactions whose products have stoichiometric coefficients exponential in the depth of the network. An example is given in *SI Appendix*, section L. Understanding the scaling of the number of products is an important avenue for future work to ensure feasible CRNs.

**BReLU Networks Simulate CRNs.** Noncompetitive CRNs can compute any function computed by a BReLU network where each reaction (except for the input layer reactions) corresponds to one BReLU node. One interpretation of this is that CRNs efficiently

¶ Enumeration of small CRNs shows that this is the simplest stoichiometrically defined, composable CRN computing ReLU in the sense that ReLU cannot be computed in this manner with fewer than two reactions or five species (16).

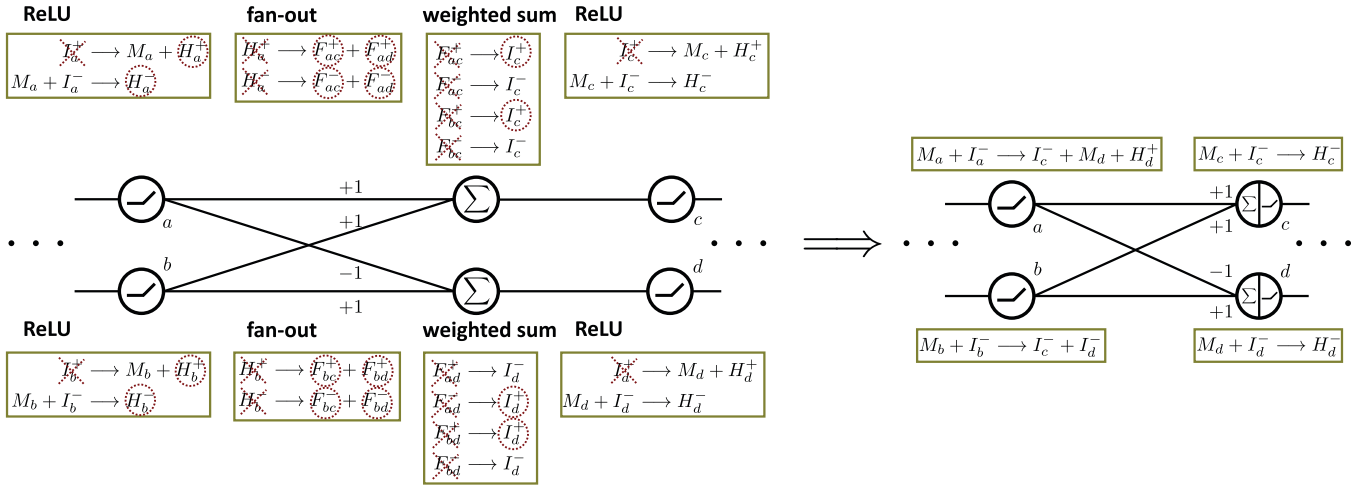


**Fig. 6.** CRN optimization procedure. (*Left*) Neural network and its corresponding CRN before the optimization. (*Right*) Neural network and its corresponding CRN after the optimization.

simulate BReLU networks. A natural question is the converse: Can any CRN be efficiently simulated by a BReLU network?

In this subsection, we show that BReLU networks and a subclass of noncompetitive CRNs we call CheLU are effectively equivalent and can simulate each other with one node per bimolecular reaction and vice versa.

To define CheLU networks, we note the restricted form of the CRNs produced by our translation, and the limitations of BReLU networks. The first restriction is that reactions have, at most, two reactants, because reactions with three or more reactants are slow and hard to implement, and reactions with one reactant can be removed via our optimization procedure. The second restriction is that the CRN is feed forward. This can be formalized by saying that there is a total ordering on reactions such that products of a reaction cannot be reactants of a reaction earlier in the ordering. Previous work (3) shows that feed-forward CRNs, and thus CheLU networks, converge to a static equilibrium under mass-action kinetics. The question of the connection between feedback BReLU networks and feedback CRNs remains open. The third restriction is that every species appears, at most, once per reaction. Intuitively, this restriction is placed because a reaction like $X + X \to \ldots$ essentially halves the signal of $X$, which has no analog in binary weight neural networks. Lastly, we restrict the CRNs to be noncompetitive.

We next define what is meant by simulation of CheLU networks by BReLU networks. Of course, BReLU networks have no sense of kinetics or dynamics. For this reason, we disregard kinetics and instead focus on initial and equilibrium states of the CheLU network, and mapping those states to inputs and outputs of a BReLU network. Formally, we say a BReLU neural network simulates a CheLU CRN if, for all initial states $\mathbf{a}$, the equilibrium state $\mathbf{b}$ given $\mathbf{a}$ is equal to the output vector of the BReLU neural network given $\mathbf{a}$ as input.

We give a small, composable BReLU network (Fig. 7) which simulates a single CheLU reaction. Composing this small network to simulate larger CheLU networks is straightforward, since we restrict CheLU networks to be feed forward. The BReLU network uses one ReLU node and two summation nodes per reaction, although the summation nodes can be removed with the clever addition of more edges to achieve one ReLU node per reaction.

Thus, BReLU networks and CheLU networks simulate each other, one node per reaction and vice versa, and so efficient networks in one model transfer to the other. Note that the CheLU conditions are sufficient but possibly not necessary for simulation by BReLU. Although CheLU networks, at first, seem restricted, the empirical power shown of BReLU networks implies that CheLU networks are a rich and powerful class of CRNs, whose restrictions make them easy targets for implementation by synthetic means.

## Simulations

In this section, we showcase several examples of BReLU networks and show the simulation of their chemical counterparts. Our simulations empirically confirm the correctness of our construction, as well as providing information regarding the kinetics of convergence to correct output.

**Datasets.** We trained BReLU networks on Iris (18, 19), MNIST (20), virus infection (21), and spatial pattern formation (heart) datasets. These datasets are briefly summarized below; for more information on the training procedure and datasets used, see *SI Appendix*, section J. We treated every dataset as a multiclass classification problem, with an output unit for every class. The networks were trained to maximize the output unit corresponding to the correct class, and minimize the others. Following our compilation technique for BReLU networks (*Binary Weight ReLU Neural Networks*), we translated the trained neural networks to CRNs, and simulated the resulting reactions under mass-action kinetics using an ordinary differential equation simulator (22). As expected, in all cases, the outputs of the original neural network

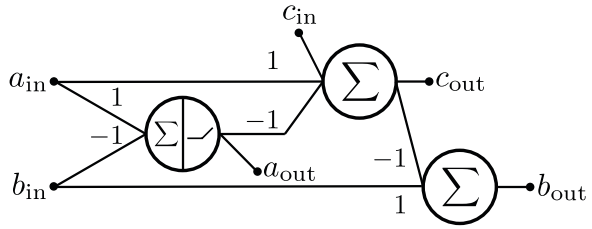


**Fig. 7.** A composable BReLU network simulating a chemical reaction $A + B \longrightarrow C$. Given any vector $\mathbf{x}$ of initial concentrations of species $A$, $B$, and $C$, the equilibrium state $\mathbf{b}$ of the reaction $A + B \longrightarrow C$ has $\mathbf{b}(A) = \mathbf{a}(A) - \min(\mathbf{a}(A), \mathbf{a}(B))$, $\mathbf{b}(B) = \mathbf{a}(B) - \min(\mathbf{a}(A), \mathbf{a}(B))$, and $\mathbf{b}(C) = \min(\mathbf{a}(A), \mathbf{a}(B))$.

and the constructed CRN matched exactly. The numerical experiments are visualized in Fig. 8.

The first two datasets, Iris and MNIST, are popular in the machine learning literature. The Iris dataset classifies flowers into three classes (Setosa, Versicolor, or Virginica) based on four continuous features (petal and sepal length and width). The MNIST dataset classifies gray-scale images as handwritten digits zero to nine. The full MNIST network is the largest network we implemented, with 64 ReLU units; however, a much smaller network of 4 ReLU units, which is potentially more realistic for a CRN implementation, can be used to effectively distinguish between digits zero and one (see *SI Appendix*, section J for additional information).

The last two datasets were chosen to take chemically available information as input. We implemented the virus infection dataset where microarray data capturing human gene expression profiles (we used the 10 most relevant genes as inputs) identify four viral infections: H1N1, H3N2, respiratory syncytial virus, and human rhinovirus. Finally, to demonstrate spatial pattern formation, we trained a neural network to generate a target 2D shape (heart) shown in Fig. 8, *4A*. The input features were the pixel coordinates ($x_1$ and $x_2$); in a chemical system, such a coordinate system could be potentially established via a spatial gradient of different species in the vertical and horizontal directions. The output was the pixel color at that coordinate (black or white). Note that, if raw biological signals are taken as inputs, they are

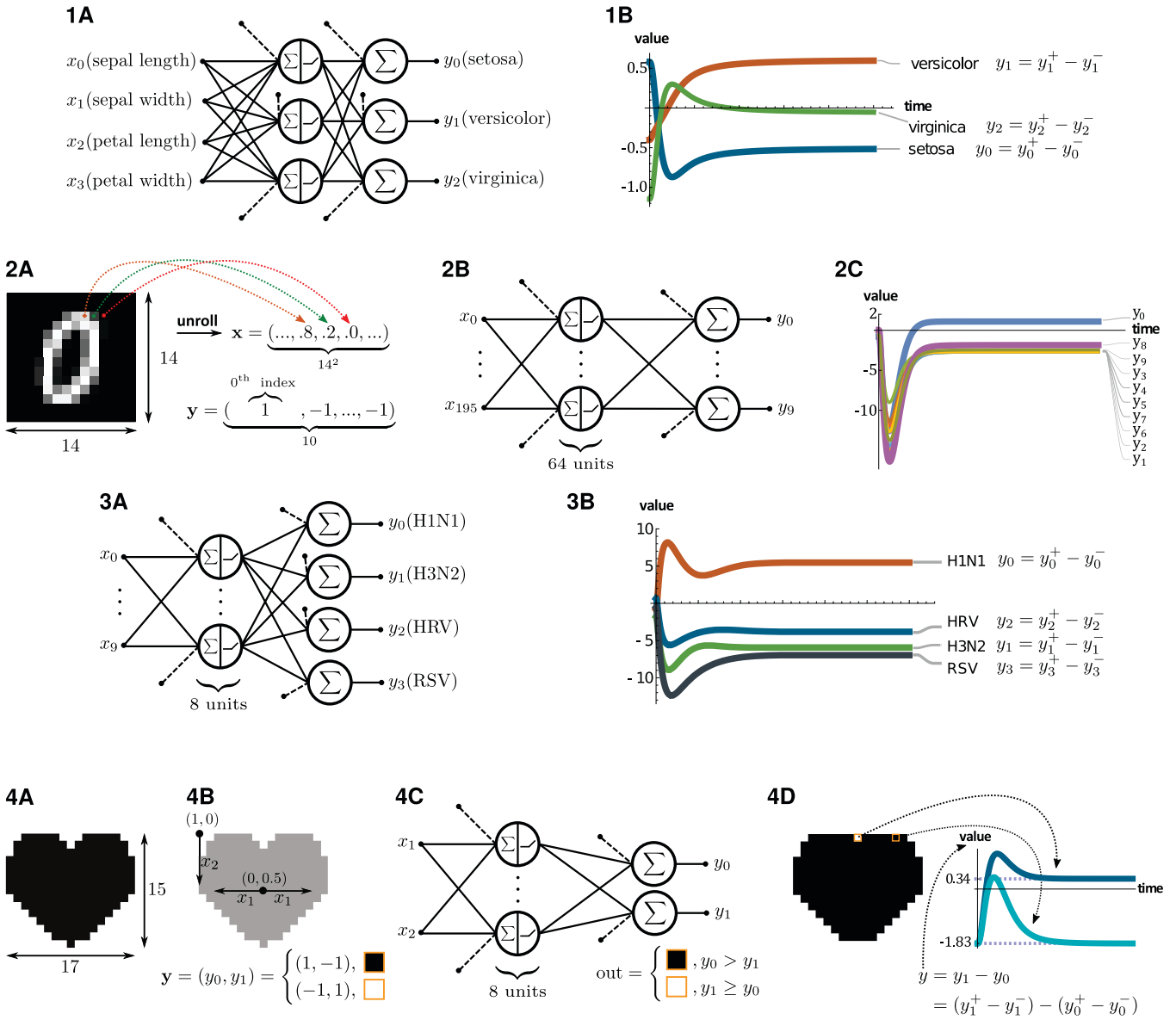


**Fig. 8.** Neural network architecture, input/output encoding, and CRN simulations for different datasets. Subfigures *1A* and *1B* show Iris neural network architecture and the kinetic trajectory of the corresponding CRN for an example input. The classification is "versicolor" because it is the highest (dual-rail) output value. Subfigure *2A* shows an example MNIST input image and its input/output encoding. Each image from the MNIST dataset is unrolled into a vector, and the output label is represented as a 10D vector. Subfigures *2B* and *2C* show MNIST neural network architecture and the kinetic trajectory of the corresponding CRN for the input shown in *2A*, which the network correctly classifies as a zero. Subfigures *3A* and *3B* show virus infection neural network architecture and the kinetic trajectory of the corresponding CRN for an example input. Subfigures *4A* through *4D* show a pattern formation (heart) neural network: *4A* shows the image used to construct the dataset; *4B* shows input and output encoding for a position (pixel) in the input image. An input is encoded using *2D* coordinates: ($x_1$) symmetric horizontal coordinates (starting in the image center) and ($x_2$) vertical coordinates starting from the top left edge of the image. An output, which can be either black or white pixel, is encoded as a *2D* vector as shown in the figure. Subfigure *4C* shows neural network architecture. Subfigure *4D* shows an image learned by the neural network, and the kinetic trajectories of the corresponding CRN for two different input values (positions) corresponding to white and black pixels.

not expected to be dual rail, and one species from each dual-rail pair could be omitted (corresponding reactions removed). In our simulations, however, we used standardized dual-rail signals as inputs to our neural networks which could be positive or negative (*SI Appendix*, section J).

**Convergence Time Analysis.** Chemical implementation of neural networks presents an additional dimension in the form of the kinetics of converging to the output. We focus on two aspects of the kinetics: how computation time changes from input to input, and how it varies with the depth of the network.

For a classifier neural network, the logical output of the corresponding CRN at any given time is the output with the highest (dual-rail) value. As the computation proceeds, the logical output can change over time until eventually stabilizing to the output of the original network. The time to stabilization (stabilization time) captures how soon the readout output is correct.

For the pattern formation (heart) neural network, we found that the stabilization time varied from 0 to 12 units of time across different inputs ($x_1$,$x_2$ coordinates). (Since the bias weights established "black" as the default output, stabilization time can be zero.) Intriguingly, the inputs at the border of the generated shape had higher stabilization time than inputs farther inside or outside the shape (Fig. 9, *Left*). The greater stabilization time could be due to the integration of multiple conflicting signals of similar magnitude at the output nodes, whereas, farther away from the border, one or the other signal dominated.

Across different datasets and neural networks, we observed a correlation between stabilization time and the classifier confidence. The confidence (23) is a measure of how "convinced" the classifier is of its prediction; see *SI Appendix*, section K. However, the fact that the correlation was only moderate suggests that more work is needed to expound the factors contributing to the speed of our CRNs on different inputs.

It is known that the speed of signal propagation in chemical networks can be qualitatively different than in their electronic counterparts. For example, Seelig and Soloveichik (24) analyzed a chemical implementation of a tree of AND gates, finding that the time to reach 90% completion scales quadratically with the depth of the tree. This scaling contrasts with the usual electronic time complexity, which is linear in the depth.

To better understand the speed of signal propagation in BReLU CRNs, we trained two neural networks, each with five hidden layers, on the heart dataset. We observed that, in each network, the time to produce 90% of the output signal for each layer scaled linearly with the depth of the layer (Fig. 9, *Right*). We further trained 10 independent networks on the heart dataset with a varying number of hidden layers (one to five). We similarly observed that the time to produce 90% of the output species in the last layer scaled linearly with the number of layers (*SI Appendix*, Fig. S6, *Left*).

While the time to produce species to 90% of their equilibrium value exhibited linear scaling with the number of layers, we did observe that stabilization time increased superlinearly (*SI Appendix*, Fig. S6, *Right*). The delay in converging to the correct logical output may be due to overall closer signal values in deeper networks that required more time to approach their respective ranked order.

## Related Work

A brief conference version of this work focused on the BReLU network implementation (25). In this full version, we introduce the machinery of noncompetitive CRNs allowing for proofs of correctness, the general construction for RReLU networks, and the inverse construction showing simulation of CRNs by ReLU networks.

Prior work has studied a number of properties of CRNs that arise from stoichiometry alone and are independent of rates (26, 27). In the context of using CRNs to perform computation, computation by stoichiometry (2) was directly motivated by the notion of stable computation in population protocols (11). Other notions of nearly rate-independent computation involved a coarse separation into fast and slow reactions (28).

Recent work took a different but related approach to formalizing and verifying rate independence (7). It considered a broad class of rate laws and identified three easy-to-check conditions that force convergence to the same point under any rate law in this class. Specifically, it showed that it is sufficient for the CRN to be synthesis-free, loop-free, and fork-free. The first condition means that every reaction decreases some species, the second condition is equivalent to our feed-forward condition, and the last is a more restricted version of noncompetition. Notably, the *Theorem* does not require the loop-free condition, allowing us to prove rate-independent convergence of CRNs with cycles such as our rational multiplication CRN shown in Fig. 5. Similarly, the convergence of our rational multiplication CRN construction cannot be proven using the technique of Chen et al. (3), since it does not satisfy their feed-forward definition.

The connection between CRNs and neural networks has a long history. It has been observed that biological regulatory networks may behave in a manner analogous to neural networks. For example, both phosphorylation protein–protein interactions

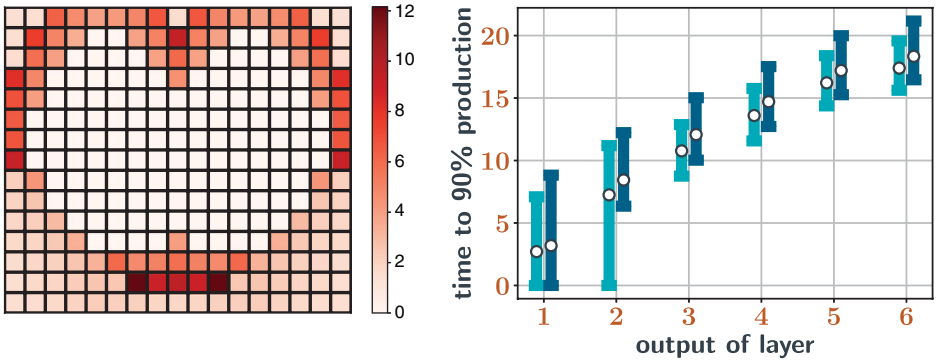


**Fig. 9.** Time analysis for the pattern formation (heart) dataset. (*Left*) Stabilization time for each position in the input pattern for the one-layer neural network (heart) shown in Fig. 8 (subfigures *4A* through *4D*). (*Right*) Time to 90% production of output species of hidden layers (1 through 5) and the output layer (6) of two different neural networks trained on the heart dataset (cyan and blue). The circles represent mean values, while the lower and upper bars represent 10th and 90th percentiles over all inputs and output nodes.

(29, 30) and transcriptional networks (31) can be viewed as performing neural network computation. Hjelmfelt et al. (32) proposed a binary-valued chemical neuron, whose switch-like behavior relies on competition between excitation and inhibition. More recently, Moorman et al. (33) proposed an implementation of ReLU units based on a fast bimolecular sequestration reaction which competes with unimolecular production and degradation reactions. Anderson et al. (34) developed a different mass-action CRN for computing the ReLU and smoothed ReLU function. Recently, Linder et al. (17) proposed a rate-independent CRN implementation of digital binarized neural networks, in which both signals and weights are binary. Further from our work, Poole et al. (35) recently showed a connection between discrete CRNs operating under stochastic kinetics and feedback stochastic neural networks (Boltzmann machines), and simulated their method on the MNIST dataset.

In contrast to prior work on chemical analog neural networks, our implementation relies solely on the stoichiometric exchange of reactants for products, and is thus completely independent of the reaction rates. Our CRN is also significantly more compact, using only a single bimolecular reaction per neuron for BReLU networks, with two species per every connection (without additional species for the neuron itself).

We use neural networks as a way to program chemistry. The programming is done offline in the sense that neural networks are trained in silico. However, there is a body of work on creating chemical systems that are capable of learning in chemistry (36, 37). Although these constructions are much more complex than ours, and arguably difficult to realize, they demonstrate the proof of principle that chemical interactions such as those within a single cell are capable of brain-like behavior.

Besides the above-mentioned theoretical work on chemical neural networks, wet-lab demonstration of synthetic chemical neural computation argues that the theory is not vapid and that neural networks could be realized in chemistry. A chemical linear classifier reading gene expression levels could perform basic disease diagnostics (38). Larger systems based on strand displacement cascades were used to implement Hopfield associative memory (39), and winner-take-all units were used to distinguish digits "6" and "7" in the MNIST dataset (40). Interestingly, the direct strand displacement implementation of a neuron by our construction is significantly simpler (in terms of the number of components needed) than the previous laboratory implementations, arguing for its feasibility.

## Conclusion

While computation in CRNs typically depends on reaction rates, rate-independent information processing occurs in the stoichiometric transformation of reactions for products. In order to better program such computation, we advance noncompetition as a useful property, allowing us to analyze an infinite continuum of possible, highly parallel trajectories via a simple sequential analysis. We further demonstrate embedding complex information processing in such rate-independent CRNs by mimicking neural network computation. For binary weight neural networks, our construction is surprisingly compact in the sense that we use exactly one reaction per ReLU node. This compactness argues that neural networks may be a fitting paradigm for programming rate-independent chemical computation.

As proof of principle, we demonstrate our scheme with numerical simulations of traditional machine learning tasks (Iris and MNIST), as well as tasks better aligned with potential biological applications (virus identification and pattern formation). The last two examples rely on chemically available information for input, and thus argue for the potential biological and medical utility of programming chemical computation via a translation from neural networks.

Neural networks can be evaluated on the speed of the corresponding chemical kinetics, in addition to the typical measures of size and accuracy. How does the speed of chemical convergence vary with the structure of the neural network, and could speed be explicitly optimized in neural network training? Does the speed of convergence of the chemical kinetics relate to nonchemical measures of neural network performance such as confidence or the degree of overfitting? Conversely, it might provide a new measure that might itself be useful for analysis of neural networks outside of chemistry. Future work is needed to better understand the time dimension empirically or theoretically.

Although, in principle, arbitrary CRNs can be implemented using DNA strand displacement reactions, current laboratory demonstrations have been limited to small systems (6), and many challenges remain in constructing large CRNs in the laboratory. Rate-independent CRNs possibly offer an attractive implementation target, due to their absolute robustness to reaction rates.

Only three kinds of computing hardware are currently widespread: electronic computers, living brains, and chemical regulatory networks, the last occurring within every cell in every living organism. Given the society-changing success of electronic computers and the recent neural networks revolution inspired by computation in the brain, it may be argued that chemical computation is the least understood of the three. Upon the refinement of theoretical principles and experimental methods, the impact of chemical computation could be felt in far-reaching ways in synthetic biology, medicine, and other fields. Chemical computation by stoichiometry, and methods of programming and training such computation developed here, provide a distinct approach toward bottom-up engineering of molecular information processing.

**Data Availability.** Data have been deposited in GitHub (https://github.com/marko-vasic/dmp).

**ACKNOWLEDGMENTS.** This work was supported by NSF Grants CCF-1901025 to D.S. and CCF-1718903 to S.K. We thank David Doty and Erik Winfree for essential discussions.

Author affiliations: [a]Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX 78712

1. I. R. Epstein, J. A. Pojman, *An Introduction to Nonlinear Chemical Dynamics: Oscillations, Waves, Patterns, and Chaos* (Oxford University Press, 1998).
2. H. L. Chen, D. Doty, D. Soloveichik, "Rate-independent computation in continuous chemical reaction networks" in *Proceedings of the 5th Conference on Innovations in Theoretical Computer Science*, M. Naor, Ed. (Association for Computing Machinery, 2014), pp. 313–326.
3. H. L. Chen, D. Doty, W. Reeves, D. Soloveichik, Rate-independent computation in continuous chemical reaction networks. arXiv [Preprint] (2021). https://doi.org/10.48550/arXiv.2107.13681 (Accessed 20 May 2022).
4. D. Soloveichik, G. Seelig, E. Winfree, DNA as a universal substrate for chemical kinetics. *Proc. Natl. Acad. Sci. U.S.A.* **107**, 5393–5398 (2010).
5. Y. J. Chen *et al.*, Programmable chemical controllers made from DNA. *Nat. Nanotechnol.* **8**, 755–762 (2013).
6. N. Srinivas, J. Parkin, G. Seelig, E. Winfree, D. Soloveichik, Enzyme-free nucleic acid dynamical systems. *Science* **358**, eaal2052 (2017).
7. E. Degrand, F. Fages, S. Soliman, "Graphical conditions for rate independence in chemical reaction networks" in *International Conference on Computational Methods in Systems Biology*, A. Abate, T. Petrov, V. Wolf, Eds. (Springer, 2020), pp. 61–78.
8. M. Courbariaux, Y. Bengio, J. P. David, "BinaryConnect: Training deep neural networks with binary weights during propagations" in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, R. Garnett, Eds. (Curran Associates, 2015), pp. 3123–3131.

9. J. Santos-Moreno, Y. Schaerli, Using synthetic biology to engineer spatial patterns. *Adv. Biosyst.* **3**, e1800280 (2019).
10. T. Fujii, Y. Rondelez, Predator-prey molecular ecosystems. *ACS Nano* **7**, 27–34 (2013).
11. D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, R. Peralta, Computation in networks of passively mobile finite-state sensors. *Distrib. Comput.* **18**, 235–253 (2006).
12. C. A. Petri, *Communication with Automata* (Rome Air Development Center, 1966).
13. R. M. Karp, R. E. Miller, Parallel program schemata. *J. Comput. Syst. Sci.* **3**, 147–195 (1969).
14. H. L. Chen, D. Doty, D. Soloveichik, Deterministic function computation with chemical reaction networks. *Nat. Comput.* **13**, 517–534 (2014).
15. C. Chalk, N. Kornerup, W. Reeves, D. Soloveichik, Composable rate-independent computation in continuous chemical reaction networks. *IEEE/ACM Trans. Comput. Biol. Bioinformatics* **18**, 250–260 (2021).
16. M. Vasic, D. Soloveichik, S. Khurshid, "CRNs exposed: Systematic exploration of chemical reaction networks" in *26th International Conference on DNA Computing and Molecular Programming*, C. Geary, M. J. Patitz, Eds. (Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020), pp. 4:1–4:25.
17. J. Linder *et al.*, "Robust digital molecular design of binarized neural networks" in *27th International Conference on DNA Computing and Molecular Programming*, M. R. Lakin, P. Šulc, Eds. (Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021), pp. 1:1–1:20.
18. R. A. Fisher, The use of multiple measurements in taxonomic problems. *Ann. Eugen.* **7**, 179–188 (1936).
19. E. Anderson, The species problem in Iris. *Ann. Mo. Bot. Gard.* **23**, 457–509 (1936).
20. Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition. *Proc. IEEE* **86**, 2278–2324 (1998).
21. National Center for Biotechnology Information, Host gene expression signatures of H1N1, H3N2, HRV, RSV virus infection in adults. https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE73072. Accessed 4 February 2020.
22. D. Soloveichik, Mathematica package for working with networks of coupled chemical reactions. http://users.ece.utexas.edu/~soloveichik/crnsimulator.html. Accessed 20 May 2022.
23. C. Guo, G. Pleiss, Y. Sun, K. Q. Weinberger, "On calibration of modern neural networks" in *Proceedings of the 34th International Conference on Machine Learning*, D. Precup, Y. W. Teh, Eds. (PMLR, 2017), pp. 1321–1330.
24. G. Seelig, D. Soloveichik, "Time-complexity of multilayered DNA strand displacement circuits" in *International Workshop on DNA-Based Computers*, R. J. Deaton, A. Suyama, Eds. (Springer, 2009), pp. 144–153.
25. M. Vasic, C. Chalk, S. Khurshid, D. Soloveichik, "Deep Molecular Programming: A natural implementation of binary-weight ReLU neural networks" in *Proceedings of the 37th International Conference on Machine Learning*, H. Daumé III, A. Singh, Eds. (Proceedings of Machine Learning Research, 2020), Vol. 119, pp. 9701–9711.
26. B. L. Clarke, Stoichiometric network analysis. *Cell Biophys.* **12**, 237–253 (1988).
27. M. Feinberg, *Foundations of Chemical Reaction Network Theory* (Springer, 2019).
28. P. Senum, M. Riedel, "Rate-independent constructs for chemical computation" in *Biocomputing 2011*, R. B. Altman, A. K. Dunker, L. Hunter, T. A. Murray, T. E. Klein, Eds. (World Scientific, 2011), pp. 326–337.
29. K. J. Hellingwerf, P. W. Postma, J. Tommassen, H. V. Westerhoff, Signal transduction in bacteria: Phospho-neural network(s) in *Escherichia coli*? *FEMS Microbiol. Rev.* **16**, 309–321 (1995).
30. D. Bray, Protein molecules as computational elements in living cells. *Nature* **376**, 307–312 (1995).
31. N. E. Buchler, U. Gerland, T. Hwa, On schemes of combinatorial transcription logic. *Proc. Natl. Acad. Sci. U.S.A.* **100**, 5136–5141 (2003).
32. A. Hjelmfelt, E. D. Weinberger, J. Ross, Chemical implementation of neural networks and Turing machines. *Proc. Natl. Acad. Sci. U.S.A.* **88**, 10983–10987 (1991).
33. A. Moorman, C. C. Samaniego, C. Maley, R. Weiss, "A dynamical biomolecular neural network" in *58th IEEE Conference on Decision and Control* (Curran Associates, 2019), pp. 1797–1802.
34. D. F. Anderson, B. Joshi, A. Deshpande, On reaction network implementations of neural networks. *J. R. Soc. Interface* **18**, 20210031 (2021).
35. W. Poole *et al.*, "Chemical Boltzmann machines" in *International Conference on DNA-Based Computers*, R. Brijder, L. Qian, Eds. (Springer, 2017), pp. 210–231.
36. H. J. K. Chiang, J. H. R. Jiang, F. Fages, "Reconfigurable neuromorphic computation in biochemical systems" in 2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC) (Institute of Electrical and Electronics Engineers, 2015), pp. 937–940.
37. D. Blount, P. Banda, C. Teuscher, D. Stefanovic, Feedforward chemical neural network: An in silico chemical system that learns xor. *Artif. Life* **23**, 295–317 (2017).
38. R. Lopez, R. Wang, G. Seelig, A molecular multi-gene classifier for disease diagnostics. *Nat. Chem.* **10**, 746–754 (2018).
39. L. Qian, E. Winfree, J. Bruck, Neural network computation with DNA strand displacement cascades. *Nature* **475**, 368–372 (2011).
40. K. M. Cherry, L. Qian, Scaling up molecular pattern recognition with DNA-based winner-take-all neural networks. *Nature* **559**, 370–376 (2018).