

Generating Large-Scale Neural Networks Through Discovering Geometric Regularities

Jason Gauci
Evolutionary Complexity Research Group
School of EECS
University of Central Florida
Orlando, FL 32836
jgauci@cs.ucf.edu

Kenneth Stanley
Evolutionary Complexity Research Group
School of EECS
University of Central Florida
Orlando, FL 32836
kstanley@cs.ucf.edu

ABSTRACT

Connectivity patterns in biological brains exhibit many repeating motifs. This repetition mirrors inherent geometric regularities in the physical world. For example, stimuli that excite adjacent locations on the retina map to neurons that are similarly adjacent in the visual cortex. That way, neural connectivity can exploit geometric locality in the outside world by employing local connections in the brain. If such regularities could be discovered by methods that evolve artificial neural networks (ANNs), then they could be similarly exploited to solve problems that would otherwise require optimizing too many dimensions to solve. This paper introduces such a method, called Hypercube-based Neuroevolution of Augmenting Topologies (HyperNEAT), which evolves a novel generative encoding called connective Compositional Pattern Producing Networks (connective CPPNs) to discover geometric regularities in the task domain. Connective CPPNs encode connectivity patterns as *concepts* that are independent of the number of inputs or outputs, allowing functional large-scale neural networks to be evolved. In this paper, this approach is tested in a simple visual task for which it effectively discovers the correct underlying regularity, allowing the solution to both generalize and scale without loss of function to an ANN of over eight million connections.

Categories and Subject Descriptors:

I.2.6[Artificial Intelligence]: Learning—*Concept Learning, Connectionism and neural nets*

C.2.1[Computer-Communication Networks]: Network Architecture and Design—*Network Topology*

General Terms: Experimentation, Algorithms

Keywords: Compositional Pattern Producing Networks, NEAT, HyperNEAT, large-scale artificial neural networks

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

GECCO'07, July 7–11, 2007, London, England, United Kingdom.
Copyright 2007 ACM 978-1-59593-697-4/07/0007 ...\$5.00.

1. INTRODUCTION

How did evolution, an unguided search process, discover structures as astronomically complex and sophisticated as the human brain? An intriguing possibility is that physical properties of the universe constrain the space of physical cognitive structures and the problems they encounter to make such discovery likely. In particular, the physical geometry of space provides a bias toward brain-like solutions.

The geometry of space is effectively three-dimensional Cartesian. An ubiquitous useful property that results from Cartesian geometry is the principle of locality. That is, what happens at one coordinate in space is often related to what happens in adjacent locations. This fact is exploited effectively in biological brains, wherein the physical world is often projected onto Cartesian arrays of neurons [5]. Just by virtue of the neurons that are related to nearby events in space also being near each other, it becomes likely that 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. That is, the properties of physical space inherently bias cognitive organization towards local connectivity, which happens to be useful for solving problems that are projected from the physical world. Thus it may not be simply through brute-force search that the human brain employs significant local connectivity [14, 24].

This hypothesis suggests that techniques for evolving artificial neural network (ANN) structure should implement a means for evolution to exploit the task geometry. Without such a capability, even if useful regularities are inherent in the task, they offer no advantage or accessible bias to the learning method. Thus the ability to exploit such structure may be critical for genuinely sophisticated cognitive apparatus to be discovered.

If large-scale ANNs that exploit geometric motifs are to be evolved with computers, a powerful and efficient encoding will be necessary. This paper introduces a novel generative encoding, *Hypercube-based NeuroEvolution of Augmenting Topologies (HyperNEAT)*, for representing and evolving large-scale ANNs that exploit geometric regularities in the task domain. HyperNEAT employs the NEAT method [19] to evolve a novel abstraction of developmental encoding called *connective Compositional Pattern Producing Networks (connective CPPNs)*, which compose simple canonical functions into networks that can efficiently represent complex connectivity patterns with regularities and symmetries.

This approach is tested in a variable-resolution visual task that requires discovering a key underlying regularity across the visual field. HyperNEAT not only discovers the regularity, but is able to use it to generalize and apply its solution at higher resolutions without losing performance. The conclusion is that connective CPPNs are a powerful new generative encoding for evolving large-scale ANNs.

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. These two topics are foundational to the novel approach introduced in this paper, in which CPPNs evolve patterns that are interpreted as ANN connectivities.

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. Thus, the only way to discover such high complexity may be through a mapping between genotype and phenotype that translates few dimensions into many, i.e. through an *indirect encoding*. A most promising area of research in indirect encoding is *developmental and generative encoding*, which is motivated from biology [2, 3, 9]. In biological development, DNA maps to a mature phenotype through a process of growth that builds the phenotype over time. Development facilitates the reuse of 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 generative and developmental systems [3, 4, 7, 9, 10, 12, 13, 20, 22]. The aim is to find an abstraction of natural development for a computer running an evolutionary algorithm, so that Evolutionary Computation can begin to discover complexity on a natural scale. Prior abstractions range from low-level cell chemistry simulations to high-level grammatical rewrite systems [20].

Compositional Pattern Producing Networks (CPPNs) are a novel abstraction of development that can represent sophisticated repeating patterns in Cartesian space [16, 17]. Unlike most generative and developmental encodings, CPPNs do not require an explicit simulation of growth or local interaction, yet still exhibit their essential features. The remainder of this section reviews CPPNs, which will be augmented in this paper 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 1a shows how a two-dimensional phenotype can be generated by a function of two parameters. A mathematical abstraction of each stage of development then can be represented explicitly inside this function.

Stanley [16, 17] showed how simple canonical functions can be composed to create networks 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 the division of the body into discrete segments (e.g. with a periodic func-

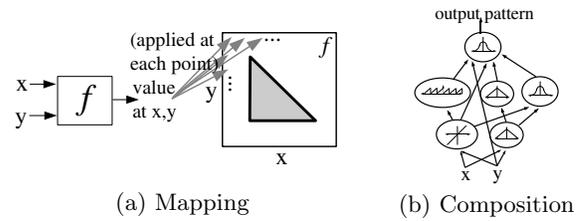


Figure 1: CPPN Encoding. (a) The function f takes arguments x and y , which are coordinates in a two-dimensional space. When all the coordinates are drawn with an intensity corresponding to the output of f , the result is a spatial pattern, which can be viewed as a phenotype whose genotype is f . (b) The CPPN is a graph that determines which functions are connected. The connections are weighted such that the output of a function is multiplied by the weight of its outgoing connection.

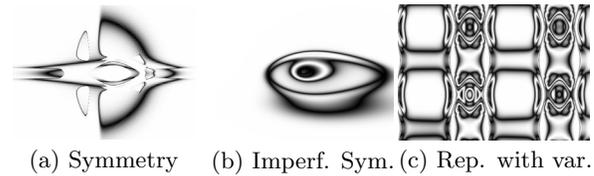


Figure 2: CPPN-generated Regularities. Spatial patterns exhibiting (a) bilateral symmetry, (b) imperfect symmetry, and (c) repetition with variation are depicted. These patterns demonstrate that CPPNs effectively encode fundamental regularities of several different types.

tion such as sine). Figure 1b shows how such a composition is represented as a network.

Such networks are called *Compositional Pattern Producing Networks* because they produce spatial patterns by composing basic functions. While CPPNs are similar to ANNs, they differ in their set of activation functions and how they are applied. Furthermore, they are an abstraction of development rather than of biological brains.

Through interactive evolution, Stanley [16, 17] showed 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 [24]), and repetition with variation (e.g. cortical columns [8]). Figure 2 shows examples of several such important motifs produced through interactive evolution of CPPNs.

These patterns are generated by applying the right activation functions (e.g. symmetric functions for symmetry; periodic functions for repetition) in the right order in the network. The order of activations is an abstraction of the unfolding process of development.

2.2 CPPN-NEAT

Because NEAT was originally designed to evolve increasingly complex ANNs, it is naturally suited to doing the same with CPPNs, which are also represented as graphs. The NEAT method begins evolution with a population of small, simple networks and *complexifies* them over generations, leading to increasingly sophisticated solutions. While the NEAT method was originally developed to solve difficult

control and sequential decision tasks, yielding strong performance in a number of domains [19, 21, 18], in this paper it is used to evolve CPPNs. This section briefly reviews the NEAT method. See also Stanley and Miikkulainen [19, 21] for detailed descriptions of original NEAT.

NEAT is based on three key ideas. 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, 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 during crossover unchanged, and allow NEAT to perform crossover without the need for expensive topological analysis. That way, genomes of different organizations and sizes stay compatible throughout evolution.

Second, NEAT speciates the population, so that individuals compete primarily within their own niches instead of with the population at large. This way, topological innovations are protected and 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, NEAT begins with a uniform population of simple networks with no hidden nodes, differing only in their initial random weights. Speciation protects new innovations, allowing diverse topologies to gradually complexify over evolution. Thus, NEAT can start minimally, and grow the necessary structure over generations. A similar process of gradually adding new genes has been confirmed in natural evolution [11, 23] and shown to improve adaptation [1]. Through complexification, high-level features can be established early in evolution and then elaborated and refined as new genes are added [11].

For these reasons, in this paper the NEAT method is used to evolve increasingly complex CPPNs. CPPN-generated patterns evolved with NEAT exhibit several essential motifs and properties of natural phenotypes [15]. If such properties could be transferred to evolved *connectivity patterns*, the representational power of CPPNs could potentially evolve large-scale ANNs, as explained in the next section.

3. HYPERNEAT

The spatial patterns in Section 2.1 present a challenge: How can such spatial patterns describe connectivity? This section explains how CPPN output can be effectively interpreted as a connectivity pattern rather than a spatial pattern. Furthermore, this novel interpretation allows neurons, sensors, and effectors to exploit meaningful geometric relationships. The next section introduces the key insight, which is to assign connectivity a geometric interpretation.

3.1 Geometric Connectivity Patterns

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.

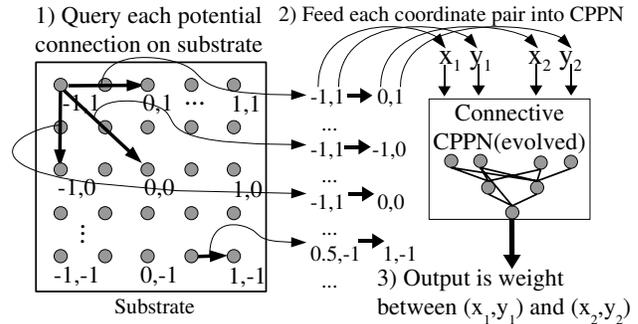


Figure 3: Hypercube-based Geometric Connectivity Pattern Interpretation. A grid of nodes, called the *substrate*, is assigned coordinates such that the center node is at the origin. (1) Every potential connection in the substrate is queried to determine its presence and weight; the dark directed lines 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. In this way, *connective CPPNs* produce regular patterns of connections in space.

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 3), where $(0, 0)$ is the center of the grid. Assuming that these nodes and their positions are given *a priori*, a *geometric connectivity pattern* 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.

A 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 magnitude 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 3).

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

Because the 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. Thus, because CPPNs generate regular spatial patterns (Section 2.1), by extension they can be expected to produce geometric connectivity patterns with corresponding regularities. The next section demonstrates this capability.

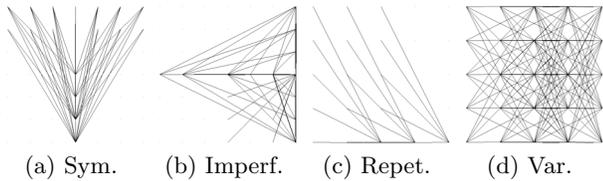


Figure 4: Connectivity Patterns Produced by Connective CPPNs. These patterns, produced through interactive evolution, exhibit several important connectivity motifs: (a) bilateral symmetry, (b) imperfect symmetry, (c) repetition, and (d) repetition with variation. That these fundamental motifs are compactly represented and easily produced suggests the power of this encoding.

3.2 Producing Regular Connectivity Patterns

Simple, easily-discovered substructures in the connective CPPN produce important connective motifs 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. Gaussian) to x_1 or x_2 (figure 4a).

The human brain is roughly symmetric at a gross resolution, but its symmetry is imperfect. Thus, imperfect symmetry is an important structural motif in ANNs. Connective CPPNs can produce imperfect symmetry by composing *both* symmetric functions of one axis along with an asymmetric coordinate frame such as the axis itself. In this way, the CPPN produces varying degrees of imperfect symmetry (figure 4b).

Another important motif in biological brains is repetition, particularly repetition with variation. Just as symmetric functions produce symmetry, periodic functions such as sine produce repetition (figure 4c). Patterns with variation are produced by composing a periodic function with a coordinate frame that does not repeat, such as the axis itself (figure 4d). 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

CPPNs produce connectivity patterns among nodes on the substrate by querying the CPPN for each pair of points in the substrate to determine the weight of the connection between them. The layout of these nodes can take forms other than the planar grid (figure 3) discussed thus far. Different such *substrate configurations* are likely suited to different kinds of problems.

For example, Churchland [6] 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 $CPPN(x_1, y_1, x_2, y_2)$, 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 be used to search for useful patterns within state-space sandwich substrates (figure 5), as is done in this paper.

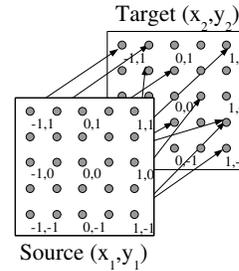


Figure 5: State-Space Sandwich Substrate. The two-dimensional grid configuration depicted in figure 3 is only one of many potential substrate configurations. This figure shows a “state-space sandwich” configuration in which a source sheet of neurons connects directly to a target sheet. Different configurations are likely suited to problems with different geometric properties. The state space sandwich is particularly suited to visual mappings.

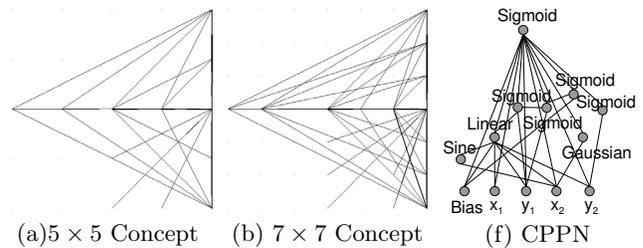


Figure 6: An Equivalent Connectivity Concept at Different Substrate Resolutions. A connectivity concept is depicted that was evolved through interactive evolution. The CPPN that generates the concept at (a) 5×5 and (b) 7×7 is shown in (c). This figure demonstrates that CPPNs represent a mathematical concept rather than a single structure. Thus, the same connective CPPN can produce patterns with the same underlying concept at different substrate resolutions (i.e. node densities).

Because connective CPPN substrates are aware of their geometry, they can use this information to their advantage. 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 [5]. That way, they can exploit *locality* by connecting to adjacent neurons with simple, repeating motifs. Connective CPPNs have the same capability.

3.4 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*, i.e. the underlying mathematical relationships that produce a particular pattern. The consequence is that *same connective CPPN* can represent an equivalent concept at different resolutions (i.e. different node densities). Figure 6 shows a connectivity concept at different resolutions.

For neural substrates, the important implication is that the same ANN 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! This operation, i.e. increasing substrate resolution, introduces a powerful new kind of complexification to ANN evolution. It is an interesting question whether, 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.

3.5 Computational Complexity

In the most general procedure, a connective CPPN is queried for every potential connection between every node in the substrate. Thus, the computational complexity of constructing the final connectivity pattern is a function of the number of nodes. For illustration, consider a $N \times N$ state-space sandwich. The number of nodes on each plane is N^2 . Because every possible pair of nodes is queried, the total number of queries is N^4 .

For example, a 11×11 substrate requires 14,641 queries. Such numbers are realistic for modern computers. For example, 250,000 such queries can be computed in 4.64 seconds on a 3.19 Ghz Pentium 4 processor. Note that this substrate is an enormous ANNs with up to a quarter-million connections. Connective CPPNs present an opportunity to evolve structures of a complexity and functional sophistication genuinely commensurate with available processing power.

3.6 Evolving Connective CPPNs

The approach in this paper 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 graph. NEAT is the natural choice for evolving CPPNs because it is designed to evolve increasingly complex network topologies. Therefore, as CPPNs complexify, so do the regularities and elaborations (i.e. the *global dimensions of variation*) that they represent in their corresponding connectivity pattern. Thus, HyperNEAT is a powerful new approach to evolving large-scale connectivity patterns and ANNs. The next section describes initial experiments that demonstrate the promise of this approach in a vision-based task.

4. EXPERIMENT

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 [5]. 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.

Therefore, in this paper a simple visual discrimination task is used to demonstrate HyperNEAT's capabilities. The task 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 in 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 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 8a). The goal is to locate the center of the largest 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 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 individual input nodes 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 them. Thus, the solution can exploit the geometric concept of *locality*, which is inherent in the arrangement of the two-dimensional grid. Only a representation that takes into account substrate geometry can exploit such a concept. Furthermore, an ideal encoding should develop a representation of the concept that is independent of the visual field 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.

4.1 Evolution and Performance Analysis

The field coordinates range between $[-1, 1]$ 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, a total of 14,641 potential connection strengths. 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 zero and three. The sign is preserved, so negative output values correspond to negative connection weights.

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 larger 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 are tested for a total number of 75 trials on the 11 by 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. Fitness is calculated from 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.

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 (P-NEAT):** P-NEAT is a reduced version of NEAT that evolves perceptrons (i.e. not CPPNs). ANNs with 121 nodes and 14,641 (121×121) links are evolved without structure-adding mutations. P-NEAT is run with the same settings as HyperNEAT, since both are being applied to the same problem. Because P-NEAT 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, which is difficult for directly-encoded methods. HyperNEAT is then tested for its ability to scale solutions to higher resolutions *without* further evolution, which is *impossible* with direct encodings such as P-NEAT.

4.2 Experimental Parameters

Because HyperNEAT extends NEAT, it uses similar parameters [19]. The population size was 100 and each run lasted for 300 generations. The disjoint and excess node coefficients were both 2.0 and the weight difference coefficient was 1.0. The compatibility threshold was 6.0 and the compatibility modifier was 0.3. The target number of species was eight and the drop off age was 15. The survival threshold within a species is 20%. Offspring had a 3% chance of adding a node, a 10% chance of adding a link, and every link of a new offspring had an 80% chance of being mutated. Recurrent connections within the CPPN were not enabled. Signed activation was used, resulting in a node output range of $[-1, 1]$. These parameters were found to be robust to moderate variation in preliminary experimentation.

5. 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 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 P-NEAT were compared to quantify the advantage provided by generative encoding on this task.

During evolution, both HyperNEAT and P-NEAT improved over the course of a run. Figure 7a 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 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 P-NEAT ($p < 0.01$).

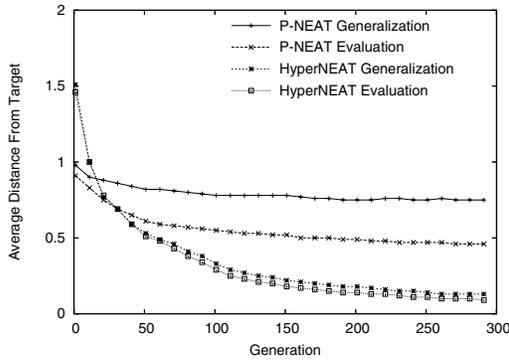
HyperNEAT learned to generalize from its training; the difference between the performance of HyperNEAT in generalization and evaluation is not significant past the first generation. Conversely, P-NEAT performed significantly worse in the generalization test after generation 51 ($p < 0.01$). This disparity in generalization reflects HyperNEAT’s fundamental ability to learn the geometric concept underlying the task, which can be generalized across the substrate. P-NEAT can only discover each proper connection weight independently. Therefore, P-NEAT has no way to extend its solution to positions in the substrate on which it was never evaluated. Furthermore, the search space of 14,641 dimensions (i.e. one for each connection) is too high-dimensional for P-NEAT to find good solutions while HyperNEAT discovers near-perfect (and often perfect) solutions on average.

5.1 Scaling Performance

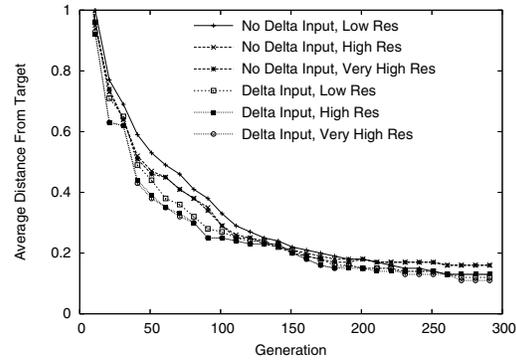
The best individuals of each generation, which were evaluated on 11×11 substrates, were later scaled with the *same* CPPN to resolutions of 33×33 and 55×55 by requerying the substrate at the higher resolutions without further evolution. These new resolutions cause the substrate size to expand dramatically. 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 learning. In contrast, P-NEAT has no means to scale to a higher resolution and cannot even learn effectively at the lowest resolution.

When scaling, a potential problem is that if the same activation level is 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 account for this disparity, the input activation levels are scaled for larger substrate resolutions proportional to the difference in unit cell size.

Two variants of HyperNEAT were tested for their ability to scale (figure 7b). The first evolved the traditional CPPN with inputs x_1 , y_1 , x_2 , and y_2 . The second evolved CPPNs with the additional *delta* inputs ($x_1 - x_2$) and ($y_2 - y_1$). The intuition behind the latter approach is that because distance is a crucial concept in this task, the extra inputs can provide a useful bias to the search.



(a) P-NEAT and HyperNEAT Generalization



(b) HyperNEAT Generation Champion Scaling

Figure 7: Generalization and Scaling. The graphs show performance curves over 300 generations averaged over 20 runs each. (a) P-NEAT is compared to 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 the HyperNEAT generalizes significantly better than P-NEAT ($p < 0.01$) and scales almost perfectly.

While the deltas did perform significantly better on average between generations 38 and 70 ($p < 0.05$), the CPPNs without delta inputs were able to catch up and reach the same level of performance after generation 70. Thus, although applicable geometric coordinate frames provide a boost to evolution, HyperNEAT is ultimately powerful enough to discover the concept of distance on its own.

Most importantly, both variants were able to scale almost perfectly from 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 because 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 structures produced by evolutionary computation of which the authors are aware.

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!

5.2 Typical Solutions

Typical HyperNEAT solutions developed several different geometric strategies to solve the domain. These strategies are represented in the connectivity between the visual field and the target field. For example, one common strategy produces a soft *halo* around each active locus, creating the largest overlap at the center of the large object, causing it to activate highest. HyperNEAT discovered several other repeating motifs that are equally effective. Thus, HyperNEAT is not confined to a single geometric strategy. Because HyperNEAT discovers how to repeat its strategy across the

substrate, solutions were able to generalize by leveraging substrate geometry.

Examples of the halo activation pattern are shown in figure 8, which also shows how the activation pattern looks at different resolution substrates generated by the same CPPN. This motif is sufficiently robust that it works at variable resolutions i.e. the encoding is scalable.

Although HyperNEAT generates ANNs with millions of connections, such ANNs can run in real time on most modern hardware. Using a 2.0GHz processor, an eight million connection networks takes on average 3.2 minutes to create, but only 0.09 seconds to process a single trial.

6. DISCUSSION AND FUTURE WORK

Like other direct encodings, P-NEAT can only improve by discovering each connection strength individually. Furthermore, P-NEAT solutions cannot generalize to locations outside its training corpus because it has no means to represent the general pattern of the solution. In contrast, HyperNEAT discovers a general connectivity concept that naturally covers locations outside its training set. Thus, HyperNEAT creates a novel means of generalization, though substrate geometry. That is, HyperNEAT *exploits the geometry of the problem* by discovering its underlying regularities.

Because the solution is represented conceptually, the same solution effectively scales to higher resolutions, which is a new capability for ANNs. Thus a working solution was possible to produce with over eight million connections, among the largest working ANNs ever produced through evolutionary computation. The real benefit of this capability however is that in the future, further evolution can be performed at the higher resolution.

For example, one potentially promising application is machine vision, where general regularities in the structure of images can be learned at low resolution. Then the resolution can be increased, still preserving the learned concepts, and further evolved with higher granularity and more detailed images. For example, a system that identifies the location of a face in a visual map might initially learn to spot a circle on a small grid and then continue to evolve at increasingly high resolutions with increasing face-related detail.

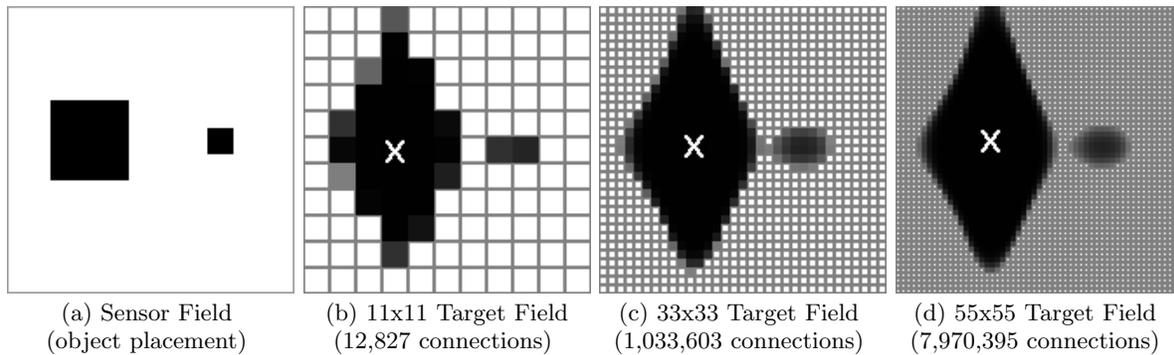


Figure 8: Activation Patterns of the Same CPPN at Different Resolutions. The object placements are shown in (a). For the same CPPN, the activation levels of the substrate are shown at (a) 11×11 , (b) 33×33 , and (c) 55×55 resolutions. Each square represents a neuron in the target field. Darker color signifies higher activation and a white X denotes the point of highest activation, which is correctly at the center of the large object in (a).

It is important to note that HyperNEAT is not restricted to state-space sandwich substrates. The method is sufficiently general to generate arbitrary two-dimensional or three-dimensional connectivity patterns, including hidden nodes and recurrence. Thus it applies to a broad range of high-resolution problems with potential geometric regularities. In addition to vision, these include controlling artificial life forms, playing board games, and developing new kinds of self-organizing maps.

7. CONCLUSIONS

This paper presented a novel approach to encoding and evolving large-scale ANNs that exploits task geometry to discover regularities in the solution. Connective Compositional Pattern Producing Networks (connective CPPNs) encode large-scale connectivity patterns by interpreting their outputs as the weights of connections in a network. To demonstrate this encoding, Hypercube-based Neuroevolution of Augmenting Topologies (HyperNEAT) evolved CPPNs that solve a simple visual discrimination task at varying resolutions. The solution required HyperNEAT to discover a repeating motif in neural connectivity across the visual field. It was able to generalize significantly better than a direct encoding and could scale to a network of over eight million connections. The main conclusion is that connective CPPNs are a powerful new generative encoding that can compactly represent large-scale ANNs with very few genes.

8. REFERENCES

- [1] L. Altenberg. Evolving better representations through selective genome growth. In *Proceedings of the IEEE World Congress on Computational Intelligence*, pages 182–187, Piscataway, NJ, 1994. IEEE Press.
- [2] P. J. Angeline. Morphogenic evolutionary computations: Introduction, issues and examples. In J. R. McDonnell, R. G. Reynolds, and D. B. Fogel, editors, *Evolutionary Programming IV: The Fourth Annual Conference on Evolutionary Programming*, pages 387–401. MIT Press, 1995.
- [3] P. J. Bentley and S. Kumar. 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)*, pages 35–43, San Francisco, 1999. Kaufmann.
- [4] J. C. Bongard. Evolving modular genetic regulatory networks. In *Proceedings of the 2002 Congress on Evolutionary Computation*, 2002.
- [5] D. B. Chklovskii and A. A. Koulakov. MAPS IN THE BRAIN: What can we learn from them? *Annual Review of Neuroscience*, 27:369–392, 2004.
- [6] P. M. Churchland. Some reductive strategies in cognitive neurobiology. *Mind*, 95:279–309, 1986.
- [7] D. Federici. Evolving a neurocontroller through a process of embryogeny. In S. Schaal, A. J. Ijspeert, A. Billard, S. Vijayakumar, J. Hallam, and Jean-Arcady, editors, *Proceedings of the Eighth International Conference on Simulation and Adaptive Behavior (SAB-2004)*, pages 373–384, Cambridge, MA, 2004. MIT Press.
- [8] G. J. Goodhill and M. A. Carreira-Perpinn. Cortical columns. In L. Nadel, editor, *Encyclopedia of Cognitive Science*, volume 1, pages 845–851. MacMillan Publishers Ltd., London, 2002.
- [9] G. S. Hornby and J. B. Pollack. Creating high-level components with a generative representation for body-brain evolution. *Artificial Life*, 8(3), 2002.
- [10] A. Lindenmayer. Adding continuous components to L-systems. In G. Rozenberg and A. Salomaa, editors, *L Systems, Lecture Notes in Computer Science 15*, pages 53–68. Springer-Verlag, Heidelberg, Germany, 1974.
- [11] A. P. Martin. Increasing genomic complexity by gene duplication and the origin of vertebrates. *The American Naturalist*, 154(2):111–128, 1999.
- [12] J. F. Miller. Evolving a self-repairing, self-regulating, French flag organism. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004)*, Berlin, 2004. Springer Verlag.
- [13] E. Mjolsness, D. H. Sharp, and J. Reinitz. A connectionist model of development. *Journal of Theoretical Biology*, 152:429–453, 1991.
- [14] O. Sporns. Network analysis, complexity, and brain function. *Complexity*, 8(1):56–60, 2002.
- [15] K. O. Stanley. Comparing artificial phenotypes with natural biological patterns. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) Workshop Program*, New York, NY, 2006. ACM Press.
- [16] K. O. Stanley. Exploiting regularity without development. In *Proceedings of the AAAI Fall Symposium on Developmental Systems*, Menlo Park, CA, 2006. AAAI Press.
- [17] K. O. Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines Special Issue on Developmental Systems*, 2007. To appear.
- [18] K. O. Stanley, B. D. Bryant, and R. Miikkulainen. Real-time neuroevolution in the NERO video game. *IEEE Transactions on Evolutionary Computation Special Issue on Evolutionary Computation and Games*, 9(6):653–668, 2005.
- [19] K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10:99–127, 2002.
- [20] K. O. Stanley and R. Miikkulainen. A taxonomy for artificial embryogeny. *Artificial Life*, 9(2):93–130, 2003.
- [21] K. O. Stanley and R. Miikkulainen. Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, 21:63–100, 2004.
- [22] A. Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society B*, 237:37–72, 1952.
- [23] J. D. Watson, N. H. Hopkins, J. W. Roberts, J. A. Steitz, and A. M. Weiner. *Molecular Biology of the Gene Fourth Edition*. The Benjamin Cummings Publishing Company, Inc., Menlo Park, CA, 1987.
- [24] M. J. Zigmond, F. E. Bloom, S. C. Landis, J. L. Roberts, and L. R. Squire, editors. *Fundamental Neuroscience*. Academic Press, London, 1999.