Neural Networks 106 (2018) 144–151

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Memristive nanowires exhibit small-world connectivity

Ross D. Pantone [a,*], Jack D. Kendall [a], Juan C. Nino [b]

[a] Rain Neuromorphics, Inc., One Embarcadero Center, Suite #1650, San Francisco, CA 94111, United States
[b] Department of Materials Science and Engineering, University of Florida, 166 Rhines Hall, Gainesville, FL 32611, United States

ARTICLE INFO

Article history:
Received 14 March 2018
Received in revised form 19 June 2018
Accepted 5 July 2018
Available online 17 July 2018

Keywords:
Neuromorphics
Small-world
Network science
Nanowires
Memristors

ABSTRACT

Small-world networks provide an excellent balance of efficiency and robustness that is not available with other network topologies. These characteristics are exhibited in the Memristive Nanowire Neural Network (MN^3), a novel neuromorphic hardware architecture. This architecture is composed of an electrode array connected by stochastically deposited core–shell nanowires. We simulate the stochastic behavior of the nanowires by making various assumptions on their paths. First, we assume that the nanowires follow straight paths. Next, we assume that they follow arc paths with varying radii. Last, we assume that they follow paths generated by pink noise. For each of the three methods, we present a method to find whether a nanowire passes over an electrode, allowing us to represent the architecture as a bipartite graph. We find that the small-worldness coefficient increases logarithmically and is consistently greater than one, which is indicative of a small-world network.

1. Introduction

Neuromorphic hardware architectures, much like biological neural networks, are subject to constraints that are not present in traditional software-based simulations of artificial neural networks (ANNs) (Yu, Zhang, Chen, & Xie, 2018). For example, the weights of a neural network must be physically linked to the two neurons the weights connect. This introduces the wiring cost of a network, which is a measure of how much wiring is needed to connect all the neurons in the network. In both biological neural networks and neuromorphic hardware architectures, this wiring between the neurons consumes the vast majority of the available space (Raj & Chen, 2011). Therefore, minimizing this wiring cost is extremely important.

In biological neural networks, another important parameter is the global efficiency of the network, which is the inverse of the mean shortest path length between two random neurons (Achard & Bullmore, 2007). This value determines how efficiently the network can process and transmit information. For fully connected networks, the value of this parameter is maximal, while for locally connected networks, it is low. Thus, for example, biological brains are the result of a delicate balance between the competing objectives of creating a well-connected network with a high global efficiency, while simultaneously minimizing the amount of wiring needed to connect the network (Achard & Bullmore, 2007).

Similarly, any scalable neuromorphic architecture must balance these two cost functions. In biological neural networks, as well as other complex networks, the optimization of balancing these two objectives (wiring cost and global efficiency) results in a *small-world network* topology. Small-world networks have mostly local connectivity, with a nontrivial number of random, long-range connections added to the network. These small-world networks have been shown to achieve an efficient balance between wiring cost and global efficiency, allowing small-world networks to scale far more efficiently than fully or locally connected networks (Kleinberg, 2000).

In addition to the small-world property, biological neural networks display many other characteristics of complex networks, such as stochasticity, a diverse degree distribution (scale-free structure), and modularity (Holden, 1983; Martinello, Hidalgo, Maritan, & di Santo, 2017; Rodriguez, Izquierdo, & Ahn, 2017).

Despite the advantages of these complex network topologies, to date, most neuromorphic hardware architectures continue to use fully or locally connected crossbar arrays (Schuman, Potok, & Patton, 2017). This is likely due to the simplicity with which these arrays can be fabricated and integrated into conventional hardware. However, there are merits to exploring alternative methods of fabricating more complex network topologies, given the benefits of small-world connectivity, specifically, with respect to the scalability of the networks.

Here, we show that a novel method for fabricating complex networks based on memristive nanowires developed by Kendall & Nino (2015) has an extremely high density of trainable parameters (∼400 million tunable synapses per square centimeter) and more importantly, it exhibits small-world characteristics.

* Corresponding author.
E-mail addresses: ross@rain-neuromorphics.com (R.D. Pantone), jack@rain-neuromorphics.com (J.D. Kendall), jnino@mse.ufl.edu (J.C. Nino).

https://doi.org/10.1016/j.neunet.2018.07.002
0893-6080/

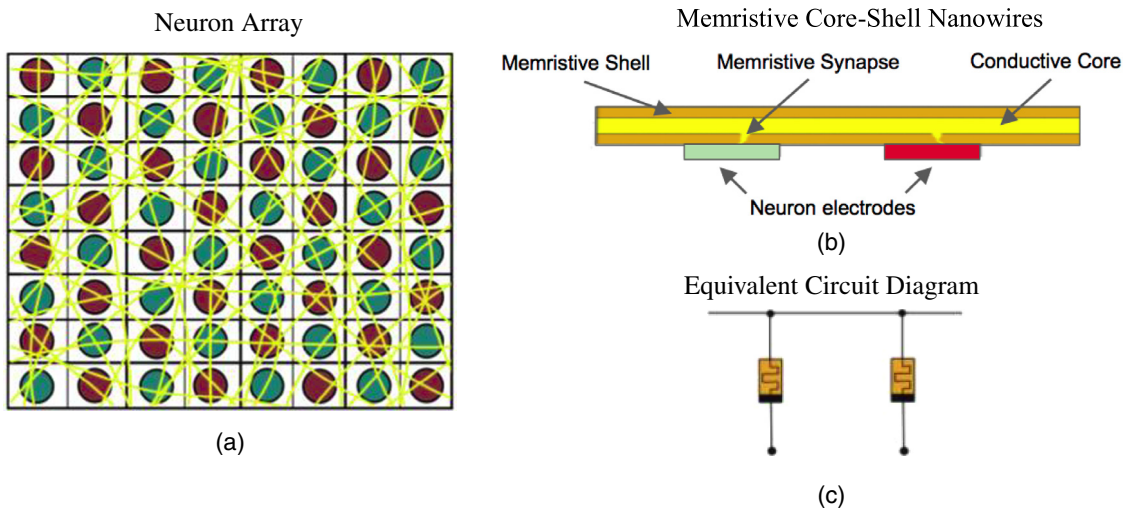

Fig. 1. (a) Each red–green pair corresponds to the input and output of a single neuron, tiled across the entire chip. The MN^3 is connected with a nanowire mesh overlaid on the neuron grid. (b) Each electrode forms a memristive synapse with the neurons below. (c) The equivalent circuit diagram of Fig. 1b. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
Network specifications.

Description	Value
Nanowire diameter	$d = 100$ nm
Neuron width/length (square)	$l = 4\ \mu$m
Neuron spacing	$a = 1\ \mu$m
Nanowire mat thickness	$s = 1\ \mu$m
Wire packing fraction	$p = 0.25$

1.1. Network architecture

The network we describe is based on a network of core–shell memristive nanowires, or nanofibers, with a conductive core and a memristive shell. The architecture, which we dub the *Memristive Nanowire Neural Network*, or MN^3, is shown in Fig. 1. The wires serve as the interconnect layer in an array of CMOS neurons tiled in a square array. Each red–green pair corresponds to a single neuron. It is important to note that unlike conventional architectures, the neurons are not connected in the CMOS layer (Yu, et al., 2018). Instead, a layer of nanowires is deposited on the surface of the silicon, connecting the neurons in a stochastic manner. Metal pillars are grown through the nanowire layer to connect them to the electrodes below. The cores of the nanowires are conductive to allow for signal transmission between neurons, while the shell is made from a memristive material, allowing for the formation of memristive synapses at the interface between each nanowire and neuron.

We can approximate the neuron and synapse densities by using a simple geometric approach. We use the assumptions in Table 1 on the achievable feature sizes in the network. Note that the wire packing fraction is the density of wires compared to that in a close-packed structure (Batch, Cumiskey, & Macosko, 2002).

From l and a, we can determine how many neurons N will fit in a 1 cm^2 area:

$$N = 1/(l + a)^2 = 1/(0.0004 + 0.0001)^2 = 4 \times 10^6. \quad (1)$$

We can now determine how many wires contact each electrode if the wires are close-packed, w_c, and then multiply by the packing fraction to get the average number of wires contacting each electrode, w. We have

$$w_c = ls/d^2 = (0.0004)(0.0001)/(0.00001)^2 = 400. \quad (2)$$

Assuming a packing fraction for the wires of 0.25, i.e. only 1/4 of the maximum packing density, we arrive at the wire density per electrode w. This value can be increased at the expense of neuron density. We have

$$w = pw_c = (0.25)(400) = 100. \quad (3)$$

Now we can calculate the total number of synapses S in the network. Since a synapse is formed at the intersection of each wire with each electrode, the total number of synapses is equal to the number of electrodes multiplied by the average number of wires per electrode. We have

$$S = Nw \approx (4 \times 10^6)(100) = 4 \times 10^8. \quad (4)$$

The density of neurons in the MN^3 (4×10^6 neurons per cm^2) is several orders of magnitude higher than state-of-the-art values reported in the literature, including Intel's Loihi (218,400 neurons per cm^2) (Davies et al., 2018), IBM's TrueNorth (~12,157 neurons per cm^2) (Merolla et al., 2014), and Stanford's Neurogrid (~39,620 neurons per cm^2) (Benjamin et al., 2014). Two factors contribute to this sharp increase in density: the integration of the synapses and the wiring into a single compound structure, and the offloading of the synapses and the wiring from the surface of the CMOS to a sparsely connected nanowire layer.

By removing the wiring and synapses from the CMOS layer, the neurons can be close-packed as tightly as possible, drastically increasing neuron density. Similarly, the stacked mat of nanowires (connected to the neuron electrodes through vertically grown metal pillars) has a high density of wires, which are capable of connecting neurons across long distances, resulting in a high synapse density. The number of overlap of nanowires is estimated to be 10 (so the nanowires are stacked approximately 10 high). This gives enough spacing so that metal deposition techniques, such as sputtering, can fully penetrate the mesh. Simulations were conducted to verify that the presented wire density is compatible with this fabrication process. Regarding polarities, there are no inhibitory synaptic polarities as all conductances are positive. The neurons (external nodes) are used as hyperbolic tangent units, so they can take both positive and negative values. The memristor polarities themselves are all aligned towards the nanowire nodes (based on the wire core acting as the bottom electrode).

The sparsity of the resulting connection layer is important in reducing the total amount of wiring needed to connect the network.

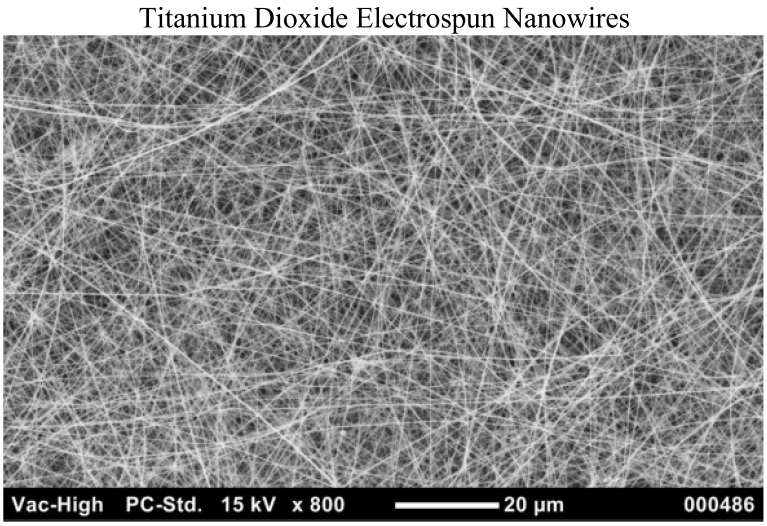

Fig. 2. Physical titanium dioxide electrospun nanowires at a 20 µm scale displaying a high level of stochastic behavior.

Here, we show that the topology of the connections in a random nanowire network displays small-world characteristics. We also show that the network has a bipartite structure and supports a variable degree diversity, with the potential to allow for scale-free network topologies.

1.2. Bipartite connectivity

We first observe that the nodes of the MN^3 can be partitioned into two groups. The first group of nodes corresponds to the neuron electrodes. These are the external electrodes of the network, as they can be connected to an external circuit (i.e. the neurons). The second group of nodes corresponds to the cores of the nanowires. Since the nanowire cores are designed and manufactured with a highly conductive core material, we can treat them as single nodes.

Next, we observe that the interface resistance between two nanowire shells will be several orders of magnitude greater than the interface resistance between a single nanowire and an electrode. This is due to a combination of several factors. First, the presence of a double Schottky barrier between the two nanowire cores means the current is rectified in both directions between the two nanowires (Jagadesh, 2005). Second, the distance between two nanowires (through both memristive shells) is twice as long as between a single nanowire and external electrode. Finally, the contact area between two nanowires is much smaller than the contact area between a nanowire and external electrode.

This allows us to conclude that, to a good approximation, the nanowires are effectively insulated from one another. The consequence of this result is that the MN^3 forms a bipartite memristive network, with the external electrodes only connected to each other via the nanowire cores, and the nanowire cores only connected to each other through the external electrodes.

2. Methods

We will refer to our model that comprises electrospun nanowires (wires for short) and electrodes arranged on a lattice graph as the *geometric model*. See Figs. 1 and 3–5 for reference. We seek to simulate the stochastic connectivity of these wires. Prior to doing so, we must first simulate the geometric model sufficiently. Above all else, visual resemblance was the figure of merit in the simulations. Fig. 2 of Kendall and Nino (2015) is one of many images of actual nanowires used as a blueprint.

The simulations took place over several iterations, increasing the realism each time. We did this for greater ease in development as well as the possibility that a simpler model may suffice. The implications for the latter are significant as generating a less rigorous model may require a lower computational complexity.

We define r_e as the electrode radius and a as the distance between electrodes directly vertical or horizontal to one another. Both values are fixed at $r_e = 0.4$ and $a = 1$ for the simulations. Further, n_e is the number of electrodes and n_w is the number of wires. Due to the grid structure of the geometric model, n_e is a perfect square. We treat n_e as an independent variable and assign n_w by $n_w = \lambda\sqrt{n_e}$ where λ is the density constant. Specifically, we use $\lambda = 30$. The dimensions of the grid are given by $l \times l$, where $l = a(1 + \sqrt{n_e})$.

2.1. Straight wire model

Initially, wires were assumed to be straight lines. Informally, wires go from one randomly selected side of the grid at a randomly picked location to another side with the same qualifications. Fig. 3 is a visualization on a 3×3 grid of electrodes (meaning nine total electrodes).

For each iteration, we must determine whether a wire passes through an electrode—that is, whether they are connected. While this is normally easily accomplished visually, it quickly becomes infeasible for large numbers of wires and electrodes. Hence, we must derive a function that outputs `TRUE` if a wire passes through an electrode and `FALSE` if it does not.

We must formally define how the wires are laid. This provides a systematic approach of recreating the model. Let Q be a random variable following the discrete uniform distribution over the set $\{0, 1, 2, 3\}$. Pick unique $q_1, q_2 \in Q$. Let T be a random variable following the continuous uniform distribution over the set $[0, l]$. Pick $x_1, y_1, x_2, y_2 \in T$. Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$, with q_1 and q_2 corresponding to P_1 and P_2, respectively. Then, reassign P_1 and P_2 according to Table 2. For example, let $l = 4$. Recall that q_1 and q_2 will not be equal by our definition. Assume that we pick $q_1 = 1$, $q_2 = 3$, $x_1 = 3.8234$, $y_1 = 1.2818$, $x_2 = 2.7613$, and $y_2 = 3.3237$. Hence, we have $P_1 = (3.8234, 1.2818)$ and $P_2 = (2.7613, 3.3237)$. Following the reassignment table, we adjust the two points to $P_1 = (0, 1.2818)$ and $P_2 = (4, 3.3237)$.

Define the straight wire w_s as the line through P_1 and P_2. For an electrode centered at $P_e = (x_e, y_e)$ on the grid, the minimum

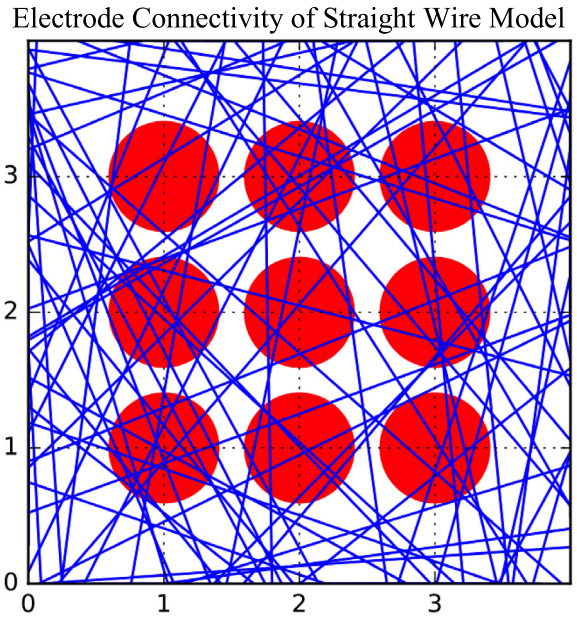

Fig. 3. Straight wire model with $n_e = 9$ electrodes (red) with radii $r_e = 0.4$ and separation $a = 1$ and $n_w = \lambda\sqrt{n_e} = 30\sqrt{9} = 90$ wires (blue). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

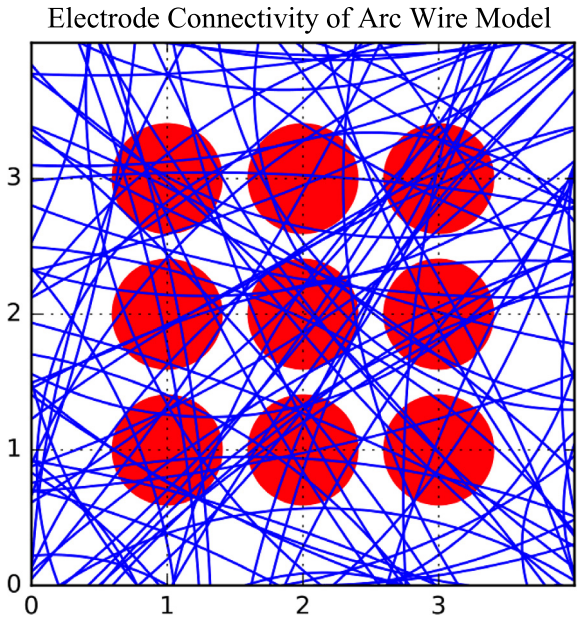

Fig. 4. Arc wire model with $n_e = 9$ electrodes (red) with radii $r_e = 0.4$ and separation $a = 1$ and $n_w = \lambda\sqrt{n_e} = 30\sqrt{9} = 90$ wires (blue). Note the increased realism.

Table 2
Endpoint reassignment values.

Q	Reassignment value
0	$y = 0$
1	$x = 0$
2	$y = l$
3	$x = l$

distance D_s between w_s and P_e is given by

$$D_s = \frac{|x_e(y_2 - y_1) - y_e(x_2 - x_1) + x_2y_1 - y_2x_1|}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}. \tag{5}$$

See Spain (2007) for a simple proof of (5). Then, for electrode radius r_e, we simply return the Boolean value `D_s <= r_e`.

2.2. Arc wire model

Fig. 2 clearly shows that the actual wires are far from perfectly linear. So, we then assumed that wire paths resemble circular arcs with random curvature. Fig. 4 shows a corresponding visualization on a 3 ×3 grid of electrodes.

Again, we seek an efficient way to test whether a wire passes through an electrode. We must alter the method presented in the previous subsection. First, we must find the minimum distance D_c between a point $P = (x_0, y_0)$ and a circle with radius R defined by $x^2 + y^2 = R^2$. To do this, draw a line from the origin to P. The line intersects the circle at the point closest to P. Hence, the minimum distance between the circle and P is given by the difference of the radius of the circle and the distance between P from the origin. Symbolically, we have $D_c = |\sqrt{x_0^2 + y_0^2} - R|$. More generally, if the circle is centered at (h, k), we have

$$D = \left|\sqrt{(x_0 - h)^2 + (y_0 - k)^2} - R\right| \tag{6}$$

We pick points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ according to the algorithm described in Section 2.1. However, q_1 and q_2 are not unique. This allows an arc to start and end on the same grid side, which occurs frequently in the nanowire fabrication process. This was not possible for the straight wire model. Their Euclidean distance in the two-dimensional space is $d(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. Let A be a continuous random variable on $[1/2 * d(P_1, P_2), U]$, where U is an arbitrary upper limit. Define r_w as the radius of the wire's arc such that $r_w \in A$. As U approaches infinity, the model tends to the former model. For the calculations presented, $U = 3d(P_1, P_2)$ was used. Suppose an arc-shape wire w_a is defined by the arc through P_1 and P_2 with radius r_w and center $P_c = (x_c, y_c)$. Note that P_c is found from P_1, P_2, and r_w by writing $(x_1 - x_c)^2 + (y_1 - y_c)^2 = {r_w}^2$ and $(x_2 - x_c)^2 + (y_2 - y_c)^2 = {r_w}^2$ and solving for x_c and y_c. This will give

$$x_c = x_1 + x_a \pm \frac{by_a}{a}, \tag{7}$$

$$y_c = y_1 + y_a \mp \frac{bx_a}{a}, \tag{8}$$

where $x_a = 1/2 * (x_2 - x_1)$, $y_a = 1/2 * (y_2 - y_1)$, $a = \sqrt{{x_a}^2 + {y_a}^2}$, and $b = \sqrt{{r_w}^2 - a^2}$. This clearly produces two possible centers. Since both fit the aforementioned criteria for the model, we may randomly pick one. We do not use the other potential arc as it would give rise to an unwanted correlation and lessen the overall stochasticity.

Suppose an electrode is centered at $P_e = (x_e, y_e)$. Let L be the line formed by points P_c and P_e. Let θ_s and θ_f be the starting and finishing angles of the arc with respect to the x-axis, respectively. Let θ be the angle formed by L with respect to the x-axis. The minimum distance D_a between w_a and P_e is given by

$$D_a = \begin{cases} |d(P_c, P_e) - r_w| & \theta_s \le \theta \le \theta_f \\ \min(d(P_c, P_1), d(P_c, P_2)) & otherwise \end{cases} \tag{9}$$

This is easily seen when breaking down the cases. If L bisects the arc, then we simply apply (6). If it does not, then the only possible points are the arc's endpoints. We choose the one that gives the smaller distance. The polar representation of a circle formed by fully extending its arc is given by $(x_c + r_w\cos(\psi), y_c + r_w \sin(\psi))$, where $\psi \in \mathbb{R}$. To find θ_s, temporarily consider $\theta_{s,x} \in [0, \pi)$ and $\theta_{s,y} \in$

Table 3
Angle values.

$\theta_{s,x} \in$	$\theta_{s,y} \in$	$\theta_s =$
$[0, \pi/2)$	$[0, \pi/2)$	$\theta_{s,x}$
$[0, \pi/2)$	$[-\pi/2, 0)$	$2\pi + \theta_{s,y}$
$[\pi/2, \pi)$	$[0, \pi/2)$	$\theta_{s,x}$
$[\pi/2, \pi)$	$[-\pi/2, 0)$	π - $\theta_{s,y}$

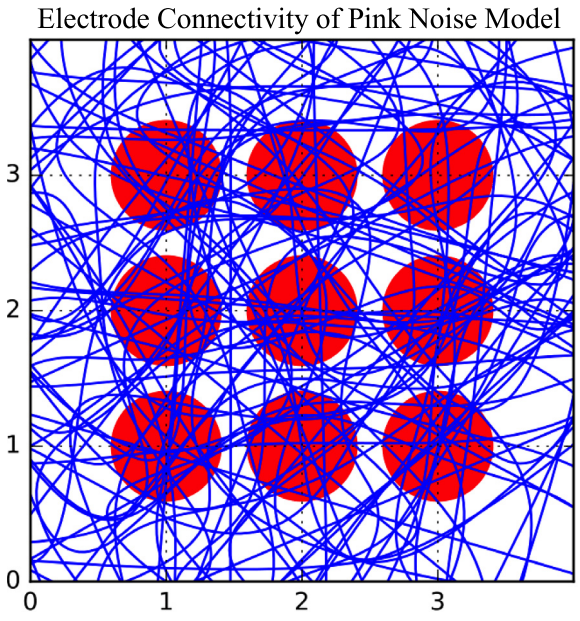

Fig. 5. Pink noise wire model with $n_e = 9$ electrodes (red) with radii $r_e = 0.4$ and separation $a = 1$ and $n_w = \lambda\sqrt{n_e} = 30\sqrt{9} = 90$ wires (blue). Note that the wires in this model are longer than the previous two. This makes it appear that there are more wires in this model than the previous two, but that is not the case.

$[-\pi/2, \pi/2)$. Using the circle's polar form, we have

$$x_1 = x_c + r_w \cos(\theta_{s,x}), \tag{10}$$

$$y_1 = y_c + r_w \sin(\theta_{s,y}). \tag{11}$$

Solving for $\theta_{s,x}$ and $\theta_{s,y}$ yields

$$\theta_{s,x} = \arccos\left(\frac{x_1 - x_c}{r_w}\right), \tag{12}$$

$$\theta_{s,y} = \arccos\left(\frac{y_1 - y_c}{r_w}\right). \tag{13}$$

Use Table 3 to find the exact value of θ_s. Repeat this process to find θ_f. If $\theta_s > \theta_f$, swap the two values. To find θ, use the polar form $(x_c + d(P_c, P_e)\cos(\psi), y_c + d(P_c, P_e)\sin(\psi))$ and follow the previous approach. Finally, for electrode radius r_e, we return the Boolean value `D_a <= r_e`.

2.3. Pink noise model

Last, we generated 100 points using a pink noise, or $1/f$ noise, simulation algorithm. This ensures that the value of a random variable is correlated with random variables near it (Zanella, 2006). Fig. 5 presents a visualization on a 3×3 grid of electrodes. We use the method presented below to simulate pink noise. It is a simplification of the methods presented in Timmer and Koenig (1995). Strictly speaking, this method approximates pink noise, but for large sample sizes, this point is negligible.

First, we denote n as the number of points and T as the length of the sequence. Specifically, $n = T = 100$ was used. This provides the spectral space for wave numbers $-k_{\max}$ to $k_{\max}$, where $k_{\max} = n$, and produces frequencies $f_k = kT/(2\pi)$. Define the magnitude of the spectral coefficients as $C_k = 1/|f_k|$ for non-zero values of k and $C_0 = 0$. Each spectral coefficient is given a random phase. We use symmetry in the exponents to prevent complex results. That is, the sum of the imaginary components of C_k and C_{-k} is zero. Let Φ be a random variable with a continuous uniform distribution on $[0, 2\pi)$. For $k = 1, \ldots, k_{\max}$, let $\varphi_k \in \Phi$. Set $C_k = C_k \exp\{i\varphi_k\}$ and $C_{-k} = C_{-k} \exp\{-i\varphi_k\}$. Then, find the inverse Fourier transform of the spectral coefficients, $Y = \{y_1, y_2, \ldots, y_{1+2k\max}\} = \mathfrak{F}^{-1}(\{C_k\})$.

We will leverage this method to simulate the wires. We scale Y so that the minimum value is 0 and the maximum value is l. For index i, update each value element-wise using

$$y_i \leftarrow \frac{l(y_i - \min\{Y\})}{\max\{Y\} - \min\{Y\}}. \tag{14}$$

Let R be a random variable following a continuous uniform distribution on [0,1]. For each $y_i \in Y$, pick $r_i \in R$ and update each value by $y_i \leftarrow y_i {r_i}^3$. Initially, multiplying y_i by r_i was used, but ${r_i}^3$ produced results more visually similar to the actual nanowires. Next, we randomly shifted all the points vertically such that all points still lay in the grid. Let V be a random variable on the continuous uniform distribution on $[-\min\{Y\}, l - \max\{Y\}]$. For each $y_i \in Y$, pick $v_i \in V$ and update each value by $y_i \leftarrow y_i + v_i$. Let $X = \{x_1, x_2, \ldots, x_{1+2k\max}\} = \{1, 2, \ldots, 1 + 2k_{\max}\}$. Scale X to $[0, l]$ following (14). Let $C = \{(x_1, y_1), (x_2, y_2), \ldots, (x_{1+2k\max}, y_{1+2k\max})\}$. The set of coordinates were then fed through a Gaussian kernel smoother using SciPy's Gaussian filter function (Van derWalt, Colbert, & Varoquaux, 2011), and a quintic polynomial was fitted on top. This last step makes the upcoming distance formula much more derivable. The choice of a quintic polynomial was made strictly from a visual standpoint. Any degree lower did not accurately capture the random paths the wires take, and any degree higher presented too much chaos in the wires' movements. We then multiply our function by a rotation matrix to rotate it around an arbitrary point by a random θ. We chose to rotate about $(l/2, l/2)$.

More formally, let $x \in \mathbb{R}$ and $f(x)$ be a polynomial of degree n. Let W be a random variable with a continuous uniform distribution on $[0, 2\pi)$. Rotating $f(x)$ about point (x_r, y_r) by $\theta \in W$ can be represented as

$$\begin{aligned}\begin{bmatrix} x' \\ y' \end{bmatrix} &= \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x - x_r \\ y - y_r \end{bmatrix} + \begin{bmatrix} x_r \\ y_r \end{bmatrix} \\ &= \begin{bmatrix} (x - x_r)\cos\theta - (f(x) - y_r)\sin\theta + x_r \\ (x - x_r)\cos\theta + (f(x) - y_r)\sin\theta + y_r \end{bmatrix}.\end{aligned} \tag{15}$$

The derivation of the rotation matrix can be found in Familton (2015). We will now derive a general equation for the minimum distance between a point and a polynomial. We define D as the distance between the point and polynomial f. We have $D = \sqrt{(x - x_0)^2 + (f(x) - y_0)^2}$. We seek the minimum of D. Accordingly, we have $dD/dx = 1/2 * (2(x - x_0) + 2f'(x)(f(x) - y_0))((x - x_0)^2 + (f(x) - y_0)^2)^{(-1/2)}$. The critical points are given by solving for the roots of the equation $0 = x - x_0 + f'(x)(f(x) - y_0)$. Plugging these points into D and finding the minimum output value will provide the minimum distance, D_p.

We seek the minimum distance from the parametric equation given above to center of the electrode $P_e = (x_e, y_e)$. Per the previous derivation, this amounts to finding the roots of an equation of the form

$$g(x) = x + \alpha + f'(x)(f(x) + \beta), \tag{16}$$

where function g is a polynomial and $\alpha = (x_r - x_e)\cos(\theta) + (y_r - y_e)\sin(\theta) - x_r$ and $\beta = (y_r - y_e)\cos(\theta) + (x_r - x_e)\sin(\theta) - y_r$.

We then solve for the real roots using a computer algebra package, such as NumPy's Polynomial module (Van der Walt et

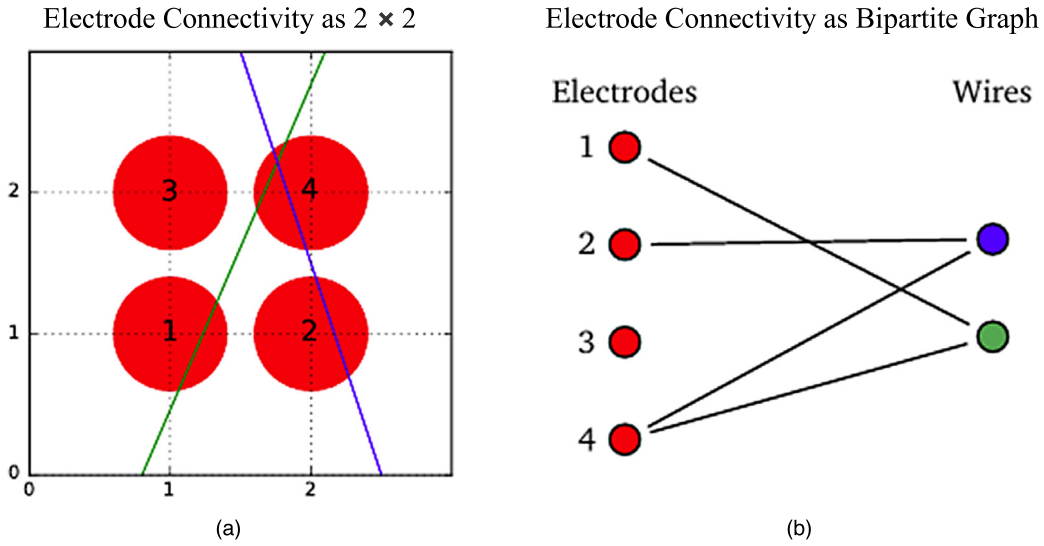

Fig. 6. (a) We present a simple straight wire example with $n_e = 4$. We pick a small number of wires, $n_w = 2$, strictly for demonstration purposes. The wire colors and electrode numbers are also used for demonstration purposes. (b) This is corresponding bipartite representation of (a). Electrodes 1 and 4 map to the green wire as they pass over it in (a), and electrodes 2 and 4 map to the blue wire as they pass over it in (a).

al., 2011). This gives several coordinates. We plug these values into (15) to find the corresponding rotated points. Then, we plug these candidate points into the standard distance formula and call the minimum value D_p. Just as we did above, we return `D_p <= r_e`.

3. Analysis

3.1. Bipartite graphs

As previously mentioned, the geometric model can be represented as a bipartite graph, which allows us to calculate various graph metrics. We will use the structure displayed in Figs. 6 and 7 as an example. Doing so enables us to write the graph as an adjacency matrix $\boldsymbol{A}$. We have

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}. \tag{17}$$

Note that, because our graph is bipartite, we can construct the adjacency matrix from its sub-matrix on rows $\{1, 2, \ldots, n_e\}$ and columns $\{n_e + 1, \ldots, n_e + n_w\}$. This is because none of the vertices in the two groups are interconnected and the graph is undirected, implying symmetry. For this example, the sub-matrix $\boldsymbol{\omega}$ is

$$\boldsymbol{\omega} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}. \tag{18}$$

The values in rows $\{1, 2, \ldots, n_e\}$ and columns $\{1, 2, \ldots, n_e\}$ equal 0, as do the values in rows $\{n_e + 1, \ldots, n_e + n_w\}$ and columns $\{n_e + 1, \ldots, n_e + n_w\}$. The values in rows $\{n_e + 1, \ldots, n_e + n_w\}$ and columns $\{1, 2, \ldots, n_e\}$ equal $\boldsymbol{\omega}^T$. This is due to the unique structure of bipartite graphs. Hence, $\boldsymbol{A}$ can be coded solely by $\boldsymbol{\omega}$. This may be fruitful for memory storage purposes.

Representing our graph as an adjacency matrix allows us to apply a barrage of graph algorithms. We will use algorithms for the shortest path length and square clustering coefficient in the next section.

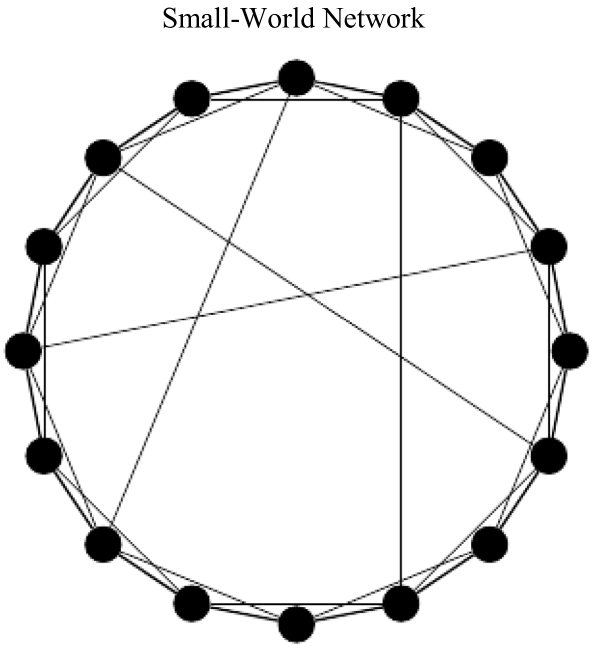

Fig. 7. This is a conventional small-world network; the nodes are not densely connected but each node can be reached in a small number of steps.

3.2. Small-world networks

With the network model in place, it is interesting to calculate its average clustering coefficient and average shortest path length to gain a better understanding of its overall structure when compared to a random graph. This can be quantified by the small-world coefficient. In a small-world network, most nodes are not neighbors but can be traveled to in a small number of steps (Newman, 2010). Fig. 7, which appears to be markedly different from the geometric model, shows a typical small-world network. We will investigate the geometric model for small-worldness properties.

For our geometric model, let L and C be the average shortest path length between any two nodes and our square clustering coefficient, respectively. For a randomly generated bipartite graph

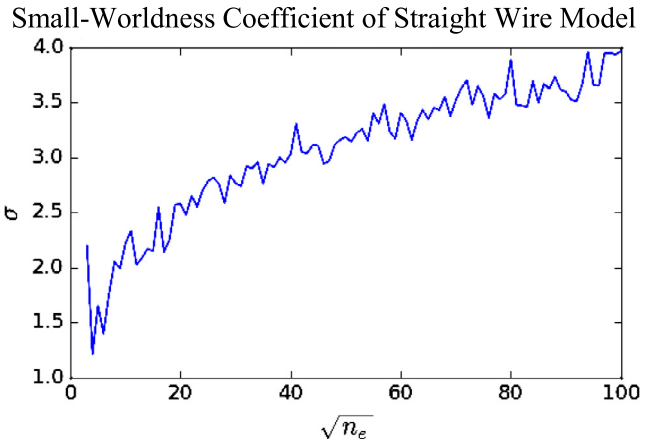

Fig. 8. For the straight wire model in Section 2.1, using Eq. (20), we see that small-worldness is achieved for every value tested. As the number of electrodes increases, the small-world coefficient increases logarithmically.

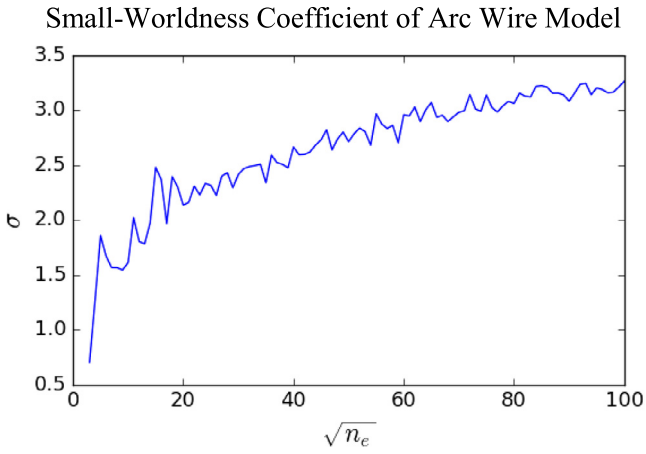

Fig. 9. For the arc wire model in Section 2.2, small-worldness is achieved for every value tested with the exception of the smallest case, $n_e = 9$. The logarithmic behavior also holds for this model.

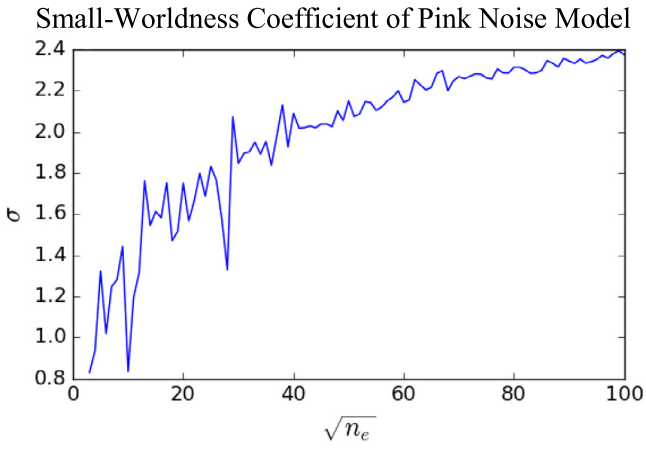

Fig. 10. For the pink wire model in Section 2.3, small-worldness is achieved for all values except $n_e = 3$, 4, and 10. There is a considerable increase in noise in this model, and that is expected with the increase in degrees of freedom. The logarithmic behavior persists.

with the same number of electrodes, wires, and connections, let L_r and C_r be the average shortest path length and square clustering coefficient, respectively. The conventional clustering coefficient is not applicable to bipartite networks; hence, the square clustering coefficient was used. Specifically, for a given node v, we have

$$C_4(v) = \frac{\sum_{u=1}^{k_v}\sum_{w=u+1}^{k_v} q_v(u,w)}{\sum_{u=1}^{k_v}\sum_{w=u+1}^{k_v}[a_v(u,w)+q_v(u,w)]}, \tag{19}$$

where $q_v(u, w)$ is the number of common neighbors between u and w other than v, and $a_v(u, w) = (k_u - (1 + q_v(u, w) + \theta_{uv}))(k_v - (1 + q_v(u, w) + \theta_{uv}))$, where $\theta_{uw} = 1$ if u and w are connected and 0 otherwise (Lind, González, & Herrmann, 2005).

We sought the average shortest path between electrodes and the clustering coefficient of electrodes. Hence, our focus was only on rows $\{1, 2, \ldots, n_e\}$ and columns $\{1, 2, \ldots, n_e\}$ in the adjacency matrix. There are many shortest path algorithms available (Madkour, Aref, Rehman, Rahman, & Basalamah, 2017). Since our graph is sparse and unweighted, the most economical choice is to use breadth-first search over the n_e electrodes for E connections, which will provide time complexity $O(n_e(n_e + n_w + E)) \approx O(n_e(n_e + n_w)) \approx O(n_e{}^2)$ (Cormen, Leiserson, Rivest, & Stein, 2001). The average of the lengths is taken, and this gives L. Similarly, the square clustering coefficient is taken over $v = \{1, 2, \ldots, n_e\}$ and then averaged. This gives C.

The small-worldness coefficient is defined as

$$\sigma = \frac{\frac{C}{C_r}}{\frac{L}{L_r}}. \tag{20}$$

If $\sigma > 1$, the network is small-world (Humpries & Gurney, 2008). Typically, $L \approx L_r$ and $C > C_r$. Note that it is assumed that both the geometric model and the random model are connected. A given geometric model is only used if it is connected, and Saltykov's (1995) results suggest that the random graph is connected with extremely high probability (Saltykov, 1995).

3.3. Small-Worldness of the MN^3

These results were obtained by generating the three aforementioned geometric models and calculating their small-world coefficients for varying grid size while scaling the number of wires by $n_w = 30\sqrt{n_e}$ and fixing electrode radius r_e and distance between electrodes a (see Figs. 8–10).

Notice that all of these models follow the same basic logarithmic curve. This is expected. We clearly have an increase in wire length, as the models get more complex. This will result in a shorter average shortest path length as more connections are added. However, this will cause the clustering coefficient to decrease as each wire affects more regions. This is why all the models follow the same basic curve. However, their actual small-worldness coefficients clearly differ. The straight wire model and arc wire model roughly possess the same amount of noise, while the pink noise model is slightly more chaotic.

3.4. Diverse degree distribution by varying radii

Our final claim is that the degree distribution can be manipulated by altering the radii of the electrodes. To test this, we arbitrarily fix n_e and a. We find n_w by n_e. For varying radii, we count the number of wires passing through an arbitrary electrode.

Fig. 11 clearly shows an increasing linear relationship between an electrode's radius and the number of wires passing through the electrode. This indicates that the degree distribution is a controllable parameter.

4. Conclusion

We have identified several important characteristics concerning the connection topology of a novel memristive neuromorphic

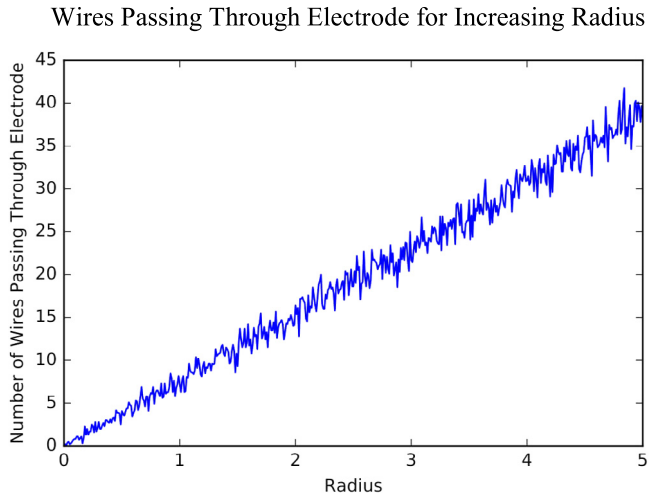

Fig. 11. The degree of an electrode may be manipulated by adjusting its radius. Here, we arbitrarily fix $n_e = 25$, $a = 5$, and $n_w = \lambda\sqrt{n_e} = 30\sqrt{25} = 150$. For each radius in the set $\{0.01, 0.02, \ldots, 5.0\}$, we count the number of wires passing through the center electrode.

architecture, the MN^3. First, we have shown that the neuron density possible with an architecture based on the MN^3 is several orders of magnitude higher than current state-of-the-art neuromorphic devices, with synapse densities approaching a billion synapses per square centimeter. Next, we argue based on geometric and materials science properties that the network possesses an intrinsic bipartite structure between the two classes of nodes: external electrodes and nanowire cores. Finally, we investigate the clustering coefficient and mean shortest path length of a set of geometric models of the MN^3 network.

We determine that MN^3 networks display small-world characteristics, with a mean shortest path that scales logarithmically in the size of the network. While the small-world coefficient σ also scales logarithmically, we find that this is due to a logarithmic increase in the clustering coefficient of the network. This could be improved, for example, by adding nearest-neighbor connections in the CMOS layer. The ability to control the degree distribution of the nodes in the network allows for the construction of scale-free networks.

Acknowledgments

JCN acknowledges the support of the National Science Foundation under ECCS Award Number 1709641. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Achard, S., & Bullmore, E. (2007). Efficiency and cost of economical brain functional networks. *PLoS Computational Biology*, *3*(2), e17.
Batch, G. L., Cumiskey, S., & Macosko, C. W. (2002). Compaction of fiber reinforcements. *Polymer Composites*, *23*, 307–318.
Benjamin, B. V., et al. (2014). Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations. *Proceedings of the IEEE*, *102*(5), 699–716.
Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2001). *Introduction To Algorithms*. (pp. 594–597). MIT Press and McGraw-Hill.
Davies, Mike, et al. (2018). Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, *38*, 82–99.
Familton, J. C. (2015). Quaternions: a history of complex noncommutative rotation groups in theoretical physics (Ph.D. thesis). arXiv:1504.04885 [physics.hist-ph].
Holden, A. V. (1983). Stochastic processes in neurophysiology: Transformation from point to continuous processes. *Bulletin of Mathematical Biology*, *45*, 443–465.
Humpries, M. D., & Gurney, K. (2008). Network small-world-ness: A quantitative method for determining canonical network equivalence. *PLoS ONE*, *3*.
Jagadesh, M. (2005). ShOC rectifier: A new metal-semiconductor device with excellent forward and reverse characteristics. *IEEE Transactions on Electron Devices*, *52*, 130–132.
Kendall, J. D., & Nino, J. C. (2015). U.S. Application No. WO2015195365A1. Washington, DC: U.S. Patent and Trademark Office.
Kleinberg, J. (2000). The small-world phenomenon: An algorithmic perspective. pp. 163-170.
Lind, P. G., González, M. C., & Herrmann, H. J. (2005). Cycles and clustering in bipartite networks. *Physical Review E*, *72*, 056127.
Madkour, A., Aref, W. G., Rehman, F. U., Rahman, M. A., & Basalamah, S. (2017). A Survey of Shortest-Path Algorithms. arXiv:1705.02044 [cs.DS].
Martinello, M., Hidalgo, J., Maritan, A., & di Santo, S. (2017). Neutral theory and scale-free neural dynamics. *Physical Review X*, *7*, 041071.
Merolla, Paul A., et al. (2014). A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, *345*, 668–673.
Newman, M. (2010). *Networks: An introduction*. (pp. 54–58). Oxford University Press.
Raj, A., & Chen, Y. (2011). The wiring economy principle: Connectivity determines anatomy in the human brain. *PLoS ONE*, *6*, 1–11.
Rodriguez, N., Izquierdo, E., & Ahn, Y. Y. (2017). Optimal modularity and memory capacity of neural networks. arXiv:1706.06511 [cs.NE].
Saltykov, A. I. (1995). The number of components in a random bipartite graph. *Discrete Mathematics and Applications*, *7*(4), 86–94.
Schuman, C. D., Potok, T. E., & Patton, R. M. (2017). A Survey of Neuromorphic Computing and Neural Networks in Hardware. arXiv:1705.06963 [cs.NE].
Spain, B. (2007). *Analytical conics*. (p. 5). Mineola (New York): Dover Publications.
Timmer, J., & Koenig, M. (1995). On generating power law noise. *Astronomy and Astrophysics*, *300*, 707.
Van der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The NumPy array: A structure for efficient numerical computation. *Computational Science & Engineering*, *13*, 22–30.
Yu, J., Zhang, Y., Chen, W., & Xie, Y. (2018). Bridging the gap between neural networks and neuromorphic hardware with A neural network compiler. In *Proceedings of the twenty-third international conference on architectural support for programming languages and operating systems* (ASPLOS '18).
Zanella, G. (2006). 1/f Noise from the Coincidences of Similar Single-sided Random Telegraph Signals. arXiv:physics/0607044 [physics.data-an].