1-1-2009

A Hypercube-Based Encoding for Evolving Large-Scale Neural Networks

Kenneth O. Stanley
University of Central Florida

David B. D'Ambrosio
University of Central Florida

Jason Gauci
University of Central Florida

A Hypercube-Based Encoding for Evolving Large-Scale Neural Networks

Kenneth O. Stanley*,**
University of Central Florida

David B. D'Ambrosio**
University of Central Florida

Jason Gauci**
University of Central Florida

Abstract Research in neuroevolution—that is, evolving artificial neural networks (ANNs) through evolutionary algorithms—is inspired by the evolution of biological brains, which can contain trillions of connections. Yet while neuroevolution has produced successful results, the scale of natural brains remains far beyond reach. This article presents a method called hypercube-based NeuroEvolution of Augmenting Topologies (HyperNEAT) that aims to narrow this gap. HyperNEAT employs an indirect encoding called connective compositional pattern-producing networks (CPPNs) that can produce connectivity patterns with symmetries and repeating motifs by interpreting spatial patterns generated within a hypercube as connectivity patterns in a lower-dimensional space. This approach can exploit the geometry of the task by mapping its regularities onto the topology of the network, thereby shifting problem difficulty away from dimensionality to the underlying problem structure. Furthermore, connective CPPNs can represent the same connectivity pattern at any resolution, allowing ANNs to scale to new numbers of inputs and outputs *without further evolution*. HyperNEAT is demonstrated through visual discrimination and food-gathering tasks, including successful visual discrimination networks containing over eight million connections. The main conclusion is that the ability to explore the space of regular connectivity patterns opens up a new class of complex high-dimensional tasks to neuroevolution.

Keywords
Compositional pattern-producing networks, CPPNs, HyperNEAT, indirect encoding, hypercube-based NeuroEvolution of Augmenting Topologies, artificial embryogeny

1 Introduction

Many defining characteristics of natural brains have so far eluded attempts to evolve artificial neural networks (ANNs). Perhaps most dramatic is the astronomical complexity of biological brains. With 100 trillion connections whose collective function no artificial system yet can even approach, the human brain is the most complex system known to exist [30, 62]. Upon closer inspection, the brain's complexity is manifested through precise, intricate motifs that repeat throughout it, often with variation on a theme, such as cortical columns [47]. In contrast, neuroevolution (i.e., the artificial evolution of ANNs) produces networks with orders of magnitude fewer neurons and significantly less organization and regularity [18, 51, 60]. These differences between the natural and artificial suggest that something is missing from present evolutionary algorithms that prevents them from achieving similar feats.

* Contact author.
** School of Electrical Engineering and Computer Science, University of Central Florida, 4000 Central Florida Blvd., Orlando, FL 32816-2362. E-mail: kstanley@cs.ucf.edu (K.O.S.); ddambro@cs.ucf.edu (D.B.D.); jgauci@cs.ucf.edu (J.G.)

Researchers in *generative and developmental encoding*, which is a branch of evolutionary computation concerned with genetic encodings motivated by biology, often point out that repetition through genetic reuse explains how such massive structures can be represented compactly in DNA [5, 23, 26, 52]. That is, a structure that repeats many times can be represented by a single set of genes that is *reused* in mapping from genotype to phenotype.

Yet repetition through reuse is not the only important clue to such astronomical complexity. There is an important organizing principle behind many observed regularities that explains their origin: The geometry of the brain is often organized to reflect and exploit the geometry of the physical world. By preserving the most salient physical relationships of the outside world, such as symmetry (e.g., left and right eyes and ears) and locality (e.g., the retinotopic map of visual experience), sensory configuration allows neural organization to largely mirror the same relationships. For example, the visual cortex preserves the retinotopic layout of the eye [7]. In this way, nearby events in physical space are easily represented by similarly proximal neurons.

In fact, because neurons that are related to nearby events in space are also near each other in the brain, many relevant operations can be performed through local connectivity. Such connectivity is natural in a physical substrate in which longer distance requires more resources, greater accuracy, and better organization. Thus, the properties of physical space inherently bias cognitive organization toward local connectivity, which happens to be useful for solving problems that are projected from the physical world.

Interestingly, neuroevolution and ANN optimization in general are normally cast as unknown mappings between inputs and outputs [44, 51, 55, 60]. Yet although this perspective is widespread, in effect it obfuscates the underlying problem geometry, because geometric relationships among inputs and outputs are discarded when they are represented only as an unstructured set of independent parameters (Figure 1). Correlations and regularities in the task domain thereby become opaque to the learning algorithm, which must consequently learn related functions in different parts of the network independently. For example, a visual field or board game is often split into dozens of independent coordinates with no a priori explicit relationship to each other [29, 31, 38]. A promising alternative would be to learn a *conceptual* representation of the solution that is a function of the problem's geometric structure.

This article presents such a neuroevolution method, called *hypercube-based NeuroEvolution of Augmenting Topologies* (HyperNEAT), which is designed to evolve large-scale ANNs by exploiting geometry. HyperNEAT employs an encoding called *connective compositional pattern-producing networks* (CPPNs), which can represent connectivity patterns as functions in Cartesian space. Connective CPPNs are

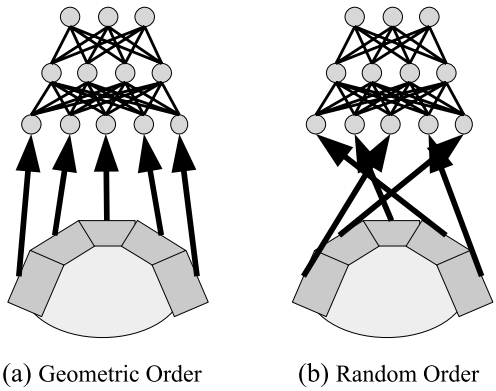


Figure 1. Regular ANN: Input order is irrelevant. Even though the order of inputs to the ANN in (a) is correlated directly with the geometry of the front-facing robot sensors from which they are activated, traditional ANN learning algorithms such as backpropagation [44] or current neuroevolution methods [51, 60] are blind to such ordering. In fact, the arbitrary input order in (b) is identical from the perspective of the learning algorithm. Thus, counterintuitively, organizing inputs (or outputs) to respect task geometry provides no advantage.

an extension of regular CPPNs, an indirect encoding for spatial patterns that is abstracted from biological development [48, 49]. Specifically, connective CPPNs represent patterns in hyperspace that are mapped to lower-dimensional connectivity patterns. That way, connective CPPNs evolved with HyperNEAT can encode large-scale ANNs by discovering regularities along geometric dimensions of variation in a manner motivated by the evolution of biological brains in nature.

The two experiments in this article aim to provide sufficient technical and conceptual insight to allow progress in this new direction to proceed systematically from here. Thus, the first experiment explores how geometry enables large-scale representation through reuse in a simple visual discrimination task with scalable resolution. The second experiment isolates the issue of exploiting geometric regularities by comparing two different sensor and effector layouts for a simple food-gathering robot. An analysis of scaling concludes by scaling an evolved ANN *without further evolution* to a size of over eight million connections without loss of functionality.

Because it not only implements repetition through reuse but also exploits domain geometry, HyperNEAT opens up significant new directions for future exploration in evolving ANNs.

The article begins with a review of CPPNs and NEAT in the next section. The HyperNEAT approach is then detailed in Section 3. Sections 4 and 5 describe and present results in the visual discrimination experiment. Sections 6 and 7 then present the food gathering experiment and results. The article concludes with a discussion and outlines future work in Section 8.

2 Background

This section provides an overview of CPPNs, which are capable of generating complex spatial patterns in Cartesian space, and then describes the NEAT method that is used to evolve them.

2.1 Compositional Pattern-Producing Networks

In biological genetic encoding the mapping between genotype and phenotype is indirect. The phenotype typically contains orders of magnitude more structural components than the genotype contains genes. For example, a human genome of 30,000 genes (about three billion amino acids) encodes a human brain with 100 trillion connections [10, 12, 30]. Thus, the only way to discover structures with trillions of parts may be through a mapping between genotype and phenotype that translates few dimensions into many, that is, through an *indirect encoding*. Because phenotypic structures often occur in repeating patterns, each time a pattern repeats, the same gene group can provide the specification. The numerous left-right symmetries of vertebrates [40: 302–303], the receptive fields in the visual cortex [17, 27], and fingers and toes are examples of repeating patterns in biology.

A most promising area of research in indirect encoding is *developmental encoding*, which is motivated from biology [2, 5, 26, 52]. Development facilitates reusing genes because the same gene can be activated at any location and any time during the development process.

This observation has inspired an active field of research in artificial developmental encodings [2, 4–6, 9, 11, 13–15, 21, 26, 28, 32–34, 36, 37, 39, 46, 52, 57]. The aim is to find the right abstraction of natural development for a computer running an evolutionary algorithm, so that evolutionary computation (EC) can begin to discover complexity on a natural scale. Prior abstractions range from low-level cell chemistry simulations to high-level grammatical rewrite systems [52].

Compositional pattern-producing networks (CPPNs) are a novel abstraction of development that can represent sophisticated repeating patterns in Cartesian space [48, 49]. Unlike most generative and developmental encodings, CPPNs do not require an explicit simulation of growth or local interaction, yet still realize their essential functions. This section reviews CPPNs, which will be augmented in this article to represent connectivity patterns and ANNs.

Consider the phenotype as a function of n dimensions, where n is the number of dimensions in physical space. For each coordinate in that space, its level of expression is an output of the function that encodes the phenotype. Figure 2a shows how a two-dimensional phenotype can be generated by a function of two parameters.

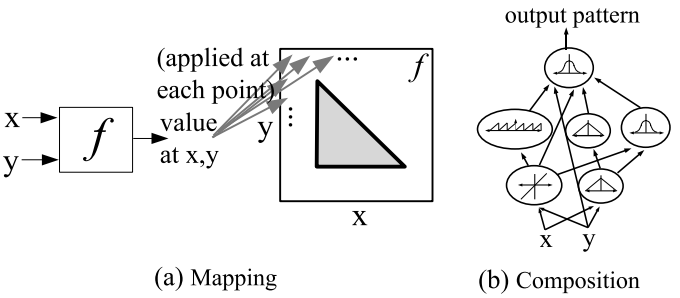


Figure 2. CPPN encoding. (a) The function *f* takes arguments *x* and *y*, which are coordinates in a two-dimensional space. When all the coordinates are drawn with an intensity corresponding to the output of *f*, the result is a spatial pattern that can be viewed as a phenotype whose genotype is *f*. (b) The CPPN is a graph that determines which functions are connected. The connections are weighted so that the output of a function is multiplied by the weight of its outgoing connection. Note that the topology is unconstrained and can represent any relationships.

Stanley [48, 49] showed how simple canonical functions can be composed to create an overall network that produces complex regularities and symmetries. Each component function creates a novel geometric *coordinate frame* within which other functions can reside. The main idea is that these simple canonical functions are abstractions of specific events in development, such as establishing bilateral symmetry (e.g., with a symmetric function such as Gaussian) or dividing the body into discrete segments (e.g., with a periodic function such as the sine). Figure 2b shows how such a composition can be represented by a network.

Such networks are called *compositional pattern-producing networks* because they produce spatial patterns by composing basic functions. Unlike ANNs, which often contain only sigmoid functions (and sometimes Gaussian functions), CPPNs can include both those types of functions and many others. Furthermore, the term *artificial neural network* would be misleading in the context of this research, because ANNs were so named to establish a metaphor with a different biological phenomenon, namely, the brain. The terminology should avoid the implication that biological, thinking brains are in effect the same as developing embryos or genetic encodings. In this article, because CPPNs are used to encode ANNs, it is especially important to differentiate these concepts.

Through interactive evolution, Stanley [48, 49] demonstrated that CPPNs can produce spatial patterns with important geometric motifs that are expected from generative and developmental encodings and seen in nature. Among the most important such motifs are symmetry (e.g., left-right symmetries in vertebrates), imperfect symmetry (e.g., right-handedness), repetition (e.g., receptive fields in the cortex [62]), and repetition with variation (e.g., cortical columns [19]). Figure 3 shows examples of several such important motifs produced through interactive evolution of CPPNs.

That CPPNs and ANNs are so similar from a structural perspective is fortunate in that methods designed to evolve ANNs can also evolve CPPNs. In particular, the NeuroEvolution of Augmenting

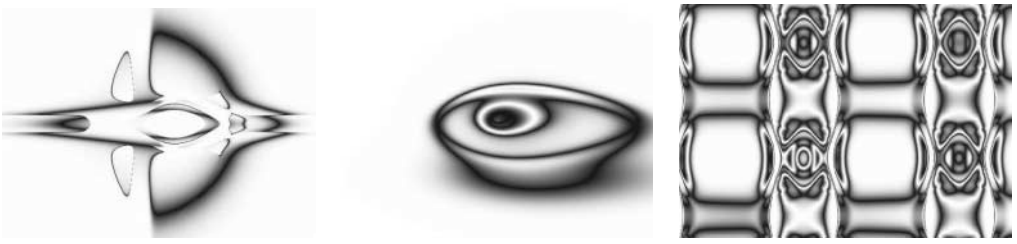


Figure 3. CPPN-generated regularities. Spatial patterns exhibiting (a) bilateral symmetry, (b) imperfect symmetry, and (c) repetition with variation (notice the nexus of each repeated motif) are depicted. These patterns demonstrate that CPPNs effectively encode fundamental regularities of several different types.

Topologies (NEAT) method is a good choice for evolving CPPNs because NEAT increases the complexity of evolving networks over generations, allowing increasingly elaborate regularities to accumulate. The next subsection describes the NEAT method.

2.2 NeuroEvolution of Augmenting Topologies

The NEAT method was originally developed to evolve ANNs to solve difficult control and sequential decision tasks [50, 51, 53]. Evolved ANNs control agents that select actions according to their sensory inputs. NEAT is unlike many previous methods that evolved neural networks (i.e., *neuroevolution* methods), which traditionally evolve either fixed-topology networks [18, 45] or arbitrary random-topology networks [3, 22, 60]. Instead, NEAT begins evolution with a population of small, simple networks and *complexifies* the network topology into diverse species over generations, leading to increasingly sophisticated behavior. A similar process of gradually adding new genes has been confirmed in natural evolution [35, 58] and shown to improve adaptation in a few prior evolutionary [1] and neuroevolutionary [24] approaches. However, a key feature that distinguishes NEAT from prior work in complexification is its unique approach to maintaining a healthy diversity of complexifying structures simultaneously, as this section reviews. Complete descriptions of the NEAT method, including experiments confirming the contributions of its components, are available in Stanley and Miikkulainen [51, 53] and Stanley et al. [50].

Before describing the CPPN extension, let us review the three key ideas on which the basic NEAT method is based. First, in order to allow network structures to increase in complexity over generations, a method is needed to keep track of which gene is which. Otherwise, it is not clear in later generations which individual is compatible with which in a population of diverse structures, or how their genes should be combined to produce offspring. NEAT solves this problem by assigning a unique *historical marking* to every new piece of network structure that appears through a structural mutation. The historical marking is a number assigned to each gene corresponding to its order of appearance over the course of evolution. The numbers are inherited unchanged during crossover, and allow NEAT to perform crossover among diverse topologies without the need for expensive topological analysis.

Second, NEAT speciates the population, so that individuals compete primarily within their own niches instead of with the population at large. Because adding new structure is often initially disadvantageous, this separation means that unique topological innovations are protected and therefore have time to optimize their structure before competing with other niches in the population. NEAT uses the historical markings on genes to determine to which species different individuals belong.

Third, many systems that evolve network topologies and weights begin evolution with a population of random topologies [22, 60]. In contrast, NEAT begins with a uniform population of simple networks with no hidden nodes, differing only in their initial random weights. Because of speciation, novel topologies gradually accumulate over evolution, thereby allowing diverse and complex phenotype patterns to be represented. No limit is placed on the size to which topologies can grow. New structures are introduced incrementally as structural mutations occur, and only those structures survive that are found to be useful through fitness evaluations. In effect, then, NEAT searches for a compact, appropriate topology by incrementally complexifying existing structure.

NEAT is easily extended to evolve CPPNs. While networks in the original NEAT only include hidden nodes with sigmoid functions, CPPN NEAT allows each hidden node to specify its own activation function chosen from a given set. These functions are intended to abstract canonical gradient patterns from nature, such as symmetric and periodic gradients. When a new node is created, it is assigned a random activation function from the canonical set (e.g., including Gaussian, sigmoid, and periodic functions).

If the regularities of CPPN-generated patterns evolved with NEAT could be transferred to evolved *connectivity patterns*, the representational power of CPPNs could potentially evolve large-scale ANNs and other graph structures with symmetries and complex repeating patterns such as in biological brains. The next section introduces an approach that enables CPPNs to represent and evolve just such networks.

3 HyperNEAT

If CPPNs are to evolve and represent connectivity patterns, the problem is to find the best interpretation of their output to effectively describe such a structure. The two-dimensional patterns in Section 2.1 present a challenge: How can such spatial patterns describe connectivity? This section explains how spatial patterns generated by CPPNs can be mapped naturally to connectivity patterns while at the same time effectively disentangling task structure from network dimensionality.

3.1 Mapping Spatial Patterns to Connectivity Patterns

It turns out that there is an effective mapping between spatial and connectivity patterns that can elegantly exploit geometry. The main idea is to input into the CPPN the coordinates of the *two points* that define a connection, rather than inputting only the position of a single point as in Section 2.1. The output is interpreted as the *weight* of the connection rather than the intensity of a point. This way, connections can be defined in terms of the locations that they connect, thereby taking into account the network's geometry.

The CPPN in effect computes a four-dimensional function CPPN$(x_1, y_1, x_2, y_2) = w$, where the first node is at (x_1, y_1) and the second node is at (x_2, y_2). This formalism returns a weight for every connection between every node in the grid, including recurrent connections. By convention, a connection is not expressed if the magnitude of its weight, which may be positive or negative, is below a minimal threshold $w_{\min}$. The magnitudes of weights above this threshold are scaled to be between zero and a maximum magnitude in the substrate. That way, the pattern produced by the CPPN can represent any network topology (Figure 4).

For example, consider a 5×5 grid of nodes. The nodes are assigned coordinates corresponding to their positions within the grid (labeled "Substrate" in Figure 4), where (0, 0) is the center of the grid. Assuming that these nodes and their positions are given a priori, a connectivity pattern among nodes in two-dimensional space is produced by a CPPN that takes any two coordinates (source and target) as input, and outputs the weight of their connection. The CPPN is queried in this way for every potential connection on the grid. Because the connection weights are thereby a function of the *positions* of their source and target nodes, the distribution of weights on connections throughout the grid will exhibit a pattern that is a function of the geometry of the coordinate system.

The connectivity pattern produced by a CPPN in this way is called the *substrate* so that it can be verbally distinguished from the CPPN itself, which has its own internal topology. Furthermore, in

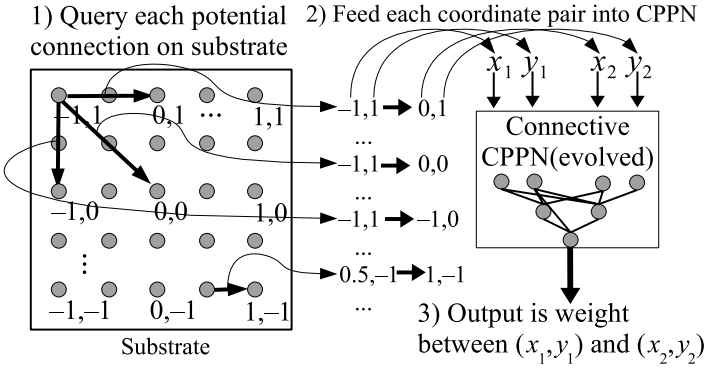


Figure 4. Hypercube-based geometric connectivity pattern interpretation. A grid of nodes, called the *substrate*, is assigned coordinates such that the center node is at the origin. (1) Every potential connection in the substrate is queried to determine its presence and weight; the dark directed lines shown in the substrate represent a sample of connections that are queried. (2) For each query, the CPPN takes as input the positions of the two endpoints and (3) outputs the weight of the connection between them. After all connections are determined, a pattern of connections and connection weights results that is a function of the geometry of the substrate. In this way, *connective CPPNs* produce regular patterns of connections in space.

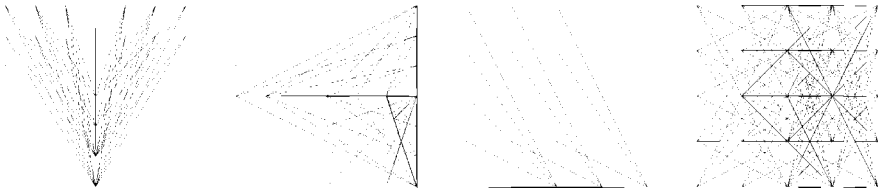


Figure 5. Connectivity patterns produced by connective CPPNs. These patterns, produced through interactive evolution, exhibit several important connectivity motifs: (a) bilateral symmetry, (b) imperfect symmetry, (c) repetition, and (d) repetition with variation. That these fundamental motifs are compactly represented and easily produced suggests the power of this encoding.

the remainder of this article, CPPNs that are interpreted to produce connectivity patterns are called *connective* CPPNs, while CPPNs that generate spatial patterns are called *spatial* CPPNs. This article focuses on neural substrates produced by connective CPPNs.

Because the connective CPPN is a function of four dimensions, the two-dimensional connectivity pattern expressed by the CPPN is isomorphic to a spatial pattern embedded in a four-dimensional hypercube. This observation is important because it means that spatial patterns with symmetries and regularities correspond to connectivity patterns with related regularities. Thus, because CPPNs generate regular spatial patterns (Section 2.1), by extension they can be expected to produce connectivity patterns with corresponding regularities. The next section demonstrates this capability.

3.2 Producing Regular Connectivity Patterns

Simple, easily discovered substructures in the connective CPPN produce important connective regularities in the substrate. The key difference between connectivity patterns and spatial patterns is that each discrete unit in a connectivity pattern has *two* x values and *two* y values. Thus, for example, symmetry along x can be discovered simply by applying a symmetric function (e.g., a Gaussian) to x_1 or x_2 (Figure 5a).

Imperfect symmetry is another important structural motif in biological brains. Connective CPPNs can produce imperfect symmetry by composing not only symmetric functions of one axis, but also an asymmetric coordinate frame such as the axis itself. In this way, the CPPN produces varying degrees of imperfect symmetry (Figure 5b).

Similarly important is repetition, particularly repetition with variation. Just as symmetric functions produce symmetry, periodic functions such as the sine produce repetition (Figure 5c). Patterns with variation are produced by composing a periodic function with a coordinate frame that does not repeat, such as the axis itself (Figure 5d). Repetitive patterns can also be produced in connectivity as functions of invariant properties between two nodes, such as distance along one axis. Thus, symmetry, imperfect symmetry, repetition, and repetition with variation, key structural motifs in all biological brains, are compactly represented and therefore easily discovered by CPPNs.

3.3 Substrate Configuration

The layout of the nodes that the CPPN connects in the substrate can take forms other than the planar grid (Figure 6a) discussed thus far. Different such *substrate configurations* are likely suited to different kinds of problems.

For example, CPPNs can also produce three-dimensional connectivity patterns by representing spatial patterns in the six-dimensional hypercube CPPN(x_1, y_1, z_1, x_2, y_2, z_2) (Figure 6b). This formalism is interesting because the topologies of biological brains, including the human brain, theoretically exist within its search space.

It is also possible to restrict substrate configurations to particular structural motifs in order to learn about their viability in isolation. For example, Churchland [8] calls a single two-dimensional sheet of neurons that connects to another two-dimensional sheet a *state-space sandwich*. The sandwich is a restricted three-dimensional structure in which one layer can send connections only in one direction to one other layer. Thus, because of this restriction, it can be expressed by the single four-dimensional CPPN(x_1, y_1,

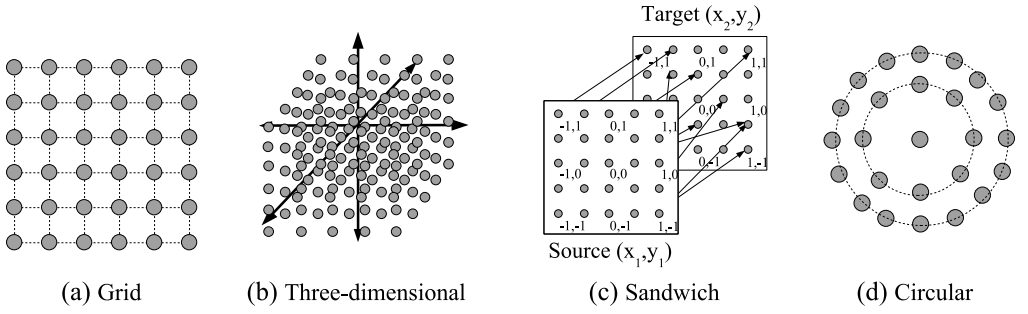


Figure 6. Alternative substrate configurations. This figure shows (a) the original grid configuration introduced in Figure 4, (b) a three-dimensional configuration of nodes centered at (0, 0, 0), (c) a *state-space sandwich* configuration in which a source sheet of neurons connects directly to a target sheet, and (d) a circular configuration. Different configurations are likely suited to problems with different geometric properties.

x_2, y_2), where (x_2, y_2) is interpreted as a location on the *target* sheet rather than as being on the same plane as the source coordinate (x_1, y_1). In this way, CPPNs can search for useful patterns within state-space sandwich substrates (Figure 6c), as is done in the visual discrimination experiment in this article.

Finally, the nodes need not be distributed in a grid. For example, nodes within a substrate that controls a radial entity such as a starfish might be best laid out with radial geometry, as shown in Figure 6d, so that the connectivity pattern can be situated with exact polar coordinates. The robot control experiment in this article compares such a circular layout with a grid-based one.

3.4 Input and Output Placement

Part of substrate configuration is determining which nodes are inputs and which are outputs. The flexibility to assign inputs and outputs to specific coordinates in the substrate creates an opportunity to exploit geometric relationships advantageously.

In many ANN applications, the inputs are drawn from a set of sensors that exist in a geometric arrangement in space. Unlike traditional ANN learning algorithms that are not aware of such geometry (as illustrated in Figure 1), connective CPPN substrates are aware of their inputs' and outputs' geometry, and thus can use this information to their advantage.

By arranging inputs and outputs in a sensible configuration on the substrate, regularities in the geometry can be exploited by the encoding. There is room to be creative and try different configurations with different geometric advantages. For example, Figure 7 depicts two methods in which the inputs and outputs of a circular robot can be configured, each of which creates an opportunity to exploit a different kind of geometric relationship.

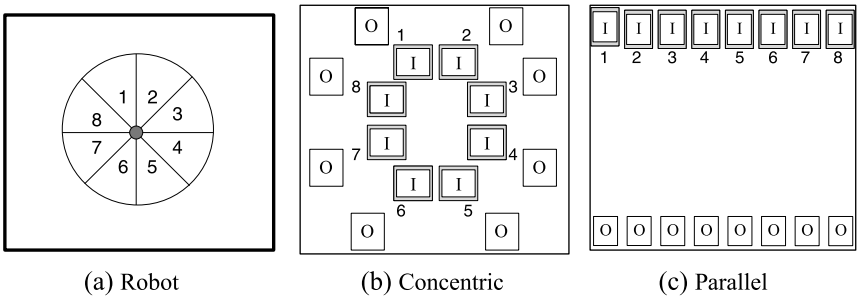


Figure 7. Placing inputs and outputs. A robot (a) is depicted with eight radar sensors along its circumference and eight motion effectors set at the same angles. In (b), the inputs (labeled *I*) and outputs (labeled *O*) are laid out literally according to the eight directions in space. In (c), the inputs are placed so that their location along *x* determines whether they represent a corresponding direction. Both arrangements create a geometric relationship between each input and its corresponding output. In this way, it is possible the give evolution a significant advantage from the start.

In one arrangement, the sensors on the circumference of the robot are arranged in a circle centered at the origin of the substrate, and outputs form a concentric circle around that (Figure 7b). In this way, if the CPPN discovers radial symmetry or bilateral symmetry, it can use those coordinate frames to create a repeating pattern that captures regularities in the relationship between inputs and outputs. An alternative arrangement places the inputs and outputs on two parallel lines whereon equivalent horizontal position denotes equivalent angle (Figure 7c). That way, evolution can exploit the similarity of horizontal positions. Each method conveys correspondence through a different geometric regularity.

By arranging neurons in a sensible configuration on the substrate, regularities in the geometry can be exploited by the encoding. Biological neural networks rely on such a capability for many of their functions. For example, neurons in the visual cortex are arranged in the same retinotopic two-dimensional pattern as photoreceptors in the retina [7]. That way, they can exploit *locality* by connecting to adjacent neurons with simple, repeating motifs. Connective CPPNs have the same capability. In fact, geometric information in effect provides evolution with domain-specific bias, which is necessary if it is to gain an advantage over generic black-box optimization methods [59].

3.5 Substrate Resolution

As opposed to encoding a specific pattern of connections among a specific set of nodes, connective CPPNs in effect encode a general *connectivity concept*, that is, a set of underlying mathematical relationships that produce a particular pattern. The consequence is that the same connective CPPN can represent equivalent concepts at different resolutions (i.e., different node densities). Figure 8 shows two connectivity concepts at different resolutions.

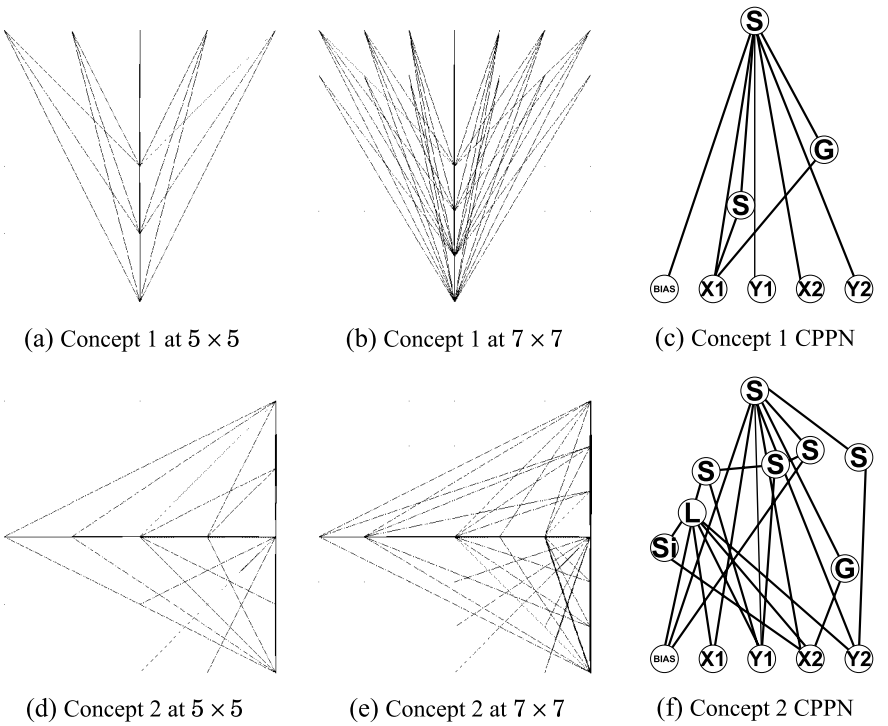


Figure 8. Equivalent connectivity concepts at different substrate resolutions. Two connectivity concepts are depicted that were evolved through interactive evolution. The CPPN that generates the first concept at 5 × 5 (a) *and* 7 × 7 (b) is shown in (c). The CPPN in (f) similarly generates the second concept at both resolutions (d) and (e). This illustration demonstrates that CPPNs represent a mathematical concept rather than a single structure. Thus, the same CPPN can produce patterns with the same underlying concept at different substrate resolutions (i.e., different node densities). CPPN activation functions in this article are denoted by *G* for Gaussian, *S* for sigmoid, *Si* for sine, *A* for absolute value, and *L* for linear.

For neural substrates, the important implication is that the same ANN functionality can be generated at different resolutions. *Without further evolution*, previously evolved connective CPPNs can be re-queried to specify the connectivity of the substrate at a new, higher resolution, thereby producing a working solution to the same problem at a higher resolution. There is no upper bound on substrate resolution, that is, a connectivity concept is infinite in resolution. While the higher-resolution connectivity pattern may contain artifacts that were not expressed at the lower resolution at which it was evolved, it will still embody a good approximation of the general solution at the higher resolution. Thus, increasing substrate resolution introduces a powerful new kind of complexification into ANN evolution.

3.6 Evolving Connective CPPNs

The approach in this article is to evolve connective CPPNs with NEAT. This approach is called *HyperNEAT*, because NEAT evolves CPPNs that represent spatial patterns in hyperspace. Each point in the pattern, bounded by a hypercube, is interpreted as a connection in a lower-dimensional connected graph. Specifically, in this article, a spatial pattern in a four-dimensional hypercube is interpreted as a two-dimensional connectivity pattern.

The basic outline of the HyperNEAT algorithm proceeds as follows:

1. Choose the substrate configuration (i.e., node layout and input-output assignments)
2. Initialize the population of minimal CPPNs with random weights.
3. Repeat until a solution is found:
   (a) For each member of the population:
      i. Query its CPPN for the weight of each possible connection in the substrate. If the absolute value of the output exceeds a threshold magnitude, create the connection with a weight scaled proportionally to the output value (Figure 4).
      ii. Run the substrate as an ANN in the task domain to ascertain fitness.
   (b) Reproduce the CPPNs according to the NEAT method to produce the next generation population.

In effect, as HyperNEAT adds new connections and nodes to the connective CPPN, it is discovering new *global dimensions of variation* in connectivity patterns across the substrate. Early on it may discover overall symmetry, whereas later it may discover the concept of receptive fields. Each new connection or node in the CPPN represents a new way that an entire pattern can vary—that is, a new regularity. Thus, HyperNEAT is a powerful new approach to evolving large-scale connectivity patterns and ANNs.

The sections that follow present experiments that demonstrate the promise of this approach.

4 Experiment 1: Visual Discrimination

The first experiment in this article is visual discrimination; the second is robot food gathering. Both tasks are chosen for their intuitive simplicity, ability to demonstrate specific HyperNEAT capabilities, and ease of analysis, thereby laying the foundation for future research. They are both designed to provide empirical evidence that HyperNEAT makes possible five novel capabilities: (1) compact encoding through regular structure, (2) exploiting sensor placement and world geometry, (3) scaling substrate resolution, (4) leveraging additional inputs that provide geometric bias, and (5) functional million-connection networks.

This section describes the visual discrimination task, and Section 5 presents its results. The food-gathering task is then explained in Section 6.

4.1 Visual Discrimination Setup

Vision is well suited to testing learning methods on high-dimensional input. Natural vision also has the intriguing property that the same stimulus can be recognized equivalently at different locations in the visual field. For example, identical line-orientation detectors are spread throughout the primary visual cortex [7]. Thus there are clear regularities among the local connectivity patterns that govern such detection. A repeating motif likely underlies the general capability to perform similar operations at different locations in the visual field. Visual tasks are also easy to scale by increasing their resolution.

Therefore, in this article a simple visual discrimination task demonstrates HyperNEAT's ability to exploit regularity in the task domain. The objective is to distinguish a large object from a small object in a two-dimensional visual field. Because the same principle determines the difference between small and large objects regardless of their location on the retina, this task is well suited to testing the ability of HyperNEAT to discover and exploit regularities.

The solution substrate is configured as a state-space sandwich (Figure 6c) that includes two sheets: (1) the *visual field* is a two-dimensional array of sensors that are either on or off (i.e., black or white); (2) the *target field* is an equivalent two-dimensional array of outputs that are activated at variable intensity between zero and one. In a single trial, two objects, represented as black squares, are situated in the visual field at different locations. One object is three times as wide and tall as the other (Figure 9). The goal is to locate the center of the larger object in the visual field. The target field specifies this location as the node with the highest level of activation. Thus, HyperNEAT must discover a connectivity pattern between the visual field and target field that causes the correct node to become the most active regardless of the locations of the objects.

An important aspect of this task is that it utilizes a large number of inputs, many of which must be considered simultaneously. To solve it, the system needs to discover the general principle that underlies detecting relative sizes of objects. The right idea is to strongly connect an individual input node in the visual field to several adjacent nodes around the corresponding location in the output field, thereby causing outputs to accumulate more activation the more adjacent loci are feeding into

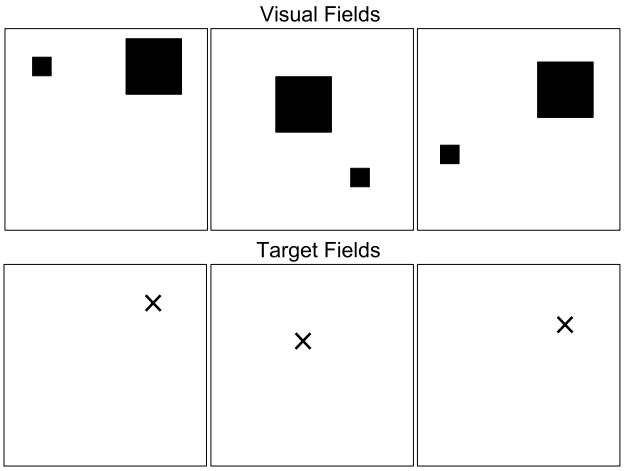


Figure 9. The visual discrimination task. The task is to identify the center of the larger box. Example visual field activation patterns (top) and the corresponding correct target fields (bottom) are depicted. The × in each target field denotes the point of highest activation, which is how the ANN specifies the location of the center of the larger box. This task effectively tests HyperNEAT's ability to discover regularity, because the same principle differentiates the larger box from the smaller one regardless of where the boxes appear on the input field.

them. Thus, the solution can exploit the geometric concept of locality, which is inherent in the arrangement of the two-dimensional grid.

While the concept is simple, only a representation that takes into account substrate geometry can exploit it effectively. Furthermore, an ideal encoding should develop a representation of the concept that is independent of the visual field's resolution. Because the correct motif repeats across the substrate, in principle a connective CPPN can discover the general concept only once and cause it to be repeated across the grid at any resolution. As a result, such a solution can scale as the resolution inside the visual field is increased, even without further evolution.

4.2 Evolution and Performance Analysis

The field coordinates range over $[-1, 1]$ in the x and y dimensions. However, the resolution within this range (i.e., the node density) can be varied. During evolution, the resolution of each field is fixed at 11×11. Thus the connective CPPN must learn to correctly connect a visual field of 121 inputs to a target field of 121 outputs, for a total of 14,641 potential connection strengths.

During evolution, each individual in the population is evaluated for its ability to find the center of the bigger object. If the connectivity is not highly accurate, it is likely the substrate will often incorrectly choose the small object over the large one. Each individual evaluation thus includes 75 *trials*, where each trial places the two objects at different locations. The trials are organized as follows. The small object appears at 25 uniformly distributed locations such that it is always completely within the visual field. For each of these 25 locations, the larger object is placed five units to the right, down, and diagonally, once per trial. The large object wraps around to the other side of the field when it hits the border. If the large object is not completely within the visual field, it is moved the smallest distance possible that places it fully in view. Because of wrapping, this method of evaluation tests cases where the small object is on all possible sides of the large object. Thus many relative positions (though not all) are tested for a total number of 75 trials on the 11×11 substrate for each evaluation during evolution.

Within each trial, the substrate is activated over the entire visual field. The unit with the highest activation in the target field is interpreted as the substrate's selection. The fitness is calculated as the sum of the squared distances between the target and the point of highest activation over all 75 trials. This fitness function rewards generalization and provides a smooth gradient for solutions that are close but not perfect.

An effective solution to this task must discover the correct underlying regularity that is distributed across the substrate. Although it is possible for humans to imagine such a repeating motif, from the perspective of a blind machine learning algorithm such discovery is nontrivial. To demonstrate HyperNEAT's ability to effectively discover the task's underlying regularity, two approaches are compared.

- *HyperNEAT:* HyperNEAT evolves a connective CPPN that generates a substrate to solve the problem (Section 3).
- *Perceptron NEAT (PNEAT):* PNEAT is a reduced version of NEAT that evolves perceptrons (i.e., it is a *direct* encoding of the ANN that does not evolve CPPNs). ANNs with 121 inputs, 121 outputs, and 14,641 (121×121) links are evolved without structure-adding mutations. This restriction makes a fair comparison, because state-space sandwich substrates also have no hidden nodes. PNEAT is run with the same settings as HyperNEAT (see the Appendix), because both are being applied to the same problem. Because PNEAT must explicitly encode the value of each connection in the genotype, it cannot encode underlying regularities and must discover each part of the solution connectivity independently.

This comparison is designed to show how HyperNEAT makes it possible to optimize very high-dimensional structures (namely, with 14,641 dimensions), which is difficult for directly encoded methods.

4.3 Scaling to Millions of Connections

HyperNEAT is finally tested for its ability to scale solutions to higher resolutions without further evolution, which is impossible with direct encodings such as PNEAT. The resolution is increased to 33 × 33 and 55 × 55, requiring HyperNEAT to set over one million and nine million connections, respectively.

4.4 Inputting Additional Geometric Bias

The connectivity pattern produced by the connective CPPN(x_1, y_1, x_2, y_2) is a function of four orthogonal axes. The axes provide a coordinate frame within which patterns are situated. A potentially useful feature of CPPNs is that they can be biased to specific kinds of geometry by taking other coordinate frames as input as well. Thus, because distance is an important factor in discrimination, the distance between the two query points is input into the CPPN in addition to the usual x_1, y_1, x_2, and y_2. In visual discrimination the additional inputs are vertical and horizontal deltas ($x_1 - x_2$ and $y_2 - y_1$). To explore the benefits of this capability in this domain, a separate experiment is performed with the additional inputs, and its performance is compared with the results without them.

Experimental parameters for both experiments in this article are provided in the Appendix.

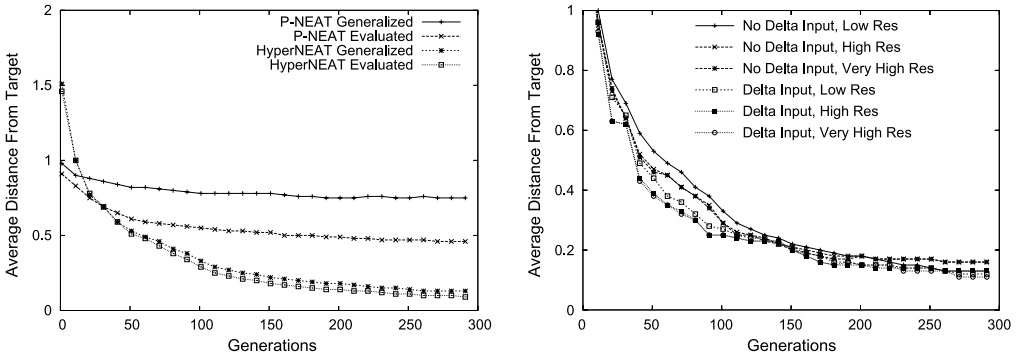
5 Visual Discrimination Results

The primary performance measure in this section is the *average distance from target* of the target field's chosen position. This average is calculated for each generation's champion across all its trials (i.e., object placements in the visual field). Reported results were averaged over 20 runs. Better solutions choose positions closer to the target. To understand the distance measure, note that the width and height of the substrate are 2.0 regardless of substrate resolution.

HyperNEAT and PNEAT were compared to quantify the advantage provided by generative encoding on this task. Figure 10a shows the performance of both methods on *evaluation* trials from evolution (i.e., a subset of all possible positions) and on a generalization test that averaged the performance over every possible valid pair of positions on the board. An input is considered valid if the smaller and the larger object are placed within the substrate and neither object overlaps the other.

The performance of both methods on the evaluation tests improved over the run. However, after generation 45, on average HyperNEAT found significantly more accurate solutions than PNEAT ($p < .01$).

HyperNEAT learned to generalize from its training; the difference between the performance of HyperNEAT in generalization and evaluation is not significant past the first generation. In contrast, PNEAT performed significantly worse in the generalization test after generation 51 ($p < .01$). This



(a) P-NEAT and HyperNEAT Generalization (b) HyperNEAT Generation Champion Scaling

Figure 10. Generalization and scaling. The graphs show performance curves over 300 generations averaged over 20 runs each. (a) PNEAT is compared with HyperNEAT on both evaluation and generalization. (b) HyperNEAT generation champions with and without delta inputs are evaluated for their performance on 11 × 11, 33 × 33, and 55 × 55 substrate resolutions. The results show that HyperNEAT generalizes significantly better than PNEAT ($p < .01$) and scales almost perfectly.

disparity in generalization reflects HyperNEAT's fundamental ability to learn the geometric concept underlying the task, which can be generalized across the substrate. PNEAT can only discover each proper connection weight independently. Therefore, PNEAT has no way to extend its solution to positions on the substrate at which it was never evaluated. Furthermore, the search space of 14,641 dimensions (one for each connection) is too high-dimensional for PNEAT to find good solutions, while HyperNEAT discovers near-perfect (and often perfect) solutions on average.

5.1 Scaling Performance

The best individuals of each generation, which were evaluated on 11 × 11 substrates, were later scaled with the same CPPN to resolutions of 33 × 33 and 55 × 55 by re-querying the substrate at the higher resolutions without further evolution. These new resolutions cause the substrate size to expand dramatically. For 33 × 33 and 55 × 55 resolutions, the weights of over one million and nine million connections, respectively, must be optimized in the substrate, which would normally be an enormous optimization problem. On the other hand, the original 11 × 11 resolution on which HyperNEAT was trained contains only up to 14,641 connections. Thus, the number of connections increases by nearly three orders of magnitude. It is important to note that HyperNEAT is able to scale to these higher resolutions without any additional evolution. In contrast, PNEAT has no means to scale to a higher resolution and cannot learn effectively even at the lowest resolution.

When scaling, a potential problem is that if the same activation level were used to indicate positive stimulus as at lower resolutions, the total energy entering the substrate would increase as the substrate resolution increases for the same images, leading to oversaturation of the target field. In contrast, in the real world, the number of photons that enter the eye is the same regardless of the density of photoreceptors. To allow for this disparity, the input activation levels are scaled for larger substrate resolutions proportionally to the difference in unit-cell size.

Evolved CPPNs with and without the additional x and y delta inputs (Section 4.4) were tested for their ability to scale (Figure 10b). While the deltas did perform significantly better on average between generations 38 and 70 ($p < .05$), the CPPNs without delta inputs were able to catch up and reach the same level of performance after generation 70.

Most importantly, both variants were able to scale almost perfectly from the 11 × 11-resolution substrate with up to 14,641 connections to a 55 × 55-resolution substrate with up to 9,150,625 connections, with no significant difference in performance after the second generation. This result is also significant in that the higher-resolution substrates were tested on *all* valid object placements, which include many positions that did not even exist on the lower-resolution substrate. Thus, remarkably, CPPNs found solutions that lose no abilities at higher resolution.

High-quality CPPNs at the 55 × 55 resolution contained on average 8.3 million connections in their substrate and performed as well as their 11 × 11 counterparts. These substrates are the largest functional ANNs produced through evolutionary computation of which the authors are aware.

5.2 Scaling Analysis

Figure 11 illustrates the ability to scale in this task by showing how the activation patterns on target substrates of varying resolution produce the same result for the same input. The larger (55 × 55) substrate (Figure 11d) contains 8,474,704 connections and still solves the task. Activation patterns at all resolutions result from several bands of output overlapping at the center of an object, causing it to activate highest. Thus, the CPPN discovered this underlying concept rather than a specific set of connection weights.

5.3 Repeating Patterns

To identify the largest object irrespective of both objects' locations, HyperNEAT must discover a fundamental connectivity motif originating from each input neuron and repeat it across the substrate. Figure 12 shows a diagonal cross connectivity motif originating from three different input neurons on the same substrate, which is generated by the connective CPPN in Figure 12a.

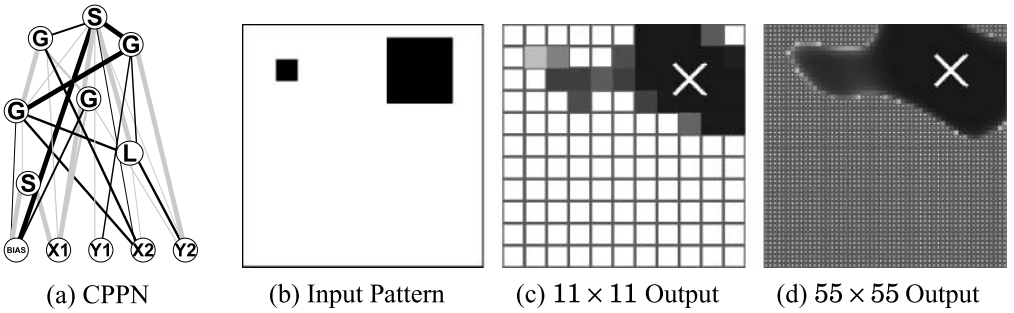


Figure 11. Activation patterns of the same connective CPPN at different resolutions. Activation patterns on the target field of a substrate generated by the CPPN in (a) from the input trial shown in (b) are displayed at resolution 11 × 11 in (c) and 55 × 55 in (d). The darker color signifies higher activation, and the position of highest activation is marked with a white ×. The same 26-connection CPPN generates solutions at both resolutions, with 10,328 and 8,474,704 connections, respectively, demonstrating the ability of the solution to scale significantly.

CPPN encoding is highly compact. If *good* solutions are those that achieve an average distance under 0.25, the average complexity of a good solution CPPN was only 24 connections. In comparison, at 11 × 11 resolution the average number of connections in the substrate was 12,827 out of the possible 14,641 connections. Thus the genotype is smaller than the evaluated phenotype on average by a factor of 534, demonstrating the significant efficiency gained by removing the need to find the same motif multiple times, even for substrates that contain millions of connections.

HyperNEAT discovered several motifs that are all effective at this task. The most intuitive motif is the *halo*, which produces activation patterns such as those in Figure 13. Halo motifs were discovered in eight of the 20 runs. Although they were less common, diagonal cross motifs evolved separately four times. The remaining eight runs produced a variety of different shapes that all work equally well. These results show that HyperNEAT creatively exploits the task geometry to find a variety of effective repeating patterns.

5.4 Discovering Regularities

Figure 13 shows a solution at two different generations in the same run, illustrating the unique process through which regularities in the solution are discovered. In generation 20 (Figure 13a–c), the substrate produces halo connectivity patterns projecting from a single input node that are at the correct vertical position, but it has not yet learned the principle of horizontal locality. Four

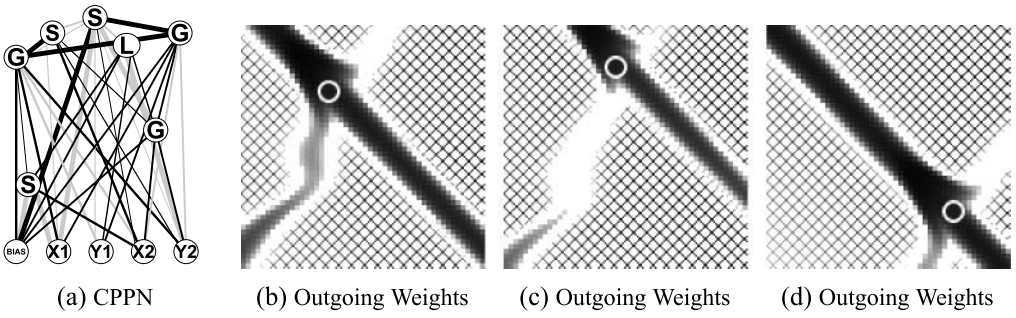


Figure 12. Connectivity motifs of the same substrate at different locations. The CPPN in (a) generates the motifs shown in (b–d), which represent *outgoing connectivity patterns* from a single node in the visual field, whose position is denoted by a small dot (i.e., each frame is a two-dimensional cross section of the four-dimensional hypercube encoded by the CPPN). Note that these patterns, which each originate from only one node, differ from those in Figure 11, which shows *activation patterns* from an entire trial with multiple simultaneous active nodes. The cross-diagonal hatch background represents areas of negative weight, while solid colors between white and black represent increasingly high positive weights. The figure shows that the connective CPPN is able to repeat the same motif across the substrate.

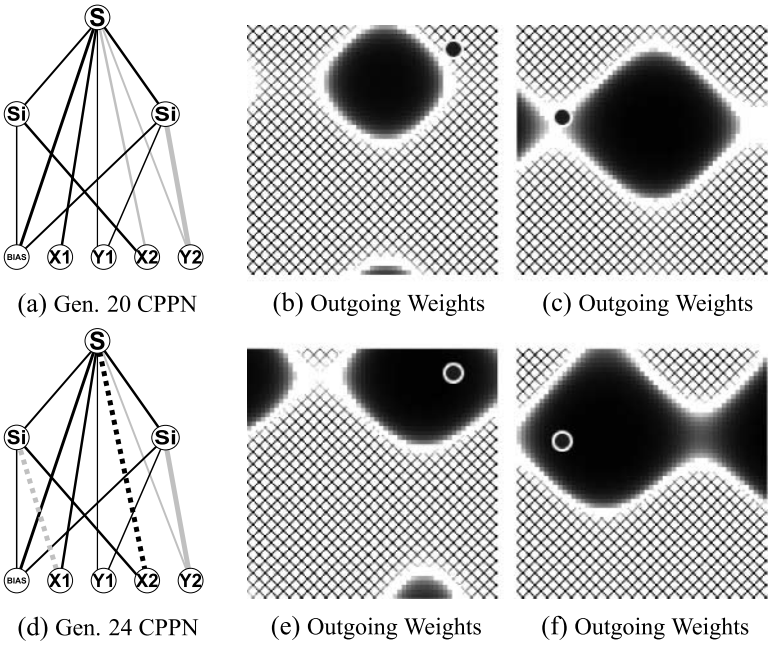


Figure 13. Discovering regularities through CPPN complexification. CPPNs and their respective substrate output are depicted at generations 20 (a–c) and 24 (d–f). As in Figure 12, the connectivity pattern originates in each case from the location of the small dot. The figure shows that the CPPN learned to horizontally calibrate the positions of positive-weighted connections in the substrate by discovering a new connection from x_1 and changing the sign of the connection from x_2. Both connections are highlighted in (d) as dotted lines. Thus, HyperNEAT learns high-level concepts rather than searching for the weight of individual connections in a massive ANN independently. The 13-connection CPPN in (d) produces the 8,644,480-connection substrate in (e) and (f).

generations later (Figure 13d–f), it augments this concept by adding a new connection from x_1 (i.e., its horizontal position) to an internal sine function (Figure 13d). This change recalibrates the horizontal position of the halo center to be closer to the source node. This example explicitly demonstrates the process through which geometric relationships are discovered. In effect, the problem is recast from finding the correct weight of every connection originating from the visual field to a problem of finding the geometric concepts that underlie the solution.

Whereas this experiment has demonstrated HyperNEAT's ability to exploit regularities to optimize very large ANNs, the next experiment focuses on the novel capability to test differing sensor placement schemes in the same task, which is not possible with traditional methods.

6 Experiment 2: Food Gathering

If sensors and outputs are placed so that they respect regularities in the outside world, HyperNEAT can discover those regularities through connective CPPNs and exploit them to solve the problem, as the visual discrimination task demonstrates. Interestingly, there can be more than one way to place inputs and outputs while still respecting the right regularities. The food-gathering task is designed to demonstrate this capability and its implications. This task was chosen for its simplicity as a proof of concept; it effectively isolates the issue of sensor and output placement. In the experiment, two different sensor placement arrangements are compared that present a chance to exploit regularity in different ways.

6.1 Food-Gathering Setup

The food-gathering domain works as follows. A single piece of food is placed within a square room with a robot at the center (as in Figure 7a). A set of n rangefinder sensors, placed at regular angular

intervals, encircle the robot's perimeter. The robot has a compass that allows it to maintain the same orientation at all times, that is, its north-facing side always faces north and never rotates. Internally, the robot also contains a set of n effectors. Each effector, when activated, causes the robot to move in one of n directions, as with the nonrotating robot in Zhao and Jin [61]. Thus, there is one effector for each sensor that points in the same direction. The robot's objective is to go to the food.

The interpretation of effector outputs constrains the problem and the potential solutions. For the experiment in this article, the motion vector resulting from effector output is interpreted to incentivize HyperNEAT to find holistic solutions, that is, solutions that require more than a single connection. The robot moves in the direction corresponding to the largest effector output. In the case of a tie, the robot moves in the direction of the first tied output in sampling order. The robot's speed s is determined by

$$s = (s_{\max} o_{\max}) \left(\frac{o_{\max}}{o_{\text{tot}}} \right), \tag{1}$$

where $s_{\max}$ is the maximum possible speed, $o_{\max}$ is the maximum output, and o_{tot} is the sum of all outputs. The first factor correlates speed with output, so that to go at the maximum speed, the robot must maximize the output corresponding to the direction of the food. The second factor encourages the robot to excite a single output by penalizing it for activating more than one at a time. Furthermore, outputs have sigmoidal activation, which means that if their input is zero, they will output 0.5. Thus, the robot also needs to inhibit effectors that point in the wrong direction, because they will otherwise slow down motion in the chosen direction. Thus, while diverse solutions still work in this domain, many are not optimal in terms of speed. The best solutions require a correct pattern connecting to all the outputs from all the sensors.

It is important to note that the compass-based robot is intentionally designed not to rotate, so that it is forced to learn a separate output for each discrete direction. That way, HyperNEAT's ability to geometrically correlate inputs to outputs and scale the resolution of both inputs and outputs can be tested explicitly.

Each robot attempts r trials, where r is twice the resolution; thus higher resolutions are evaluated on more trials. For each trial a single piece of food is placed 100 units away from the robot at either the center of a sensor or the border between two sensors. Each trial tests a different such location. If a robot is not able to get food for a particular trial after 1,000 ticks, its trial ends. Individuals are evaluated based on their amount of food collected and the average speed at which they obtain each item:

$$\text{fitness} = 10,000 \frac{f_c}{r} + t_{\text{tot}} \cdot 1,000r, \tag{2}$$

where f_c is the total amount of food collected and t_{tot} is the total time spent on all trials.

This task is a good proof of concept because it transparently requires discovering the underlying regularity of the domain: Two nodes at the same angle (the sensor and the effector) should be connected, and the others inhibited. If this concept is discovered, the task is effectively trivial. Because HyperNEAT can discover the general concept, it can use it to solve the task efficiently. Demonstrating this fact helps to explicate how HyperNEAT works.

6.2 Sensor Placement

Two different sensor placements and substrate configurations are attempted that capture the key correlation in different ways:

(1) *Two concentric circles of nodes (Figure 7b).* The inner circle (radius 0.5) is the sensors, and the outer circle (radius 1.0) is the effectors. The nodes are placed at the same angle as in the robot. In this layout, the key regularity is captured by shared angle. It is also interesting in that the sensors and

effectors are placed exactly in the shape of the robot, an intuitive scheme that would not be meaningful in traditional methods.

(2) *Two parallel lines of nodes (Figure 7c).* The top row of sensors are placed in clockwise order, starting from the sensor closest to 45° above west. In the bottom row, effectors are placed in the same order. In this way, the key regularity is geometrically expressed as two nodes being in the same column.

6.3 Scaling

Similar to the visual discrimination task, the numbers of inputs and outputs of each generation's champion from all runs of both configurations are doubled several times, starting from the evolved 8×8 resolution, without further evolution. Each double-resolution substrate samples twice as many points as its predecessor by decreasing the angular sampling interval around the robot by half. Doubling for each champion proceeds from the initial 8×8 resolution to a maximum size of 128×128, a 16-fold increase in resolution. As a consequence, few of the angles present at the prior resolution are sampled at the new resolution. This disparity further requires the CPPN to learn a general connectivity concept, rather than a static mapping between points.

6.4 Additional Geometric Bias in Food Gathering

As in visual discrimination, the distance between two nodes is an important factor in their relationship. Therefore, in food gathering, because there are two differing substrate configurations, the Euclidean distance between (x_1, y_1) and (x_2, y_2) is input in addition to the usual x_1, y_1, x_2, and y_2. As with visual discrimination, the benefit of such additional inputs is investigated by comparing results with and without such inputs.

Experimental parameters for food gathering are described in the Appendix. The next sections describe the results from the food-gathering experiment.

7 Food-Gathering Results

All sensor configurations were able to collect food at all positions within the first few generations except for the concentric case without the Euclidean-distance input, which took on average 33 generations to learn how to get food at every position. Thus, for most configurations, the main challenge was to learn to get food *efficiently*. The performance measure in this section is thus the average time (i.e., number of ticks) it takes the robot to get a piece of food over all its trials. Robots that cannot get the food in a trial are assigned the maximum time 1,000 for that trial. Results are averaged over 20 runs.

Figure 14a shows how performance improved over 500 generations for both placement schemes, with and without the Euclidean-distance-input geometric bias. Parallel placement on average evolved significantly faster strategies ($p < .05$) than concentric. This disparity is explained by the more complex relationship, in Cartesian coordinates, between corresponding nodes in the concentric case. However, the added geometric information in this experiment significantly increased performance of both methods after the fourth generation ($p < .01$). Furthermore, the Euclidean-distance input is so useful that it erases the difference between the two schemes, causing them to perform similarly when present. Thus, parallel placement is easier to exploit for HyperNEAT except when the CPPN is provided the connection length as input.

7.1 Scaling Performance

As in visual discrimination, substrates were scaled and tested without further evolution. Individuals generally did retain the ability to collect food, although, unlike the case of visual discrimination, there is some degradation in performance at each increment in resolution. To illustrate this degradation for different configurations, Figure 14b shows the average difference in efficiency between resolution 8 and resolution 32; lower numbers imply better ability to scale.

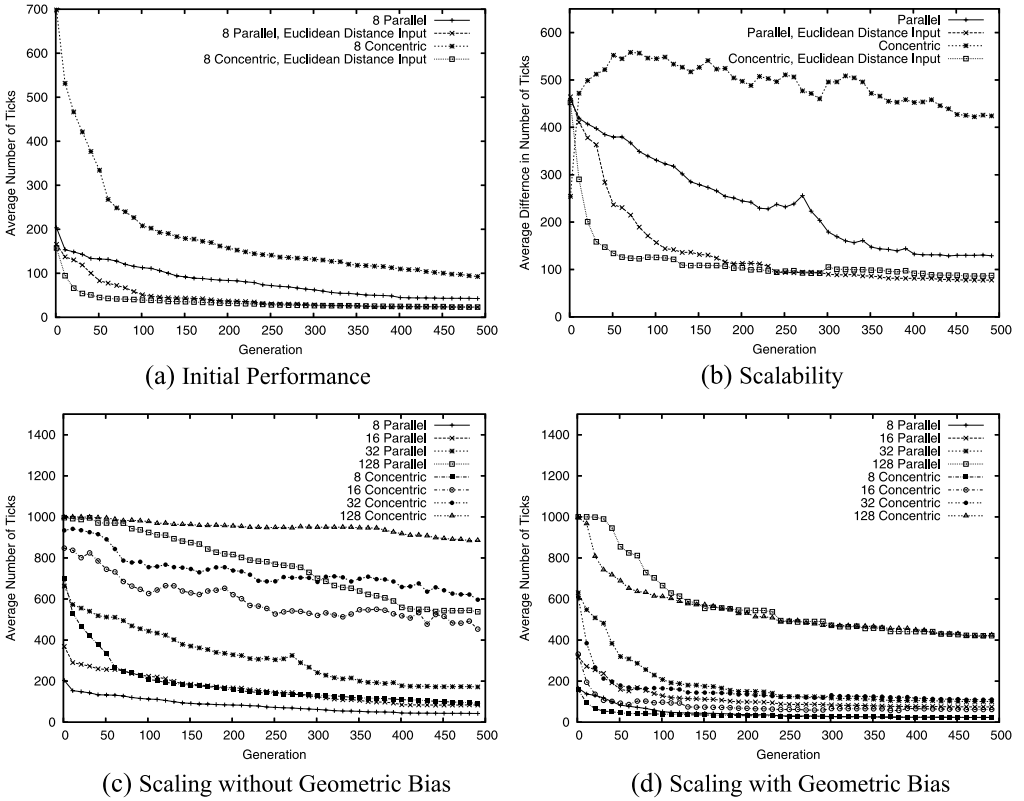


Figure 14. HyperNEAT food-gathering performance. The performance of both sensor layouts at resolution 8, with and without the extra Euclidean-distance input, is shown in (a). The difference in speeds when different methods are scaled from 8 to 32 is shown in (b). Graphs (c) and (d) show the speeds of the two sensor placement schemes at all resolutions, with and without the distance input (i.e., bias toward exploiting locality), respectively. The conclusion is that HyperNEAT learns to exploit the placement geometry.

Parallel placement scaled significantly more effectively than concentric except in the earliest generations ($p < .01$). However, as with efficiency, when concentric placement's CPPN was provided length as input, it scaled as well as parallel placement did with length input. In both cases with the extra input, scaling significantly improved over runs using concentric placement without the additional input ($p < .01$), but not significantly over those using regular parallel placement.

As Figure 14b shows, concentric placement without the length input degraded significantly between resolution 8 and 32; in fact, individuals could no longer collect food at every position. However, it turns out that information about the task was still retained implicitly at the higher resolution: When allowed to continue evolving at the higher resolution, solutions that collect all the food were always found within five generations (2.5 on average). On the other hand, when concentric evolution is started from scratch at a *lower* resolution, it takes on average 33 generations to learn to get food on every trial. Thus, even when performance degrades significantly after scaling, the resultant individual still retains important geometric information that can be quickly tweaked to work at the higher resolution.

Figure 14c shows the average absolute performance at different resolutions for CPPNs without the additional input, and 14d shows the same comparison for those with it. Parallel placement consistently outperformed concentric at the same resolution (Figure 14c). Again, however, when the length input was provided (Figure 14d), the performance of the two placement schemes no longer differed significantly. Although each increment in resolution leads to a graceful degradation in performance, scaled individuals at all resolutions and in all configurations significantly outperformed a

set of random individuals, showing further that scaling indeed retains abilities present in the lower-resolution network ($p < .01$ versus random). Furthermore, although the average time degrades, most scaled networks could still collect all the food if their unscaled predecessor could.

In a more dramatic demonstration of scaling, one high-fitness individual of each sensor placement scheme was scaled to a resolution of 1024. The resulting ANNs each had over one million connections and could still gather all the food.

An important question that emerges from the scaling results on food gathering is why substrates do not scale perfectly as in visual discrimination. This question is addressed in the following analysis and revisited later in the discussion (Section 8).

7.2 Repeating Motifs

To consistently activate highly the one output pointing in the right direction and inhibit all others requires discovering such a motif and repeating it across the substrate. Figure 15 shows several examples of this motif at different locations in both concentric and parallel substrates. The benefit of CPPN representation is that it need only discover the correct motif once by exploiting how it relates to the geometry of inputs and outputs.

The CPPN in Figure 15a (top row) shows how such exploitation is possible. HyperNEAT discovered that by connecting x_1 to a Gaussian node with a positive weight and x_2 to the same node with a negative weight, the weight connecting any two nodes in the substrate is made a function of horizontal distance. This principle is all that is necessary to cause the same motif to appear at every locus of horizontal correlation.

On the other hand, the principle of radial symmetry that underlies the correct motif in concentric substrates is more challenging. Angular differences necessary to exploit concentric regularity take more structure to compute, although the concept is eventually discovered.

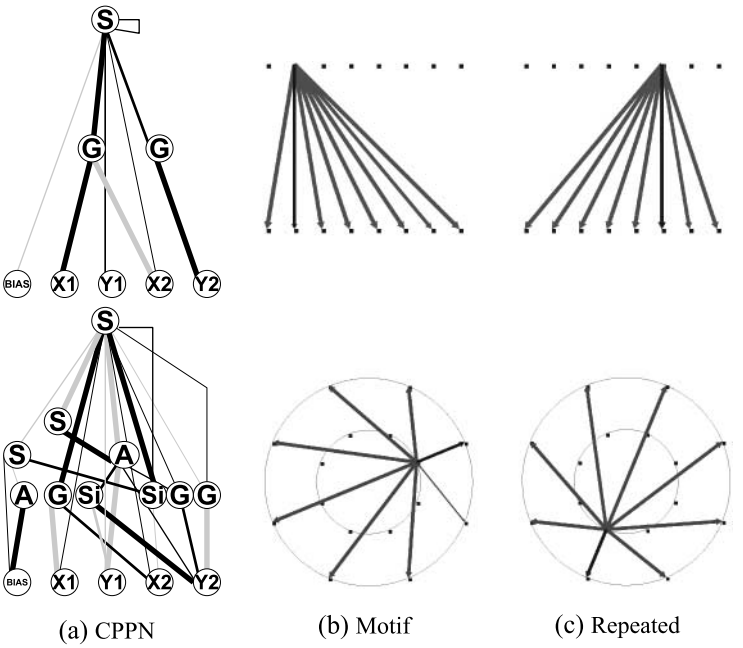


Figure 15. Repeated patterns in solutions. The CPPNs in (a) represent parallel (top row) and concentric (bottom row) substrates that solve the task. The connectivity patterns in (b) and (c) are outgoing motifs, each from a single input node in the substrate. These images show that the same motif is repeated in different locations.

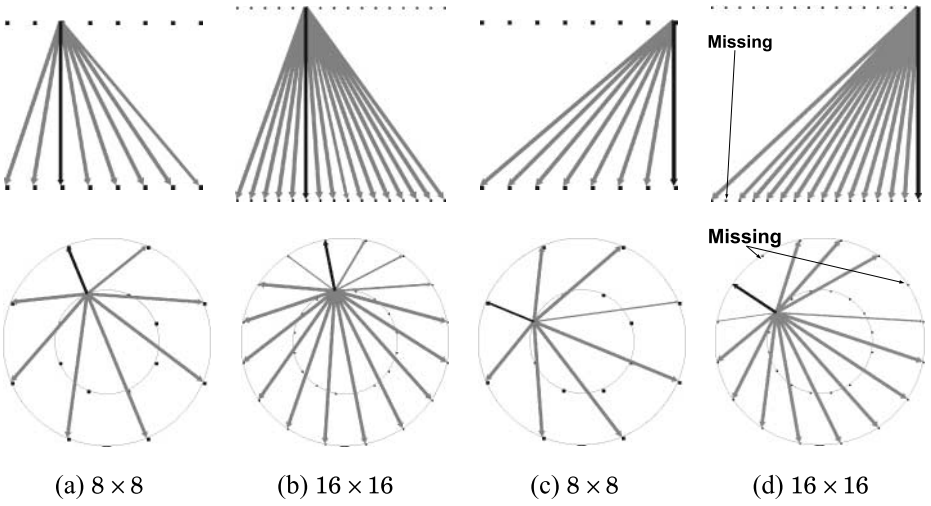


Figure 16. Perfect and imperfect scaling. The parallel (top row) and concentric (bottom row) 8 × 8 motifs in (a) scale perfectly to the 16 × 16 motifs in (b). However, because of artifacts at higher resolution, not all motifs scale perfectly, as shown by comparing (c) and (d), in which missing connections are identified by arrows. While the motif is mostly intact, slight imperfections of this type are common during scaling in this task.

7.3 Scaling Analysis

Figure 16a,b shows evolved parallel and concentric motifs at both 8 × 8 and 16 × 16 resolution.

Scaling up can sometimes produce small artifacts that are not apparent at lower resolutions (Figure 16c,d). These imperfections vary greatly, as do their effects on performance, although in all cases sufficient information was retained to quickly recover the old behavior in less than five generations.

Analyzing the source of higher-resolution imperfections explains how CPPNs represent regularities. Recall that a connective CPPN is in effect a pattern within a four-dimensional hypercube. If two of those dimensions, say x_1 and y_1, are fixed, then the remaining two dimensions define a two-dimensional cross section of the hypercube. The pattern within that cross section is in effect the infinite-resolution connectivity pattern for one input node from which all finite substrate resolutions are sampled. Figure 17 depicts two such cross sections for both concentric and parallel substrates. The figure shows that the CPPN constructs a *pattern* that, when sampled at the right resolution, defines the connectivity weights of all connections between the fixed source location and the sample point.

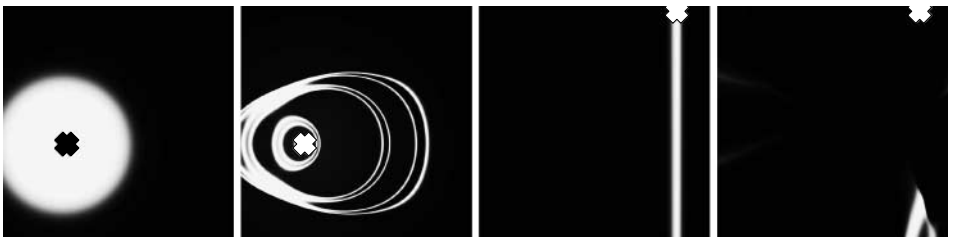


Figure 17. Hypercube cross sections. Typical (a and c) and atypical (b and d) two-dimensional cross sections of hypercubes generated by CPPNs are depicted for working concentric (a and b) and parallel (c and d) substrate configurations. Each cross section represents the infinite-resolution outgoing connectivity pattern originating from the location of the ×. When substrates are generated in practice, these cross sections are sampled at the substrate resolution. Thus, much of the detail in the patterns is discarded for low-resolution substrates, though it may reemerge when scaling to higher resolution.

Because only the sample points are required to be correct during evaluation, the unsampled portion of the pattern can exhibit artifacts that are not visible at the sample resolution, but may become so at higher resolutions. Interestingly, however, the pattern often elegantly captures the fundamental geometric relationship, as in Figure 17a and 17c, which explains why scaling sometimes is perfect or close to perfect.

In contrast to food gathering, scaled CPPNs in visual discrimination performed almost perfectly, with statistically insignificant degradation across all resolutions. What distinguishes the two tasks? An important difference between them is that the precision of the substrate in visual discrimination does not need to increase, because the sizes of the boxes remain the same relative to the size of the substrate. In food gathering, on the other hand, the size of the food is in effect shrinking when the resolution goes up, because food can only be detected by a single sensor and there are more sensors at higher resolution. Therefore, in food gathering, in order to scale perfectly, the precision of the solution would need to increase with the resolution. Because the task is never evaluated at higher resolution during training, it is not possible to ensure such an increase through training.

Thus, one conclusion on scaling is that, depending on the task, it may be perfect or imperfect; however, the more important conclusion is that it always retains at least what was known at the lower resolution, which is significantly better than starting from scratch if evolution is to continue at the higher resolution. Furthermore, a useful result of these experiments is that there is now a basis on which to predict what kinds of tasks will scale perfectly.

7.4 Evolving Motifs

Rather than finding the value of each connection separately, evolution in HyperNEAT progresses by discovering global regularities. Figure 18 traces progress over parallel and concentric runs of evolution. The early solutions are simple linear combinations of the coordinates. Although inefficient, these patterns learn to gather food through a few good connections that compensate for others. Later in evolution, symmetries and regularities begin to be discovered, although they are not always the most fundamental ones for the task. For example, the concentric substrate in Figure 18b partially solves the task by exploiting its bilateral symmetry, even though the task is more fundamentally radially symmetric. Finally, evolution discovers the essential regularity of the task domain, which is horizontal distance for parallel substrates and angular disparity in concentric substrates (Figure 18c).

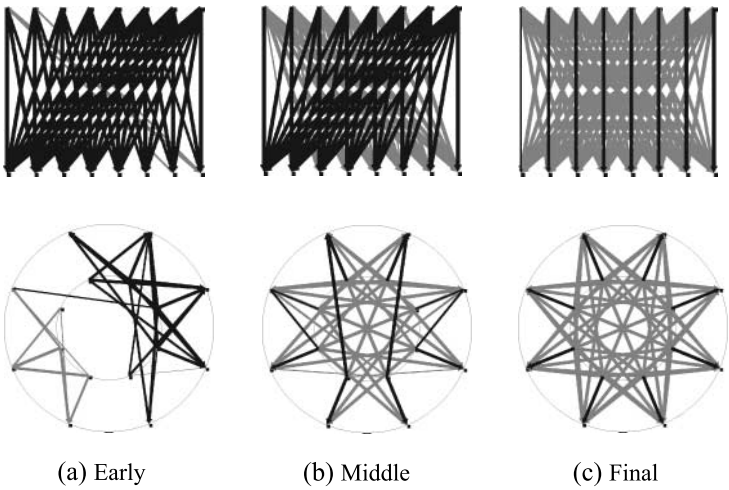


Figure 18. Evolution discovers regularities. Early-generation connective CPPNs typically produce simple connectivity patterns (a). Eventually, HyperNEAT begins to exploit regularities (b), though they may not be the most fundamental ones. Finally, HyperNEAT discovers the fundamental regularity that underlies the task for the given substrate configuration (c). Thus, instead of optimizing individual connection weights, evolution is a process of gradually discovering and exploiting holistic regularities.

The next section discusses the major ramifications of this work.

8 Discussion and Future Work

The central insight in HyperNEAT is that regularity in higher-dimensional spatial patterns is easily mapped to regular connectivity in lower-dimensional space, creating a new kind of indirect encoding for ANNs. Experimental results show how it is possible to encode up to millions of connections by discovering the pattern of recurring motifs underlying a solution. This section explores the implications of this capability and its underlying methodology.

8.1 Limitations of Direct Encoding

The comparison between PNEAT and HyperNEAT in visual discrimination highlights that direct encodings are vulnerable to high dimensionality (i.e., a large number of connections), regardless of underlying regularities in the domain. Thus, like other direct encodings, PNEAT can improve only by discovering each connection strength individually.

While previous research has confirmed the advantage of indirect encoding over direct encoding [5, 14, 25, 43, 52], the comparison with PNEAT gives a sense of its urgency in neuroevolution. If PNEAT cannot optimize a network of 14,641 connections to discover a simple repeating motif, it is evident that tasks requiring more than 15,000 connections are going to require some form of indirect encoding.

In contrast, HyperNEAT demonstrates how it is possible to optimize orders of magnitude beyond 10,000 connections. Because connective CPPNs represent the solution conceptually, a single solution in effect represents an infinite range of resolutions (i.e., node densities). Thus, the same solution can scale to higher resolutions, which is a new capability for even indirect encodings of ANNs.

8.2 Substrate Scaling Implications

Visual discrimination solutions suffer no statistically significant degradation in performance as they scale. However, scaling results on food gathering demonstrate that solutions cannot always be expected to scale perfectly. This conclusion is not surprising; in tasks like food gathering scaling requires not only higher resolution, but higher precision, because the detail present in the additional substrate is germane to the solution. A candidate trained at lower resolution has no means to evaluate such details and therefore may contain artifacts when they are brought into greater focus.

Nevertheless, while not always perfect, the ability to scale without further evolution is a significant advance for neuroevolution. The important consequence is that information learned at the lower resolution is retained at the higher resolution; it is only within the expanded portions that error may be introduced. Thus, the real benefit of this capability is that in the future, further evolution can be performed at the higher resolution. Instead of starting over from scratch, HyperNEAT can continue to build on lower-resolution solutions after they are magnified. The numbers of sensors, hidden nodes, and outputs can be multiplied without losing prior knowledge, because CPPNs in effect decouple solutions from individual inputs and outputs. Such scaling may prove an important tool for ultimately achieving solutions with a million or more connections to problems that require such high dimensionality to solve. For example, a promising application of substrate scaling is machine vision, wherein general regularities in the structure of images can be learned at low resolution and then scaled up.

Scaling the substrate connectivity concept suggests an interesting high-level explanation for how brains evolved increasing complexity in nature. It is implausible that the human brain evolved by one new connection at a time from one generation to the next. How then could complexity have increased over generations? An interesting hypothesis is that, at a high level of abstraction, the evolution of brains in biology in effect included several such increases in density on the same connectivity concept. Not only can such an increase improve the immediate resolution of sensation and action, but it can provide additional substrate for increasingly intricate local relationships to be

discovered through further evolution. If neural connectivity patterns are represented in DNA as concepts, then such increases would be possible, thereby creating new opportunities for further structural elaborations to evolve within the higher-resolution substrate. Connective CPPNs are a concrete formal explanation of how such complexification can work at a high level of abstraction.

8.3 Substrate Configuration and Geometry

The food-gathering task demonstrates that in contrast to regular ANNs, different substrate placement schemes create geometric relationships that are exploitable in different ways, some more easily than others. Connective CPPNs exploit regularities in parallel placement more efficiently than in concentric, because the angular differences necessary to exploit concentric regularity take more structure to compute with the given set of activation functions. Thus the activation functions in the CPPN are like a *language* that describes geometric structure. With the right words, it is easier to describe a particular relationship. HyperNEAT allows the experimenter to inject knowledge into the search through such configurations—for example, by grouping correlated objects or arranging sensors analogously to their real-world geometry.

Furthermore, additional geometric information provided as input to the connective CPPN can significantly simplify the structure necessary to describe key motifs. When connection length is provided as input, the two placement schemes produce equivalent results. This result is important because it shows that connective CPPNs allow the experimenter to provide hints to the learning algorithm about the kinds of geometric principles that may underlie the solution.

It is important to note that connectivity patterns can include hidden nodes and recurrence. Just as a subset of nodes in the substrate are assigned as inputs and another subset as outputs, a third subset can be reserved as hidden nodes. Regular patterns of hidden nodes, like cortical columns in the human brain [47], can potentially be discovered.

Thus more complex problems requiring hidden nodes are an important future research direction. For example, board games like Checkers [16] and Go [54] likely require hidden nodes to permit effective strategies. Such games exhibit numerous tactical regularities across different positions on the board that connective CPPNs can potentially encode efficiently.

Recurrence, on the other hand, is naturally expressed by simply querying connections in the substrate that would be recurrent. HyperNEAT can potentially evolve massive recurrent structures such as continuous-time recurrent neural networks (CTRNNs) that control walking gaits for diverse animal morphologies [41, 42, 56]. This application is promising because gaits and body morphologies are often highly regular, with left and right sides related and one leg's timing related to that of others.

It is also possible that non-neural connectivity patterns can be evolved in the future (e.g., to describe robot body morphologies). Furthermore, because CPPNs can describe spatial patterns [49] in addition to connectivity patterns, the two types of pattern may be combined in the future.

9 Conclusion

This article has addressed how it is possible to represent and discover large-scale ANNs. The suggested approach searches for regularities in the task domain by representing the solution as a function of the task geometry. That way, it is possible to generate ANNs with millions of connections based on underlying geometric motifs. To demonstrate this approach, HyperNEAT evolved connective CPPNs that solve simple visual discrimination and food-gathering tasks at varying resolutions. Both tasks required HyperNEAT to discover a repeating motif in neural connectivity. Furthermore, because CPPN encoding is independent of the number of sensors, solutions could be scaled to variable resolutions. Most importantly, HyperNEAT suggests how someday it may be possible to tackle problems that heretofore have proven too complex for modern machine learning methods because of their astronomically high dimensionality. HyperNEAT shifts the locus of difficulty away from problem dimensionality to problem structure, thereby providing hope that such domains may ultimately be conquered by discovering their underlying regularities.

Acknowledgments

All the software and source code used in this article is available through http://eplex.cs.ucf.edu.

References

1. Altenberg, L. (1994). Evolving better representations through selective genome growth. In *Proceedings of the IEEE World Congress on Computational Intelligence* (pp. 182–187). Piscataway, NJ: IEEE Press.
2. Angeline, P. J. (1995). Morphogenic evolutionary computations: Introduction, issues and examples. In J. R. McDonnell, R. G. Reynolds, & D. B. Fogel (Eds.), *Evolutionary Programming IV: The Fourth Annual Conference on Evolutionary Programming* (pp. 387–401). Cambridge, MA: MIT Press.
3. Angeline, P. J., Saunders, G. M., & Pollack, J. B. (1993). An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks*, *5*, 54–65.
4. Belew, R. K., & Kammeyer, T. E. (1993). Evolving aesthetic sorting networks using developmental grammars. In S. Forrest (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms*. San Francisco: Kaufmann.
5. Bentley, P. J., & Kumar, S. (1999). The ways to grow designs: A comparison of embryogenies for an evolutionary design problem. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999)* (pp. 35–43). San Francisco: Kaufmann.
6. Bongard, J. C. (2002). Evolving modular genetic regulatory networks. In *Proceedings of the 2002 Congress on Evolutionary Computation*.
7. Chklovskii, D. B., & Koulakov, A. A. (2004). Maps in the brain: What can we learn from them? *Annual Review of Neuroscience*, *27*, 369–392.
8. Churchland, P. M. (1986). Some reductive strategies in cognitive neurobiology. *Mind*, *95*, 279–309.
9. Dawkins, R. (1986). *The blind watchmaker*. Essex, UK: Longman.
10. Dellaert, F. (1995). *Toward a biologically defensible model of development*. Master's thesis, Case Western Reserve University, Cleveland, OH.
11. Dellaert, F., & Beer, R. D. (1996). A developmental model for the evolution of complete autonomous agents. In P. Maes, M. J. Mataric, J.-A. Meyer, J. Pollack, & S. W. Wilson (Eds.), *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*. Cambridge, MA: MIT Press.
12. Deloukas, P., Schuler, G. D., Gyapay, G., Beasley, E. M., Soderlund, C., Rodriguez-Tome, P., Hui, L., Matise, T. C., McKusick, K. B., Beckmann, J. S., Bentolila, S., Bihoreau, M., Birren, B. B., Browne, J., Butler, A., Castle, A. B., Chiannilkulchai, N., Clee, C., Day, P. J., Dehejia, A., Dibling, T., Drouot, N., Duprat, S., Fizames, C., & Bentley, D. R. (1998). A physical map of 30,000 human genes. *Science*, *282*(5389), 744–746.
13. Eggenberger, P. (1997). Evolving morphologies of simulated 3D organisms based on differential gene expression. In P. Husbands & I. Harvey (Eds.), *Proceedings of the Fourth European Conference on Artificial Life* (pp. 205–213). Cambridge, MA: MIT Press.
14. Federici, D. (2004). Evolving a neurocontroller through a process of embryogeny. In S. Schaal, A. J. Ijspeert, A. Billard, S. Vijayakumar, J. Hallam, & J.-A. Meyer (Eds.), *Proceedings of the Eighth International Conference on Simulation and Adaptive Behavior (SAB-2004)* (pp. 373–384). Cambridge, MA: MIT Press.
15. Federici, D. (2004). Using embryonic stages to increase the evolvability of development. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004) Workshop Program*. Berlin: Springer-Verlag.
16. Fogel, D. B. (2001). *Blondie24: Playing at the Edge of AI*. San Francisco: Kaufmann.
17. Gilbert, C. D., & Wiesel, T. N. (1992). Receptive field dynamics in adult primary visual cortex. *Nature*, *356*, 150–152.
18. Gomez, F., & Miikkulainen, R. (1999). Solving non-Markovian control tasks with neuroevolution. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence* (pp. 1356–1361). San Francisco: Kaufmann.
19. Goodhill, G. J., & Carreira-Perpinn, M. A. (2002). Cortical columns. In L. Nadel (Ed.), *Encyclopedia of cognitive science*, Vol. 1 (pp. 845–851). London: MacMillan.
20. Green, C. (2003–2006). SharpNEAT homepage. http://sharpneat.sourceforge.net/.
21. Gruau, F. (1994). *Neural network synthesis using cellular encoding and the genetic algorithm*. Ph.D. thesis, Ecole Normale Superieure de Lyon, France.

22. Gruau, F., Whitley, D., & Pyeatt, L. (1996). A comparison between cellular encoding and direct encoding for genetic neural networks. In J. R. Koza, D. E. Goldberg, D. B. Fogel, & R. L. Riolo (Eds.), *Genetic Programming 1996: Proceedings of the First Annual Conference* (pp. 81–89). Cambridge, MA: MIT Press.

23. Hart, W. E., Kammeyer, T. E., & Belew, R. K. (1994). *The role of development in genetic algorithms* (Technical Report CS94-394). University of California, San Diego.

24. Harvey, I. (1993). *The artificial evolution of adaptive behavior*. Ph.D. thesis, School of Cognitive and Computing Sciences, University of Sussex.

25. Hornby, G. S., & Pollack, J. B. (2001). The advantages of generative grammatical encodings for physical design. In *Proceedings of the 2002 Congress on Evolutionary Computation*.

26. Hornby, G. S., & Pollack, J. B. (2002). Creating high-level components with a generative representation for body-brain evolution. *Artificial Life*, *8*(3), 223–246.

27. Hubel, D. H., & Wiesel, T. N. (1965). Receptive fields and functional architecture in two nonstriate visual areas (18 and 19) of the cat. *Journal of Neurophysiology*, *28*, 229–289.

28. Jakobi, N. (1995). Harnessing morphogenesis. In *Proceedings of the Second International Workshop on Information Processing in Cells and Tissues* (pp. 29–41).

29. James, D., & Tucker, P. (2005). Evolving a neural network active vision system for shape discrimination. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2005) Late Breaking Papers*. New York: ACM Press.

30. Kandel, E. R., Schwartz, J. H., & Jessell, T. M. (Eds.). (1991). *Principles of neural science* (3rd ed.). Amsterdam: Elsevier.

31. Kohl, N., Stanley, K., Miikkulainen, R., Samples, M., & Sherony, R. (2006). Evolving a real-world vehicle warning system. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2006)* (pp. 1681–1688).

32. Komosinski, M., & Rotaru-Varga, A. (2001). Comparison of different genotype encodings for simulated 3D agents. *Artificial Life*, *7*(4), 395–418.

33. Lindenmayer, A. (1968). Mathematical models for cellular interaction in development: Parts I, II. *Journal of Theoretical Biology*, *18*, 280–299, 300–315.

34. Lindenmayer, A. (1974). Adding continuous components to L-systems. In G. Rozenberg & A. Salomaa (Eds.), *L Systems, Lecture Notes in Computer Science 15* (pp. 53–68). Heidelberg: Springer-Verlag.

35. Martin, A. P. (1999). Increasing genomic complexity by gene duplication and the origin of vertebrates. *The American Naturalist*, *154*(2), 111–128.

36. Miller, J. F. (2004). Evolving a self-repairing, self-regulating, French flag organism. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004)*. Berlin: Springer-Verlag.

37. Mjolsness, E., Sharp, D. H., & Reinitz, J. (1991). A connectionist model of development. *Journal of Theoretical Biology*, *152*, 429–453.

38. Moriarty, D., & Miikkulainen, R. (1993). *Evolving complex Othello strategies with marker-based encoding of neural networks* (Technical report AI93-206). Department of Computer Sciences, The University of Texas at Austin.

39. Prusinkiewicz, P., & Lindenmayer, A. (1990). *The algorithmic beauty of plants*. Heidelberg: Springer-Verlag.

40. Raff, R. A. (1996). *The shape of life: Genes, development, and the evolution of animal form*. Chicago: University of Chicago Press.

41. Reil, T., & Husbands, P. (2002). Evolution of central pattern generators for bipedal walking in a real-time physics environment. *IEEE Transactions on Evolutionary Computation*, *6*(2), 159–168.

42. Reil, T., & Massey, C. (2001). Biologically inspired control of physically simulated bipeds. *Theory in Biosciences*, *120*, 1–13.

43. Reisinger, J., Stanley, K. O., & Miikkulainen, R. (2005). Towards an empirical measure of evolvability. In *Genetic and Evolutionary Computation Conference (GECCO2005) Workshop Program* (pp. 257–264). Washington, DC: ACM Press.

44. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In *Parallel distributed processing* (pp. 318–362). Cambridge, MA: MIT Press.

45. Saravanan, N., & Fogel, D. B. (1995). Evolving neural control systems. *IEEE Expert*, *10*(3), 23–27.

46. Sims, K. (1994). Evolving 3D morphology and behavior by competition. In R. A. Brooks & P. Maes (Eds.), *Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems (Artificial Life IV)* (pp. 28–39). Cambridge, MA: MIT Press.

47. Sporns, O. (2002). Network analysis, complexity, and brain function. *Complexity*, *8*(1), 56–60.

48. Stanley, K. O. (2006). Exploiting regularity without development. In *Proceedings of the AAAI Fall Symposium on Developmental Systems*. Menlo Park, CA: AAAI Press.

49. Stanley, K. O. (2007). Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines*, *8*(2), 131–162.

50. Stanley, K. O., Bryant, B. D., & Miikkulainen, R. (2005). Real-time neuroevolution in the NERO video game. *IEEE Transactions on Evolutionary Computation*, *9*(6), 653–668.

51. Stanley, K. O., & Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, *10*, 99–127.

52. Stanley, K. O., & Miikkulainen, R. (2003). A taxonomy for artificial embryogeny. *Artificial Life*, *9*(2), 93–130.

53. Stanley, K. O., & Miikkulainen, R. (2004). Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, *21*, 63–100.

54. Stanley, K. O., & Miikkulainen, R. (2004). Evolving a roving eye for Go. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004)*. Berlin: Springer-Verlag.

55. Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

56. Terzopoulos, D., Rabie, T., & Grzeszczuk, R. (1994). Artificial fishes: Autonomous locomotion, perception, behavior, and learning in a simulated physical world. *Artificial Life*, *1*(4), 327–351.

57. Turing, A. (1952). The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society B*, *237*, 37–72.

58. Watson, J. D., Hopkins, N. H., Roberts, J. W., Steitz, J. A., & Weiner, A. M. (1987). *Molecular biology of the gene* (4th ed.). Menlo Park, CA: Benjamin Cummings.

59. Wolpert, D. H., & Macready, W. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, *1*, 67–82.

60. Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE*, *87*(9), 1423–1447.

61. Zhao, D., & Jin, W. (2005). The study of cooperative behavior in predator-prey problem of multi-agent systems. In *Proceedings of the International Symposium on Autonomous Decentralized Systems (ISADS 2005)* (pp. 90–96). Piscataway, NJ: IEEE Press.

62. Zigmond, M. J., Bloom, F. E., Landis, S. C., Roberts, J. L., & Squire, L. R. (Eds.) (1999). *Fundamental neuroscience*. London: Academic Press.

Appendix: Experimental Parameters

Because HyperNEAT extends NEAT, it uses similar parameters [51]. Furthermore, extending a given NEAT implementation to HyperNEAT requires minimal effort because HyperNEAT fits on top of the existing NEAT algorithm. Thus, the two experiments in this article were performed with different NEAT packages, each of which came already equipped with preexisting facilities for one experiment. The visual discrimination task utilizes a custom version of NEAT developed by Jason Gauci at the University of Central Florida that includes optimizations for processing visual input and output. The food-gathering experiment is implemented in Colin Green's public domain SharpNEAT package [20], which includes infrastructure for robot control experiments. The two NEAT packages differ slightly in their implementations, and the two experiments were performed separately; therefore they use slightly different parameters. However, both packages preserve the core NEAT algorithm and with it evolve CPPNs. Both packages, including HyperNEAT extensions, are available at http://eplex.cs.ucf.edu.

Table 1 shows the parameters used in both implementations of HyperNEAT. In addition, if the magnitude of the CPPN's output for a particular query is less than or equal to 0.2, then the connection is not expressed in the substrate. If it is greater than 0.2, then the number is scaled to a magnitude between 0 and 3. The sign is preserved, so negative output values correspond to negative connection weights.

While some parameters differ slightly, they follow similar logic, that is, the parameters' relative values are similar within the systems. NEAT has been found to be robust to moderate variations in parameters [50, 51, 53].

Table 1. Parameter settings. Gauci HyperNEAT was applied to visual discrimination, and food gathering was evolved in SharpNEAT HyperNEAT.

	Value	
Parameter	Gauci HyperNEAT	SharpNEAT HyperNEAT
Pop. size	100	150
c_1	2.0	2.0
c_2	2.0	2.0
c_3	1.0	0.2
c_t	Variable	Variable
Prob. add link	0.1	0.03
Prob. add node	0.03	0.01
Target no. of species	8	10 to 15
Elitism	20%	20%
CPPN output range	−1 to 1	−1 to 1
CPPN weight range	−3 to 3	−3 to 3
Function set	Gaussian, sigmoid, sine, bounded linear	Gaussian, sigmoid, sine, absolute value