LETTER Communicated by Josh Bongard

Autonomous Evolution of Topographic Regularities in Artificial Neural Networks

Jason Gauci
jgauci@eecs.ucf.edu
Kenneth O. Stanley
kstanley@eecs.ucf.edu
Evolutionary Complexity Research Group, School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816, U.S.A.

Looking to nature as inspiration, for at least the past 25 years, researchers in the field of *neuroevolution* (NE) have developed evolutionary algorithms designed specifically to evolve artificial neural networks (ANNs). Yet the ANNs evolved through NE algorithms lack the distinctive characteristics of biological brains, perhaps explaining why NE is not yet a mainstream subject of neural computation. Motivated by this gap, this letter shows that when geometry is introduced to evolved ANNs through the hypercube-based neuroevolution of augmenting topologies algorithm, they begin to acquire characteristics that indeed are reminiscent of biological brains. That is, if the neurons in evolved ANNs are situated at locations in space (i.e., if they are given coordinates), then, as experiments in evolving checkers-playing ANNs in this letter show, topographic maps with symmetries and regularities can evolve spontaneously. The ability to evolve such maps is shown in this letter to provide an important advantage in generalization. In fact, the evolved maps are sufficiently informative that their analysis yields the novel insight that the geometry of the connectivity patterns of more general players is significantly smoother and more contiguous than less general ones. Thus, the results reveal a correlation between generality and smoothness in connectivity patterns. They also hint at the intriguing possibility that as NE matures as a field, its algorithms can evolve ANNs of increasing relevance to those who study neural computation in general.

1 Introduction

For at least the past 25 years (Fogel, Fogel, & Porto, 1990; Harp, Samad, & Guha, 1989; Kampfner & Conrad, 1983), a unique and growing research community has focused on the possibility that effective artificial neural networks (ANNs) can be evolved through evolutionary algorithms (Angeline, Saunders, & Pollack, 1993; Floreano, Dürr, & Mattiussi, 2008; Fogel, Fogel, & Porto, 1990; Kitano, 1990; Stanley & Miikkulainen, 2002b;

Stanley, D'Ambrosio, & Gauci, 2009; Whitley, 1995; Yao, 1999). This field of research, called neuroevolution (NE), draws inspiration from nature, wherein brains of astronomical complexity are indeed the product of an evolutionary process. Although NE is recognized within the evolutionary computation community, it remains little known within the broader neural network and neural computation communities. Perhaps one reason for this lack of engagement is that although considerable research in neural computation has focused on modeling aspects of real brains (Andrade, Muro, & Morán, 2001; Bednar, Kelkar, & Miikkulainen, 2002; Miikkulainen & Dyer, 1989; Miikkulainen, Bednar, Choe, & Sirosh, 2005; Swindale, 1996), NE algorithms generally do not produce ANNs reminiscent of biological brains. While producing biologically plausible ANNs is a high threshold for success, the suggestive analogy between NE and the evolution of brains in nature invites such ambitious expectations. In fact, it is fair to ask why an artificial evolutionary process should not produce structures reminiscent of those seen in nature. While this question is challenging, it raises the possibility that if we can identify the missing ingredients, it might be possible to incorporate them into future algorithms, thereby dramatically opening up the possibilities encompassed by NE.

In this spirit, this letter takes a step toward artificially evolving more biologically plausible ANNs. Like other work on NE (Gomez, Schmidhuber, & Miikkulainen, 2008; Stanley & Miikkulainen, 2002c; Whiteson, Kohl, Miikkulainen, & Stone, 2003), it includes a performance comparison that demonstrates the advantage of its approach—in this case, in the domain of checkers. However, more importantly the letter concludes with a new kind of analysis that has so far been impossible with evolved ANNs. In the analysis, connectivity patterns of evolved ANNs are shown to exhibit topographic maps. Furthermore, observable geometric properties of these maps are shown to correlate to greater generalization. Thus, for the first time, macrolevel qualitative properties of evolved ANN connectivity patterns are analyzed similarly to networks engineered as cortical simulations (Bednar & Miikkulainen, 2006; Bednar et al., 2002; Choe & Miikkulainen, 2004; Swindale, 1996) or even real cortical layers (Hubel & Wiesel, 1962, 1968; Weliky, Bosking, & Fitzpatrick, 1996). This analysis becomes possible because of the unique properties of the NE algorithm studied in this letter.

Before previewing the details of this algorithm, let us return to the question of the missing ingredient: Why have past NE algorithms not produced ANNs that inspired the attention of neuroscientists or neural computation researchers? For a significant class of NE algorithms called direct encodings (Gomez & Miikkulainen, 1997; Pujol & Poli, 1998; Stanley & Miikkulainen, 2002a; Whitley, 1995), a key reason for the lack of engagement is that they lack geometry. That is, while such evolved ANNs contain an aggregate of nodes and connections, the nodes do not typically exist at defined locations in space, which means that their geometry is in effect arbitrary. There is thus no concept of locality or long versus short connections; there is no concept

of symmetry or topographic regularity. Yet geometry is a critical facet of biological brains that is responsible for a preponderance of insight into their workings. Everything from MRI studies of salient brain regions for important cognitive tasks (Rosen & Lenkinski, 2007) to topographic maps (Bednar & Miikkulainen, 2006; Sporns, 2002), such as somatotopic, visual, and auditory, depends on our ability to observe neural geometry. Without such geometry, no such structures can be discerned or understood.

In fact, the problem is deeper than geometry being unobservable. The problem is that in direct encodings, there is generally no means to represent ANN structure as a function of geometry and therefore no opportunity to exploit it at all. That is, a representation that simply evolves the weight of a connection between neurons A and B has no way to associate the weight of the connection with the relative locations of A and B when they *have no location*. Yet in any topographic map, the connectivity heavily depends on the geometric relationships among nodes in the map and nodes projecting afferent connections into the map. Thus, this lack of geometric structure has inadvertently handicapped NE from evolving brain-like structures. An NE method that endows evolved ANNs with an awareness of geometry might allow the ANNs to represent more biologically plausible structures.

In response to the limitations of direct encodings, other researchers have evolved indirectly encoded ANNs (Beer, 2000; Bentley & Kumar, 1999; Bongard, 2002; Dellaert & Beer, 1994; Gruau, Whitley, & Pyeatt, 1996; Hornby, 2004; Hornby & Pollack, 2002; Husbands, Smith, Jakobi, & Shea, 1998; Komosinski & Rotaru-Varga, 2001; McHale & Husbands, 2004; Stanley & Miikkulainen, 2003), which means that they evolve a compressed description of the ANN rather than the ANN itself. Such encodings can potentially describe geometric relationships within the ANN.

Building on this idea, this letter explores the implications of evolving networks that are aware of their geometry through the *hypercube-based neuroevolution of augmenting topologies* (HyperNEAT) method, which is uniquely sensitive to geometry. In HyperNEAT, the indirect encoding, called a *compositional pattern producing network* (CPPN), has the special property that it describes network connectivity as a function of neural geometry. The CPPN encoding is unique even among indirect encodings in that it explicitly assigns positional coordinates within the geometry to each node in the ANN, and this (x, y) coordinate of each node in the ANN is literally input into the CPPN. That way, encoded ANNs exhibit observable geometric regularities. The CPPN is able to encode an entire ANN by specifying, for any two neurons situated at locations A and B respectively, the connection weight w_{AB} between them. Thus, neurons in HyperNEAT exist at defined coordinates in Euclidean space, enabling the geometric encoding of connectivity. When geometry is important to the problem, aligning the locations of neurons with the inherent geometry of the problem domain provides HyperNEAT a significant advantage over direct encodings, which have to learn the problem geometry one connection at a time.

The game of checkers is chosen as the domain for experimentation because checkers is intuitively geometric. The rules of checkers are relatively succinct, yet the strategy is potentially complex. A nice property is that the rules of movement for a piece do not change depending on the location of that piece on the board. Therefore, a representation that can extrapolate its strategies across the board to different locations gains a significant advantage over an approach that must learn the same concept for each square on the board individually.

To establish the role of geometry in effective learning, this letter examines the relative performance of HyperNEAT and another NE algorithm that learns from a predetermined network geometry intended to suit checkers. Both methods are compared to variants of a traditional approach that is completely unaware of geometry. ANNs are trained to defeat a fixed heuristic opponent. Thus, the goal in this work is not to produce the best possible checkers player but to analyze the properties of winning solutions by examining how they exploit geometry and the implications of such exploitation.

In particular, a major focus is the relationship between the geometry of solution ANNs and general performance against variants of the original heuristic that the winner was not trained against. To understand this relationship between geometry and generality, evolved receptive fields and influence maps (Hubel & Wiesel, 1962, 1968) are studied within the connectivities of general and less general players evolved by HyperNEAT. In addition, several checkers position evaluations are explored, demonstrating how subtle changes in the checkers board position result in changes in activation within different layers of the encoded ANN. This analysis reveals a surprising and novel insight: the networks that generalize most effectively display clear, smooth geometries in their connectivity patterns. In effect, they are smooth topographic maps optimized to evaluate checker boards. In contrast, less general networks (which still defeat the heuristic) exhibit jagged and irregular connectivity. The correlation between smoothness and generality can be exposed only through an NE algorithm that evolves networks with encoded geometry.

This correlation raises an intriguing insight about general behavior and its relation to geometry: an NE algorithm that does not encode structure as a function of geometry cannot be expected to exploit the same kinds of topographic correlations that are exploited by an indirect encoding and therefore cannot generalize in the same way.

Thus, a major result is that it is now possible to artificially evolve ANNs with topographic maps, moving them closer to biological plausibility, and potentially allowing artificially evolved specimens to begin to yield insights into neural function that previously were only the province of real biological study.

The letter begins with an overview of neuroevolution and the HyperNEAT method. Section 3 outlines the specific approach designed to learn

regularities in checkers through HyperNEAT. The experiment is set up in section 4. Section 5 presents performance results, as well as an analysis of general and less general networks. This analysis is discussed in section 6, which also contains an outline of future work suggested by the main insights in this letter.

2 Background

This section first reviews prior work in NE. Then the NEAT and HyperNEAT methods, which enable ANNs with geometry to evolve, are explained.

2.1 Neuroevolution. Neuroevolution is a field within evolutionary computation that focuses on training neural networks through evolutionary algorithms (Floreano et al., 2008; Yao, 1999). This approach applies the concepts of fitness, generations, populations, mutation, and sometimes crossover from evolutionary algorithms to evolve ANNs. It also benefits from the neural model, which is based on biology. In NE, the *genotype* represents an individual in the evolutionary algorithm that is transformed into an ANN during evaluation. After evaluation, the genotype receives a *fitness* that decides the parents of the next generation of individuals. NE can evolve any kind of ANN, including recurrent and adaptive networks (Floreano & Urzelai, 2000; Risi, Vanderbleek, Hughes, & Stanley, 2009; Soltoggio, Bullinaria, Mattiussi, Dürr, & Floreano, 2008). The way that the ANN is described by a genotype is called its *encoding*. This letter focuses on a specialized encoding that leverages geometry to create regular ANNs.

In early NE research, humans dictated the topology and encoding of evolved ANNs (Beer & Gallagher, 1992; Gomez & Miikkulainen, 1997; Kampfner & Conrad, 1983; Montana & Davis, 1989; Srinivas & Paranaik, 1991; Whitley, 1995; Wieland, 1991). While this approach allows human experts to design the topology and encoding with domain-specific optimizations, it is also limited by fixed topology. In contrast, evolving structure in addition to connection weights removes the burden of deciding the network topology from humans and places it on the learning algorithm (Angeline et al., 1993; Bongard, 2002; Fogel, 1992, 1993; Gruau, 1995; Hornby & Pollack, 2002; Lipson & Pollack, 2000; Miller, Todd, & Hegde, 1989; Pujol & Poli, 1998; Sims, 1994; Stanley & Miikkulainen, 2002b; Zhang & Muhlenbein, 1993).

The first methods to evolve both network structure and connection weights encoded networks *directly*, which means that a single gene in the genotype maps to a single connection in the network (Angeline et al., 1993; Pujol & Poli, 1998; Stanley & Miikkulainen, 2002b; Yao, 1999; Zhang & Muhlenbein, 1993). Although this approach is straightforward, it requires learning each connection weight individually. As a result, it is impossible to learn a regular pattern of connectivity without learning each connection in the pattern on its own. Again, human engineering is one approach to

overcoming this limitation. For example, Togelius and Lucas (2005) introduced a symmetric ANN to power a symmetric robot, which reduced the amount of evaluations required by a factor of eight. Human engineering can capture patterns and regularities in the input and reduce them to a vector of numbers. However, ideally, evolution should be able to capture patterns and regularities on its own.

Indirect encodings give evolution the opportunity to explore patterns and regularities by encoding the genotype as a *description* that maps indirectly to the target structure (Bongard, 2002; Dellaert & Beer, 1994; Gruau et al., 1996; Hornby & Pollack, 2001; Kitano, 1990; Komosinski & Rotaru-Varga, 2001; Stanley, 2007; Stanley & Miikkulainen, 2003). That way, in neuroevolution, the genotype can be much smaller than the ANN, which results in fewer variables to optimize for the evolutionary algorithm. In fact, the CPPNs in HyperNEAT, described in section 2.3, are a geometric indirect encoding that draws inspiration from biology, capable of finding and exploiting geometric regularities in the problem domain.

However, it is possible to try to build some geometric grouping into the connectivity structure of an ANN even with a direct encoding. For example, an interesting attempt to integrate geometry into NE is Blondie24, an evolved checkers-playing ANN (Chellapilla & Fogel, 2001) that was able to reach expert-level play on a popular Internet checkers server and against an expert-level version of the program Chinook (Fogel, 2002). A similar approach is taken in Blondie25, a chess program that evolved neural networks to assist in evaluating the chess board (Fogel, Hays, Hahn, & Quon, 2004). The main idea in Blondie24 is that the ANN topology can be better *engineered* to respect the regularities inherent in the game. In particular, the weights of an ANN topology engineered by hand are evolved. Every subsquare (i.e., set of positions arranged in a square shape) of the board is input to a separate hidden node responsible for only that subsquare (see Figure 1). Connections are specified from the actual board inputs to their respective subsquares and also between the inputs and the final output node. The main idea in this engineered structure is that independent local relationships within each subsquare can be learned separately and then combined at a higher level in the network. Through coevolution (i.e., candidates were evaluated by playing against each other), Blondie24 was able to reach expert-level play on a popular Internet checkers server (Chellapilla & Fogel, 2001). However, an intriguing alternative would remove the need for engineering by learning geometric regularities on its own. This is the idea behind the HyperNEAT approach taken in this letter. The next section describes NEAT, which is extended later to implement HyperNEAT.

2.2 NeuroEvolution of Augmenting Topologies. The approaches compared in this letter are variants of the neuroevolution of augmenting topologies (NEAT) method (Stanley & Miikkulainen, 2002b, 2004), which, like the approach in Blondie24 (Chellapilla & Fogel, 2001), evolves ANNs.

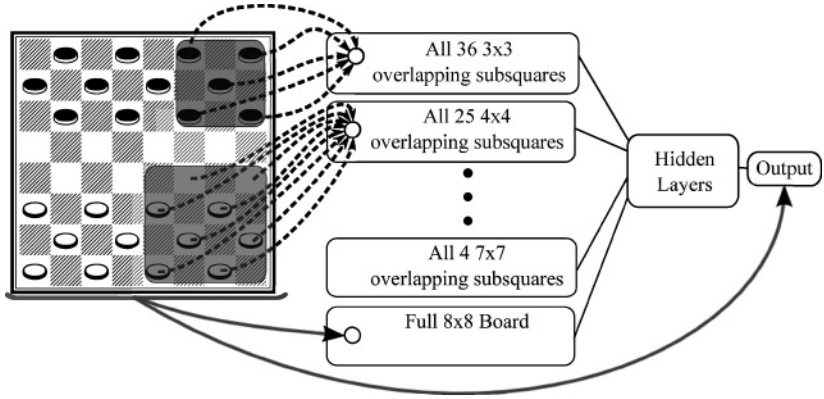


Figure 1: Blondie24 ANN Topology (Chellapilla & Fogel, 2001). The first hidden layer contains a node for every subsquare of the board of size greater than 2×2. Positions on the board are linked to the corresponding subsquares that contain these positions. This layer then connects to hidden layers that finally connect to the output node. Each valid square on the board connects directly to the output node.

In addition to evolving weights of connections, NEAT can build structure and add complexity. While the encoding in NEAT is direct, it turns out that it can be made indirect, which is the idea behind HyperNEAT. NEAT itself is a leading neuroevolution approach that has shown promise in board games and other challenging control and decision-making tasks (Aaltonen et al., 2009; Stanley & Miikkulainen, 2004; Stanley, Bryant, & Miikkulainen, 2005a; Stanley, Kohl, & Miikkulainen, 2005; Taylor, Whiteson, & Stone, 2006). This section reviews the NEAT method (for a full description see Stanley & Miikkulainen, 2002b, 2004; Stanley, Bryant, & Miikkulainen, 2005b).

NEAT is based on three key ideas. First, 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. The historical marking is a number assigned to each gene based on 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 topological analysis. That way, networks of different organizations and sizes stay compatible throughout evolution.

Second, NEAT divides the population into species, so that individuals compete primarily within their own niches instead of with the population at large. In 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.

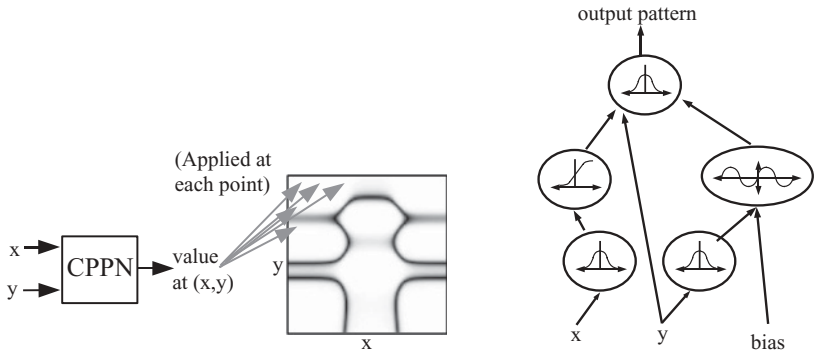
Third, NEAT begins with a uniform population of simple networks with no hidden nodes, differing only in their initial weights. Speciation protects new innovations, allowing diverse topologies to increase gradually in complexity over evolution. Thus, NEAT can start minimally and grow the necessary structure over generations. Through increasing complexity, high-level features can be established early in evolution and then elaborated and refined as new genes are added (Altenberg, 1994). Each component of NEAT was originally validated through a series of ablation studies (Stanley & Miikkulainen, 2002b). The most important concept for the purposes of this letter is that NEAT can evolve the right network structure and connection weights for the task. The next section reviews the extension of NEAT that allows it to evolve geometric relationships automatically.

2.3 CPPNs and HyperNEAT. Like many other methods in machine learning, the reason that regular NEAT cannot explicitly learn geometric regularities is that when it learns to represent important local relationships (e.g., how a checkers piece in one square can be threatened by another in an adjacent square), it cannot extend that relationship as a pattern of connectivity across the entire neural structure connecting to the board. In other words, it needs to rediscover similar concepts multiple times.

The main idea in HyperNEAT is that it is possible to learn such relationships if the solution is represented *indirectly*, which means that it is a *generative description* of the connectivity of the ANN rather than embodying the connection weights of the ANN itself. As explained briefly in section 2.1, unlike a *direct* representation, in which every dimension in the solution space (i.e., the phenotype in evolutionary computation) is described individually (i.e., by its own gene), an indirect representation can describe a pattern of values in the solution space without explicitly enumerating every such value. That is, information is reused in such an *indirect encoding*, which is a major focus in the field of *generative and developmental systems*, the subfield of evolutionary computation from which HyperNEAT originates (Astor & Adami, 2000; Bentley & Kumar, 1999; Eggenberger, 1997; Hornby & Pollack, 2002; Lindenmayer, 1974; Mattiussi & Floreano, 2007; Stanley & Miikkulainen, 2003; Turing, 1952).[1]

HyperNEAT is based on an indirect encoding called *compositional pattern producing networks* (CPPNs; Stanley, 2007). The idea behind CPPNs is that patterns such as those seen in nature can be described at a high level as a *composition of functions* that are chosen to represent several common motifs in patterns. For example, because the gaussian function is symmetric,

[1]The ideas in GDS extend back to Turing (1952), who experimented with pattern formation through reaction-diffusion systems.



(a) Mapping (b) Composition

Figure 2: CPPN encoding. (a) The CPPN 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 the CPPN, the result is a spatial pattern, which can be viewed as a phenotype whose genotype is the CPPN. (b) Internally, the CPPN is a graph that determines which functions are connected. As in an ANN, the connections are weighted such that the output of a function is multiplied by the weight of its outgoing connection. The CPPN in b actually produces the pattern in a.

when it is composed with any other function, the result is a symmetric pattern. The appeal of this encoding is that it allows patterns with regularities such as symmetry (e.g., with gaussians), repetition (e.g., with periodic functions such as sine), and repetition with variation (e.g., by summing periodic and aperiodic functions) to be represented as networks of simple functions, which means that NEAT can evolve CPPNs just as it evolves ANNs. While CPPNs are similar to ANNs, the distinction in terminology is particularly important for explicative purposes because in HyperNEAT, CPPNs *describe* ANNs. Formally, CPPNs produce a phenotype that is a function of n dimensions, where n is the number of dimensions in a geometric space. For each coordinate in that space, its level of expression is an output of the function that encodes the phenotype. Figure 2 shows how a two-dimensional pattern can be generated by a CPPN that takes two inputs.

The main idea in HyperNEAT is to extend CPPNs, which encode two-dimensional spatial patterns, to also represent *connectivity patterns* (D'Ambrosio & Stanley, 2007; Gauci & Stanley, 2007; Stanley et al., 2009). That way, NEAT can evolve CPPNs that represent ANNs with symmetries and regularities that are computed directly from the *geometry* of the task inputs. The key insight is that $2n$-dimensional spatial patterns are *isomorphic* to connectivity patterns in n dimensions (i.e., in which the coordinate of each end point is specified by n parameters). Therefore, the connectivity

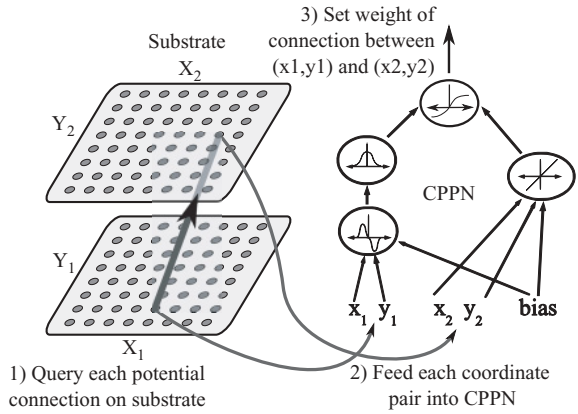
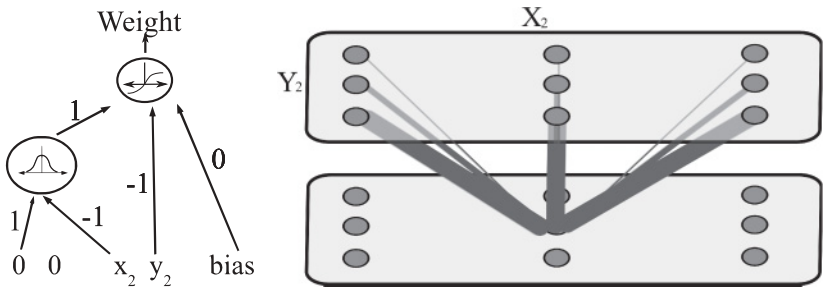


Figure 3: Hypercube-based geometric connectivity pattern interpretation. A grid of nodes, called the substrate, is assigned coordinates. (1) Every potential connection in the substrate is queried to determine its presence and weight; the directed line shown in the substrate represents a sample connection that is queried. (2) For each query, the CPPN takes as input the positions of the two end points and (3) outputs the weight of the connection between them. In this way, *connective CPPNs* produce regular patterns of connections in space.

patterns encoded by CPPNs can exhibit the same kinds of symmetries and regularities as those seen in CPPN-generated spatial patterns.

Consider a CPPN that takes four inputs labeled x_1, y_1, x_2, and y_2; this point in four-dimensional space can *also* denote the connection between the two-dimensional points (x_1, y_1) and (x_2, y_2), and the output of the CPPN for that input thereby represents the weight of that connection (see Figure 3). By querying every possible connection among a set of points in this manner, a CPPN can produce an ANN, wherein each queried point is the position of a neuron. Because the connection weights are produced as a function of their end points, the final structure is produced with *knowledge* of its geometry. In effect, the CPPN paints a pattern on the inside of a four-dimensional hypercube that is interpreted as an isomorphic connectivity pattern, which explains the origin of the name *hypercube-based NEAT* (HyperNEAT). The example in Figure 4 illustrates the natural connection between the function embodied by the CPPN and the geometry of the resultant network. Connectivity patterns produced by a CPPN in this way are called *substrates* so that they can be verbally distinguished from the CPPN itself, which has its own internal topology.

Recall that each queried point in the substrate is a node in an ANN. The experimenter defines both the location and role—hidden, input, or output—of each such node. As a rule of thumb, nodes are placed on the substrate to reflect the geometry of the task, which makes the setup straightforward (Clune, Ofria, & Pennock, 2008; D'Ambrosio & Stanley, 2007; Gauci



(a) CPPN (b) ANN (Substrate)

Figure 4: Example CPPN describing connections from a single node. An example CPPN (a) with five inputs (x_1, y_1, x_2, y_2, $bias$) and one output ($weight$) contains a single gaussian hidden node and five connections. The function produced is symmetric along x_1 and x_2 (because of the gaussian) and linear with respect to y_2 (which directly connects to the CPPN output). For the given fixed input coordinate ($x_1 = 0$, $y_1 = 0$), the CPPN in effect produces the function $Gaussian(-x_2) - y_2$. This pattern of weights from input node $(0, 0)$ is shown on the substrate (b). Weight magnitudes are indicated by thickness. Note that the pattern produces a set of weights that are symmetric along the x-axis and linearly decreasing as the value of y_2 increases. In this way, the function embodied by the CPPN encodes a geometric pattern of weights in space. HyperNEAT evolves the topologies and weights of such CPPNs.

& Stanley, 2007; Stanley et al., 2009). That way, the connectivity of the substrate becomes a direct function of the task structure.

For example, in a board game, the inputs can be placed on the substrate in a twodimensional plane just as their corresponding squares are arranged on the board, as in Figure 3. In this way, knowledge about the problem can be injected into the search, and HyperNEAT can exploit the regularities (e.g., adjacency, or symmetry) of a problem that are invisible to traditional encodings (see algorithm 1). (For full descriptions of HyperNEAT see D'Ambrosio & Stanley, 2007; Gauci & Stanley, 2007; and Stanley et al., 2009.) The next section explains in detail how checkers is represented and learned by HyperNEAT.

3 Approach: Learning Regularities in Checkers

The game of checkers is chosen for the experiments in this letter because it is intuitively geometric. While approaches like Blondie24 engineer geometry into the ANN topology in the hope that such engineering may be useful, the idea in HyperNEAT is to *learn* from geometry by generating the policy network as a direct function of task geometry. This section explains how that is done in the game of checkers.

Input: Substrate Configuration
Output: Solution CPPN
Initalize population of minimal CPPNs with random weights;
while *Stopping criteria is not met* **do**
  foreach *CPPN in the population* **do**
    foreach *Possible connection in the substrate* **do**
      Query the CPPN for weight w of connection;
      if *Abs(w)* $>$*Threshold* **then**
        Create connection with a weight scaled proportionally to w (figure 3);
      end
    end
    Run the substrate as an ANN in the task domain to ascertain fitness;
  end
  Reproduce CPPNs according to the NEAT method to produce the next generation;
end
Output the Champion CPPN;

Algorithm 1: Basic HyperNEAT algorithm.

To apply HyperNEAT to checkers, the substrate input layer is arranged in two dimensions to match the geometry of the checkers board (see Figure 5). Notice that the substrate in Figure 5 includes a hidden layer. Thus, it is analogous to two substrates (e.g., see Figure 3) stacked on top of each other. In particular, the two-dimensional input layer connects to an analogous two-dimensional hidden layer so that the hidden layer can learn to process localized geometric features. The hidden layer then connects to a single output node, whose role is to evaluate board positions. The CPPN distinguishes the set of connections between the inputs and the hidden layer from those between the hidden layer and the output node by querying the weights of each set of connections from a *separate* output on the CPPN (note the two outputs in the CPPN depiction in Figure 5). That way, the x and y positions of each node are sufficient to identify the queried connection, and the outputs differentiate one connection layer from the next. Because the CPPN can effectively compute connection weights as a function of the *difference* in positions of two nodes, it can easily map a repeating concept across the whole board.

In this way, the substrate is a board evaluation function. The function inputs a board position and outputs its value for black. To evaluate the board when it is white's turn to move, the color of the pieces can be reversed and then the sign of the result inverted. To decide which move to make, a minimax search algorithm runs to a fixed ply depth of four. Alpha-beta pruning (Knuth & Moore, 1975) and iterative deepening (Russell, Norvig, Canny, Malik, & Edwards, 1995) techniques increase performance without

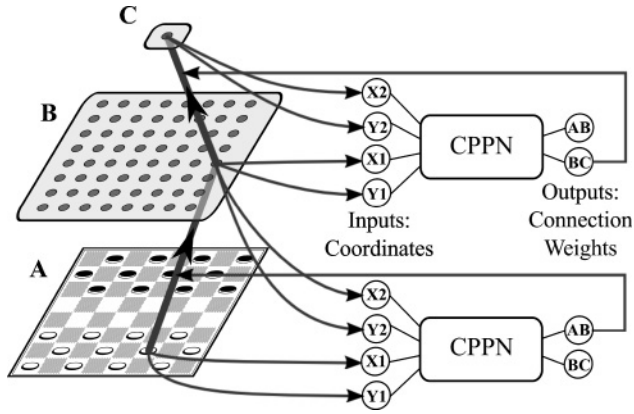


Figure 5: Checkers substrate. The substrate (at left) contains a two-dimensional input layer (A) that corresponds to the geometry of a game board, an analogous two-dimensional hidden layer (B), and a single-node output layer (C) that returns a board evaluation. The two CPPNs (at right) are depictions of the same CPPN being queried to determine the weights of two different substrate connections. The bottom CPPN depiction receives as input the x and y coordinates of a node in A and a node in B and returns the weight of this connection from its AB output node. Similarly, the top depiction of the same CPPN is being queried for the weight of a connection between B and C and therefore returns this weight from its BC output. In this way, a four-input CPPN can specify the connection weights of a two-layer network structure as a function of the positions, and hence the geometry, of each node.

changing the output. The output of the substrate is the heuristic score for the minimax algorithm.

This approach allows HyperNEAT to discover geometric regularities on the board by expressing connection weights as a function of geometry. It is therefore unnecessary to manually engineer the network topology or divide the input space into subsections in an attempt to inject a priori theories about the key regularities in the game into the representation. Because HyperNEAT discovers geometric relationships on its own, an identical substrate can be applied to other board games even without knowledge of the game rules, contributing to the generality of the approach.

4 Experiment

The experiment is designed to investigate the role of neural geometry in solving a problem that is clearly geometric. The idea is to learn to defeat a single fixed training opponent and then test for generalization against variations of this opponent. Thus, rather than producing the best possible checkers player, the aim is to analyze in detail the implications of a

geometric representation, not only for learning but especially for generalization beyond what was trained.

Board games are an effective platform to discern the importance of geometry because they depend heavily on geometric relationships that often repeat across the board. Therefore, to begin the investigation, this letter compares four evolutionary approaches that take geometry into account to varying degrees in the domain of checkers. Each approach is trained against the same hand-engineered deterministic opponent (Fierz, 2002). The opponent is a linear combination of several heuristics, including material possession, positional bias, whether pieces on the back row have been moved (which would lower the score), whether a double corner is intact, and who controls the center and the edge of a board. Thus, the deterministic opponent is nontrivial (i.e., not just a simple piece counter). During evolution, each candidate plays a single game as black against the opponent to determine its fitness. Both the evolved player and the opponent evaluate boards that are four ply ahead. Fitness is computed as a function of both the final game state and intermediate board states. At each turn t, fitness f_t is awarded based on the current board state according to:

$$f_t = 100 + 2m_s + 3k_s + 2(12 - m_o) + 3(12 - k_o), \tag{4.1}$$

where m_s and m_o are the number of regular pieces possessed by the learner and the opponent, respectively, and k_s and k_o are the number of kings. The coefficients 2 and 3 represent the values of pieces and kings, respectively, denoting that kings are roughly 1.5 times as valuable as regular pieces. Because there are at most 12 pieces of any given type, the number 12 ensures positive values for its respective terms. This function rewards incremental progress and provides a smoother learning gradient than simply awarding fitness based on the final score. The value of 100 per turn rewards individuals more who play games that last a longer number of turns. Thus, evolved players who lose quickly will receive less fitness. If the evolved player wins, fitness is awarded over 100 turns even if the game ends earlier. That way, winning early is not penalized. If the candidate wins against the training opponent, an additional 30,000 is added to the total fitness. It is important to note that this fitness function is unique and not based on Blondie24, whose results are therefore not directly comparable.

The learned strategies are then tested against a nondeterministic variant of the same opponent. This variant has a 10% chance of choosing the second-highest-scoring move instead of the optimal move found in minimax search. This approach is similar to work done by Fogel (1993), who also implemented a percentage chance of picking a random move to diversify a deterministic opponent. Methods that evolve more general solutions should produce policies that win more such games.

The four compared approaches are chosen carefully to isolate the issue of geometric processing. Therefore, they are all variants of the same

NEAT approach. This shared basis means that differences in performance are attributable to the way each approach processes its inputs. For all four approaches, input values of 0.5 and -0.5 encode black and white pieces, respectively. Kings are represented by a magnitude of 0.75, which is similar to the approach in Chellapilla and Fogel (2001), who showed that multiplying the standard piece input magnitude by 1.3 produces a good magnitude for kings in their approach. A single output expresses the value of the current board state for black.

Regular NEAT inputs a vector of length 32 in which each parameter represents a square on the board that can potentially hold a piece. NEAT evolves the topology and weights between the input and output nodes.

NEAT-EI is an attempt to enhance NEAT's ability to take into account geometric regularities across the board by supplying additional engineered inputs (EI). It has the same inputs as NEAT; however, the starting network topology is engineered as in Blondie24 to have additional inputs that focus on geometric regions of differing sizes (Chellapilla & Fogel, 2001; see Figure 1). The result of training NEAT-EI in this letter cannot be compared directly to Blondie24 because Blondie24 is the result of coevolution, while the policies in this letter are evolved against a fixed opponent. Rather than evolving the best possible player, the goal in this letter is to fairly compare the generalization of different representations, thereby isolating the issue of generalization.

HyperNEAT inputs are arranged in a two-dimensional 8×8 grid that forms the first layer of a three-layer substrate (see Figure 5). For HyperNEAT, NEAT evolves the CPPN that computes the connection weights of the substrate.

If it is indeed possible to exploit geometry to improve play, the better an approach can represent geometric relationships (through either learning or a priori engineering), the better that method should learn and generalize.

FT-NEAT (fixed topology NEAT) inputs are arranged in the same configuration as in the substrate in HyperNEAT (see Figure 5). FT-NEAT evolves the weights of this ANN but not the topology. Thus, FT-NEAT must evolve the connection weights of over 4000 directly encoded connections, helping to confirm that it is not just the particular topology of the substrate in Figure 5, but more important, the indirect encoding in HyperNEAT, that provides an advantage.

After the experimental comparison among the four methods, an extensive analysis of substrate visualizations from more and less general HyperNEAT-evolved players investigates how geometry influences generalization and the way evolved maps are organized.

4.1 Experimental Parameters. Because HyperNEAT and NEAT-EI extend NEAT, both use the same parameters as NEAT (Stanley & Miikkulainen, 2002b). FT-NEAT also uses the same parameters, except it

does not add nodes or connections. The population size was 120, and each run lasted 200 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 and a 5% chance of adding a link, and every link of a new offspring had an 80% chance of being mutated. Available CPPN activation functions were sigmoid, gaussian, sine, and linear functions. Recurrent connections within the CPPN were not enabled. Signed activation was used, resulting in a node output range of $[-1, 1]$. By convention, a connection is not expressed if the magnitude of its weight is below a minimal threshold of 0.2 (Gauci & Stanley, 2007); otherwise, it is scaled proportionally to the CPPN output. These parameters were found to be robust to variation in preliminary experimentation.

5 Results

Performance in this section is measured in two ways. First, the fitness of each approach is tracked during training over generations, which gives a sense of relative *training* performance. Second, after training is complete, the best solutions from each run play 100 games against the randomized opponent, yielding generalization. The main question is whether HyperNEAT's ability to learn from geometry benefits its performance and generalization.

5.1 Training Performance. Figure 6 shows the average generation champion fitness over evolution, averaged over 20 runs. While none of the runs of regular NEAT nor FT-NEAT was able to defeat the opponent within 200 generations, both HyperNEAT and NEAT-EI learned to defeat it in all runs. On average, it took NEAT-EI 57.85 generations to find a winning solution. HyperNEAT succeeds much more quickly, finding a winner in 8.2 generations on average. These differences are statistically significant according to Student's t-test ($p < 0.05$). This disparity highlights the critical importance of learning from geometry. While defeating the heuristic appears challenging with direct representations, it becomes easy if the solution is learned as a function of the board geometry.

5.2 Generalization. Every generation champion that defeats the deterministic opponent plays 100 games against the randomized opponent. Because regular NEAT and FT-NEAT could never defeat this opponent, they are not included in this test. To make the comparison fair, only the *most general* solutions of each run are compared, which means the generation champion with the highest score computed by $W + \frac{T}{2}$, where W and T are the number of wins and ties against the randomized opponent, respectively. The equation $W + \frac{T}{2}$ is used to convert a wins, losses, and ties

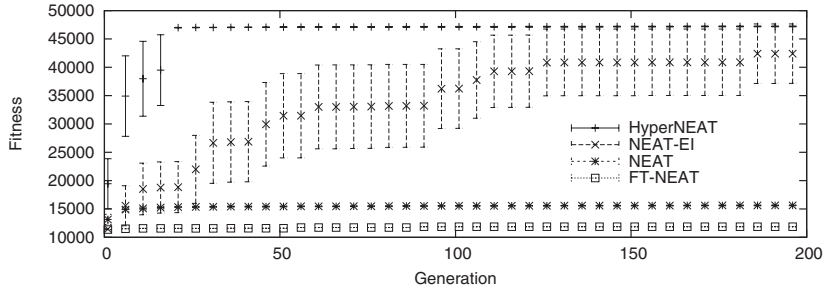


Figure 6: Fitness during training. The fitness of the generation champions of each approach is shown, averaged over 20 runs. HyperNEAT generation champions perform significantly better than NEAT-EI between generations 1 and 123 ($p < 0.05$ using Student's t-test). Error bars show the 95% confidence interval. HyperNEAT learns faster than NEAT-EI because its CPPN solutions require fewer dimensions to represent.

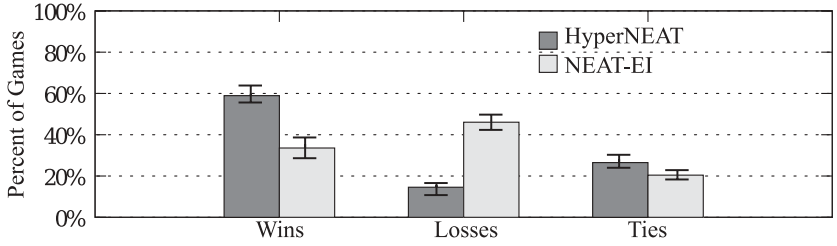
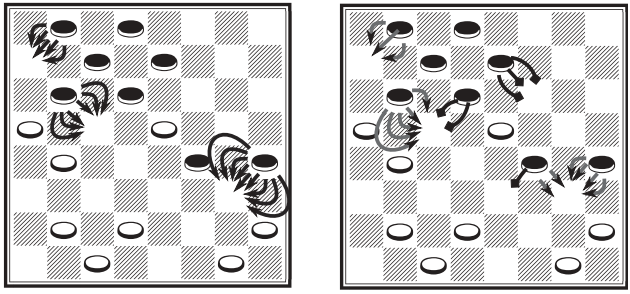


Figure 7: Generalization results. Average wins, losses, and ties in 100 games against the randomized opponent are shown for HyperNEAT and NEAT-EI, averaged over 20 runs of each. Only the most general solutions of each run are included in the test. HyperNEAT solutions win significantly more games ($p < 0.05$) and lose significantly fewer games ($p < 0.05$) than NEAT-EI. Error bars show a 95% confidence interval. The difference in ties between the two methods is not significant ($p \approx 0.06$).

metric to a single scalar score. That way, the generalization results focus on the *best possible* generalization for both methods when they learn to defeat an identical opponent. The best possible generalization represents what would result from an ideal validation of the trained opponents. While in the real world such idealized validation may not always be possible, assuming reasonable effort on the part of the experimenter, it is a yardstick for how well a system can be expected to perform in a reinforcement learning task. A similar approach to measuring generalization in such a task is taken by Gruau et al. (1996). Figure 7 shows the results of these solutions against the randomized opponent. HyperNEAT wins significantly more and loses



(a) HyperNEAT Moves (b) NEAT-EI Moves

Figure 8: Requested moves from the same board position by HyperNEAT and NEAT-EI. This figure depicts a position several moves into a game. Twenty moves requested by the champions of all NEAT-EI runs are contrasted with 20 from HyperNEAT runs. All of the HyperNEAT runs suggest neutral or positive moves. Six of the NEAT-EI runs make moves that lead to immediate, uncompensated loss. These moves are denoted with a darker line and a square end point.

significantly less than NEAT-EI. The geometric encoding allows HyperNEAT to generalize across the board.

5.3 Typical Solutions. HyperNEAT's advantage is most evident in the middle game and later. As the game tree branches, deviation from the training opponent increases. Because HyperNEAT performs better in such novel situations, it is more general. For example, Figure 8 contrasts moves chosen by NEAT-EI solutions with those from HyperNEAT from the *same* unfamiliar position. NEAT-EI players unnecessarily sacrifice pieces, while HyperNEAT players rarely do from this position. Given that the evaluations during training consist of a single game against a deterministic opponent, the ability of a solution evolved during training to perform well in generalization tests against a nondeterministic opponent is significant. These typical solutions demonstrate the idea that because HyperNEAT evolves a pattern of weights across the geometry of the substrate, HyperNEAT is able to evolve a player that can both defeat the deterministic heuristic and simultaneously perform well in generalization tests, without any need for generalization pressure in the fitness function. Conversely, NEAT-EI struggles to generalize, suggesting that NEAT-EI learned a specific subset of board states instead of a general checkers strategy. In the case of NEAT-EI, generalization would likely benefit from playing additional games in a single evaluation.

The most general solution in all runs of NEAT-EI has 126 nodes and 1,106 connections. In contrast, the most general solution of HyperNEAT is a CPPN with only 23 nodes and 84 connections, which generates an ANN with 129

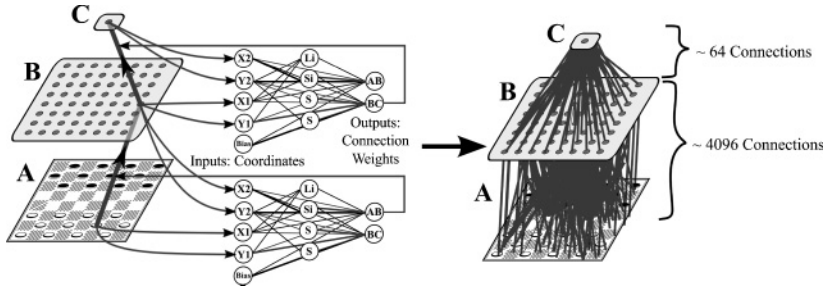


Figure 9: Compression in CPPN encoding. The CPPN at left, which is an actual solution against the heuristic, contains only 18 connections, yet it encodes the connection weights of a substrate with over 4000 connections. In this way, HyperNEAT searches a significantly lower-dimensional space than a direct encoding. In the figure above, the letters A, B, and C, represent the input, hidden, and output layer, respectively. The output labeled AB determines the connection weight for a link originating in the input layer and terminating at the hidden layer (following Figure 5).

nodes and 3,979 connections. Figure 9 illustrates this dramatic compression afforded by indirect encoding in a typical HyperNEAT solution. In this way, HyperNEAT is able to explore a significantly smaller search space (i.e., CPPNs) while still creating complex structures (i.e., substrates).

5.4 Substrate Visualizations. While the results so far establish that learning from geometry provides an advantage in both performance and generalization, an important question is exactly how this advantage is realized. This section aims to investigate this question by examining the internal connectivity and activation patterns of HyperNEAT-trained networks. It is important to note that this study of the topographic layout of nodes and connectivity within an evolved ANN is possible only because, unlike other neuroevolution algorithms (Angeline et al., 1993; Beer & Gallagher, 1992; Bongard, 2002; Braun & Weisbrod, 1993; Floreano et al., 2008; Fogel, 1993; Gomez & Miikkulainen, 1997; Gruau et al., 1996; Hornby & Pollack, 2002; Kitano, 1990; Koza & Rice, 1991; Lipson & Pollack, 2000; Montana & Davis, 1989; Moriarty & Miikkulainen, 1996; Opitz & Shavlik, 1997; Pujol & Poli, 1998; Sims, 1994; Srinivas & Paranaik, 1991; Stanley & Miikkulainen, 2002b; Yao, 1999; Zhang & Muhlenbein, 1993), the neurons within a HyperNEAT substrate are situated at geometric coordinates. This geometry is what affords the opportunity to observe patterns in their actual situated geometric context, giving insight into why such a context is important to learning in general and what kinds of opportunities it creates.

The particular focus of the analysis in this section is on the question of what kinds of connectivity patterns lead to generalization and what kinds

Table 1: Selected General and Less General HyperNEAT Solutions.

General Solutions Against Nondeterministic Heuristic				Less General Solutions Against Nondeterministic Heuristic			
Solution	Wins	Losses	Ties	Solution	Wins	Losses	Ties
1	53	22	25	1	35	24	41
2	62	17	21	2	39	8	53
3	61	17	22	3	38	33	29
4	54	28	18	4	35	17	48

Notes: the two tables show the wins, losses, and ties of selected general and less general champions against the nondeterministic heuristic that are visualized later in this section. Note that the champions selected are not necessarily the champions of the run. Because the training phase involves only a single game against a deterministic heuristic, there is no explicit reward for generality in the fitness function. Even so, some of the runs produce solutions that generalize better than others. Note that because HyperNEAT generalizes well on average, the poorest generalizers from HyperNEAT still outgeneralize average NEAT-EI champions significantly. Nevertheless, the difference between these less general champions and those that are even more general still helps to elucidate the factors underlying effective generalization.

do not. To investigate this difference, a group of four highly general HyperNEAT solutions and four HyperNEAT solutions that generalize less effectively (summarized in Table 1) are visualized in two different ways. First, the connectivity patterns of the solutions are visualized through images of influence maps and receptive fields. These images are arranged vertically within a single panel in one column (see Figure 10). The bottom image of each panel is a set of five *influence maps* that shows how individual inputs from the checkers board influence the entire hidden layer. The intensity at each position within each such map represents the magnitude of a single connection weight, and white triangles in the top-left corner of a position represent negative connection weights (the darker color denotes less influence). Thus, a full influence map shows all the weights projecting from a single input to the entire hidden layer. The five influence maps form a cross-shape, symbolizing that they represent images coming from five locations on the checkers board, as shown in Figure 10. Above the five influence maps are a similar five *receptive field* visualizations. These images are designed to show how each *hidden node* sees all of the inputs that can connect to it. Like the influence maps, the five receptive field visualizations are also shown in a cross—in this case, to represent where in the *hidden layer* the receiving node is located. The single image at the top is the receptive field of the single output node, which shows the connection weights from the hidden layer to the output, which is how the final computation of the board value is completed.

Second, visualizations of hidden-layer *activation patterns* for several board positions illustrate how boards are evaluated in the game tree (see

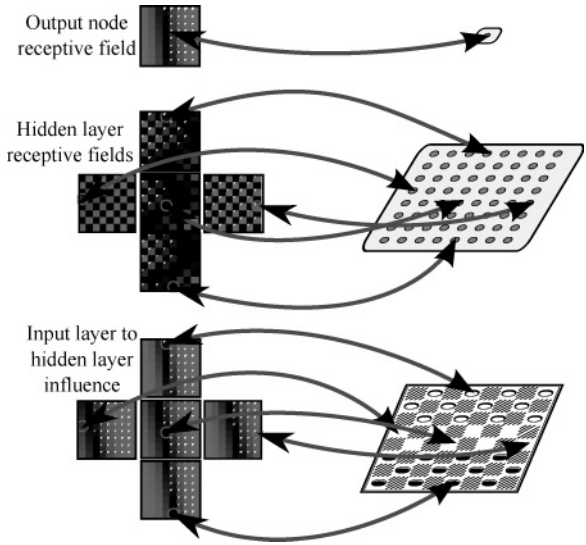


Figure 10: Visualizing connection weights. In this section (see Figures 12 and 13), connection weights within substrates are depicted as shown in this figure. For the influence maps (bottom), the lines show from which square on the checkers board each influence map originates. Similarly, receptive fields (middle and top) are shown for the hidden nodes with which each such field is connected. An influence map originates from a single input, and a receptive field terminates at a single hidden node. The cross-patterns are designed to make it easy to see how the pattern of each influence map or receptive field varies with their originating position.

Figure 11). Each figure with this type of visualization displays a row of activation patterns obtained by the champion of a particular run from Table 1. The activation pattern is illustrated by a column depicting the input board configuration (bottom), the hidden-layer node activation levels (middle), and the output activation (top), which is the value assigned to that board position for black.

These activation patterns are also organized topographically such that the activation level of each hidden neuron is depicted at that neuron's actual position in the substrate. Thus, it is possible to see how the activation levels relate to the network's geometry.

Each board position is an actual position encountered during alpha-beta search, so the overall visualization makes it possible to see how the network represents the difference between relatively good or bad situations. The board positions and activation patterns are ordered from left to right by increasing output value so that it is easy to see how increasingly good positions are represented by the network internally. Note that the sequence

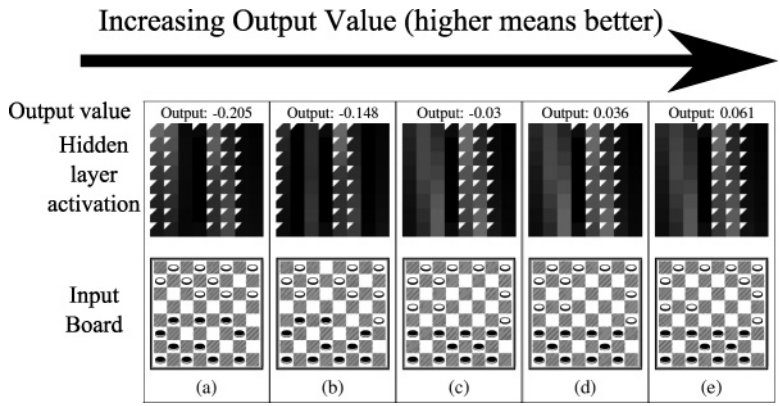


Figure 11: Visualizing hidden-layer activation patterns for different board positions. To understand how different-quality board positions influence the hidden layer of a particular general or less general substrate, board positions and hidden-layer activations in this section (Figures 14–21) are visualized as shown in this figure. Each such board position was actually encountered by an alpha-beta search with the associated substrate during game play. To elucidate how the hidden layer distinguishes worse from better positions, panels a through e are always ordered from the lowest evaluation score returned by the substrate output to the highest. That way, it is possible to understand the scenarios preferred by the learned evaluation function. See the text for full details.

from left to right depicts board positions that were encountered during alpha-beta search rather than a sequence of moves during actual gameplay. The aim is to elucidate how judgments on board quality are represented.

The contrast between connectivity patterns from general (see Figure 12) and less general (see Figure 13) players is surprisingly revealing. In fact, the distinguishing characteristic of general play is visually apparent by simply observing its geometry: the connectivity patterns of networks that generalize most effectively exhibit *smooth* boundaries, while the boundaries of those that generalize poorly are *jagged*. This difference is particularly prominent in the influence maps (at the bottom of Figures 12 and 13), suggesting that influence maps from inputs reveal an important facet of geometry. These characteristics are consistent across all general and less general solutions in Figures 12 and 13. Thus, there is a strong correlation between generality and geometric smoothness.

The role of smoothness in generalization yields the important insight that the ability to *represent* smooth regularity is a critical prerequisite to consistent generalization. In contrast, jaggedness suggests *memorization* of the specific situations encountered while playing the particular opponent heuristic (recall that both general and less general solutions defeated the

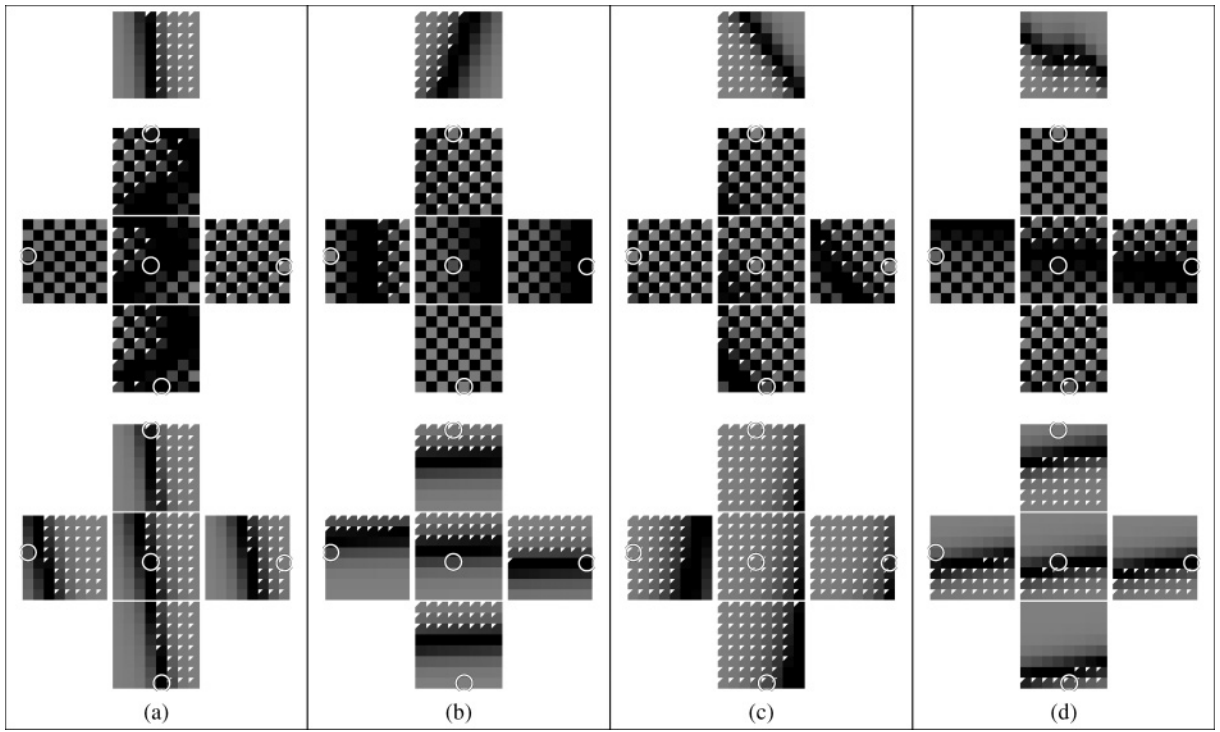


Figure 12: Connectivity patterns of general solutions. Influence maps and receptive fields (as explained in Figure 10) are shown for four general solutions. An important feature shared by all general solutions is that their connectivity patterns are smooth and continuous. They also vary in a regular fashion with their originating node's location (bottom) or hidden node location (middle).

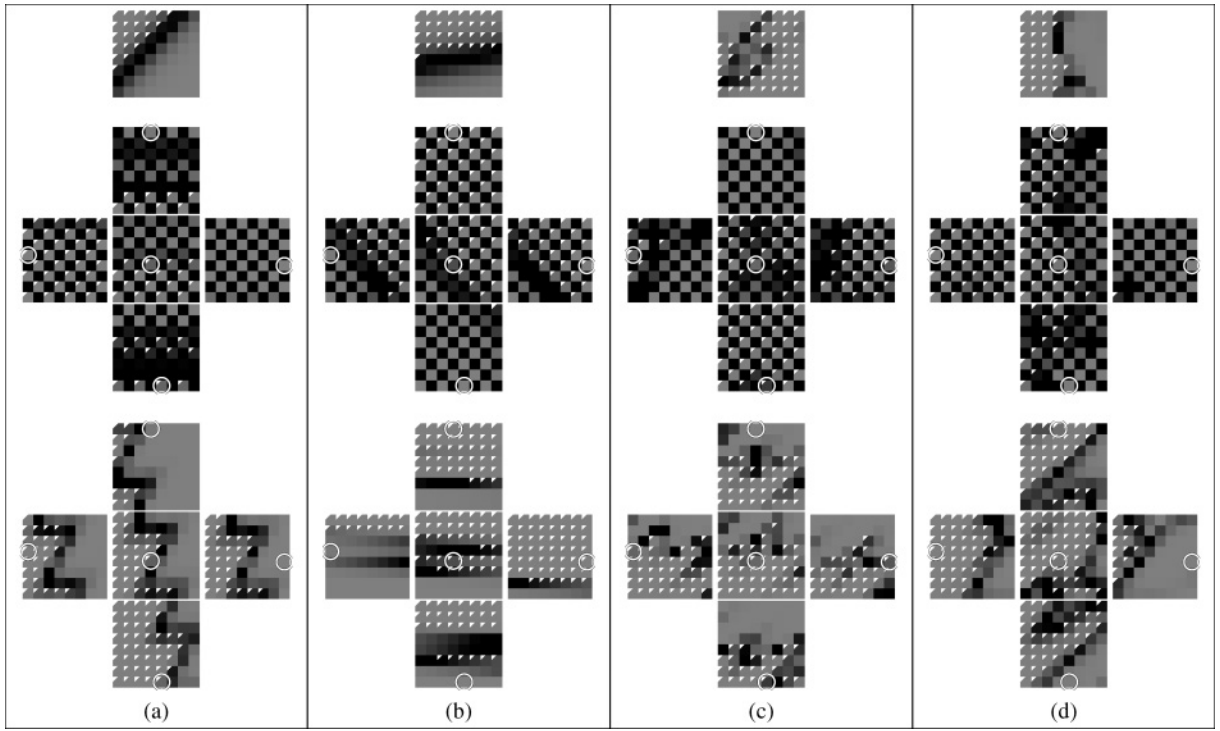


Figure 13: Connectivity patterns of less general solutions. The influence maps and receptive fields shown in this figure are for solutions that are less general than those in Figure 12. Interestingly, the connectivity patterns of less general solutions are markedly jagged and discontinuous. This property is indicative of overspecialization to the training heuristic

deterministic heuristic during training). The irregularity of the jagged solutions, which nevertheless beat the training heuristic, is an artifact of the peculiarities of the heuristic itself and not always useful when playing even slightly altered strategies.

Interestingly, smooth regularity is natural to represent *only* in a geometric context. After all, the connectivity that emanates from each input neuron in Figure 12 varies in a regular geometric fashion as the source neuron shifts position across the board (observe the pattern at different locations within each cross). Only an indirect encoding that describes connectivity as a function of geometry is likely to consistently yield such regularities. A direct encoding cannot describe how a pattern varies smoothly over space. Therefore, in a direct encoding, each individual connection is learned separately, most likely yielding jagged patterns. (Note that because direct encodings do not have a geometry, they cannot be visualized in this way; however, it is exactly this fact that prevents them from expressing smooth patterns that vary across geometry.) In fact, if the patterns yielded by direct encodings could be situated geometrically, they would likely be significantly more irregular than even those in Figure 13, which still are at least the product of indirect encoding.

In effect, the patterns in Figure 12 are *evolved topographic maps* for the game of checkers. A special (and unique) kind of receptive field is evolved in each case that moves in a predictable regular fashion across the hidden layer in accordance with the position of the source neuron from the input layer. In the less general solutions (see Figure 13), these maps are less regular and more distorted, hurting generalization.

Another important observation about the general patterns in Figure 12 is that while they are all regular and smooth, they are also all different from each other. That is, the receptive field structure in Figure 12a is unlike Figures 12b, 12c, and 12d; Figure 12b is unlike Figures 12a, 12c, and 12d, and so on. Therefore, interestingly, the implication is that as long as smoothness and regularity are achieved in the influence maps, there are multiple ways to solve the same problem effectively, perhaps explaining why HyperNEAT beats the heuristic so quickly while still generalizing often. It is the bias toward smooth topographic maps that portends the ability to generalize.

As explained in Figure 11, Figures 14 through 21 show how these connectivity patterns, both general and less general, integrate to evaluate actual board positions encountered in the game tree. It is important to note that the precise value of the output node activation (shown at the top of each panel) is not important; because evaluations are performed within an alpha-beta search, only the relative output affects decision making. For example, a negative output value does not necessarily indicate a poor evaluation, and vice versa. It is also important to note that differences in board positions are often reflected in subtle changes in neural activation among hidden nodes. Thus, it is occasionally difficult to perceive these changes visually. Nevertheless, they are often perceptible by comparing activation patterns

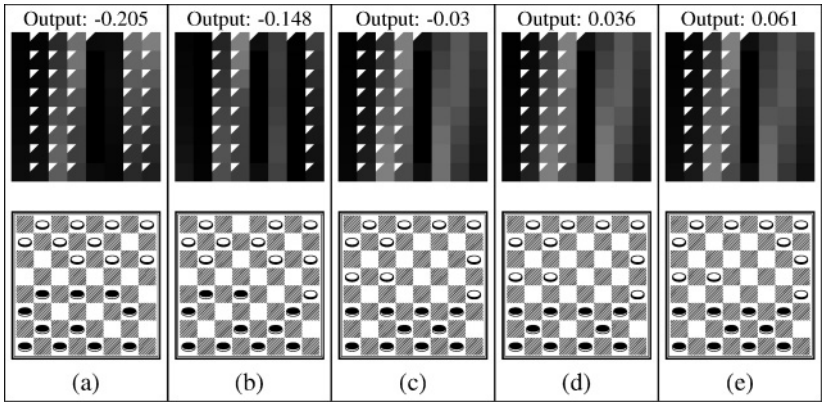


Figure 14: Board positions and associated hidden-layer activation patterns encountered by alpha-beta search for the general solution shown in Figure 12a. This substrate prefers black density in the lower-right sector of the board. As a result, black aims to bunch into a group. This structure prevents any single piece from being taken.

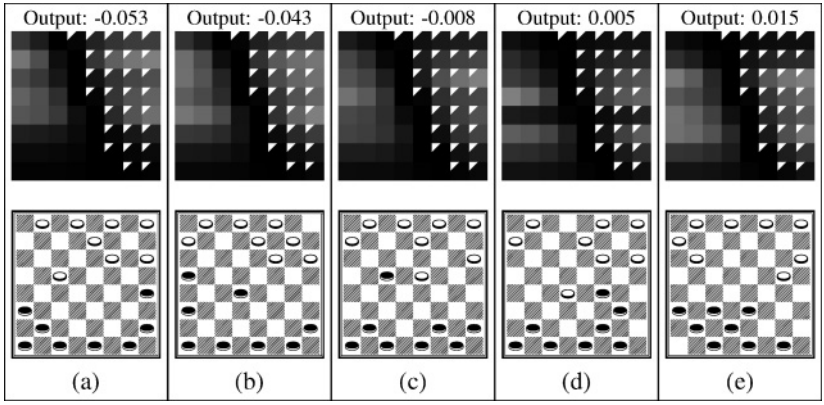


Figure 15: Board positions and associated hidden-layer activation patterns encountered by alpha-beta search for the general solution shown in Figure 12b. Like the solution in Figure 14, this substrate also prefers a defensive stance with density in the back rows. However, unlike in Figure 14, this strategy prefers density on the left.

closely, and their subtlety signifies the precision with which the substrate disambiguates similar board positions.

Through Figures 14 to 21, it is once again clear that there are many ways to solve checkers against the heuristic. However, it is also apparent that the overall activation patterns on the hidden layer exhibit definite *shapes* that

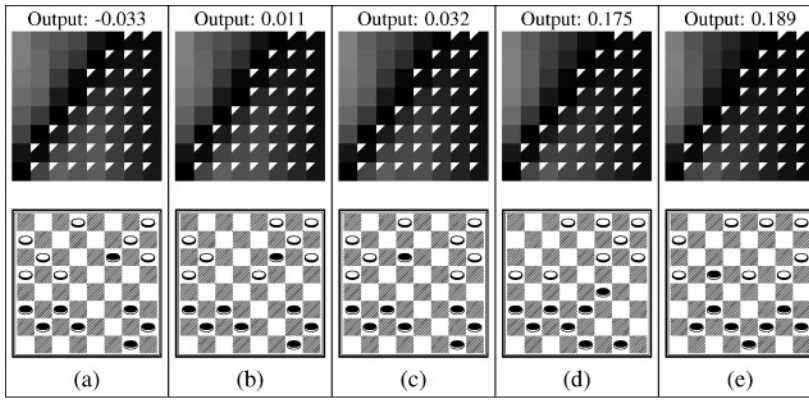


Figure 16: Board positions and associated hidden-layer activation patterns encountered by alpha-beta search for the general solution shown in Figure 12c. (a) The breakaway black piece has no chance to escape, yielding a low evaluation. In evaluation (b), the piece still has no escape but is unable to be taken directly, producing a slightly higher score. Improving the situation slightly again (c), the piece is not in immediate danger, but it is too far up the board to be defended. The situation is best in (e) because although black's piece is far up the board, it is backed up by additional pieces nearby.

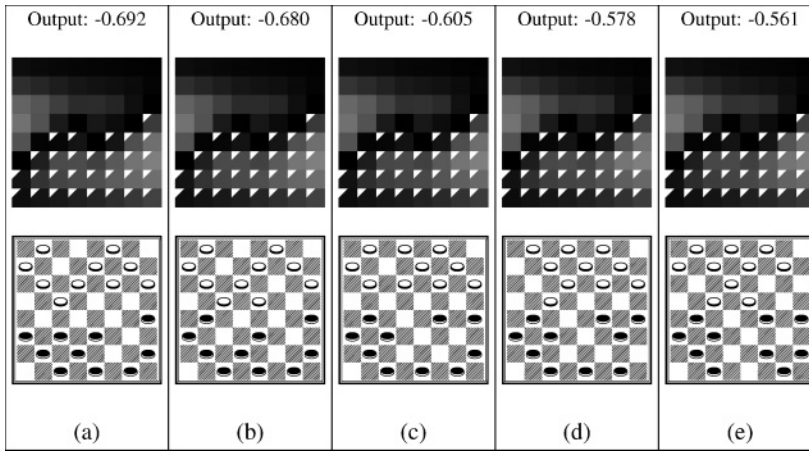


Figure 17: Board positions and associated hidden-layer activation patterns encountered by alpha-beta search for the general solution shown in Figure 12d. In the lowest scoring evaluation (a), only a single black piece is near the opponent's side of the board. There is an additional such black piece in (b) and three such black pieces in (c). In (d), the three black pieces are better defended, and in evaluation (e), white will be forced to take and will be less one piece in the center as a result. Thus, this substrate favors an aggressive stance.

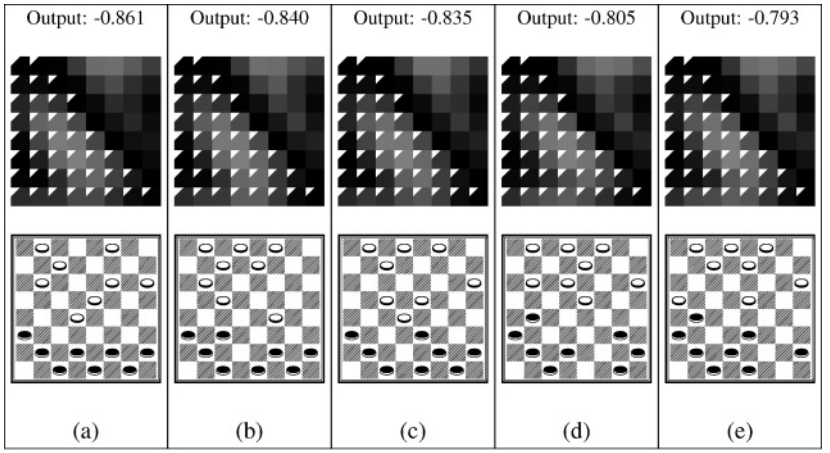


Figure 18: Board positions and associated hidden-layer activation patterns encountered by alpha-beta search for the less general solution shown in Figure 13a. (a) There are several white pieces in the center. The black pieces creep forward in (b) and (c). In (d), white pieces do not have control of the center, and in (e), white has even less material in the center. However, this less general solution considers (e) a good move even though the white piece will double-jump on its next turn.

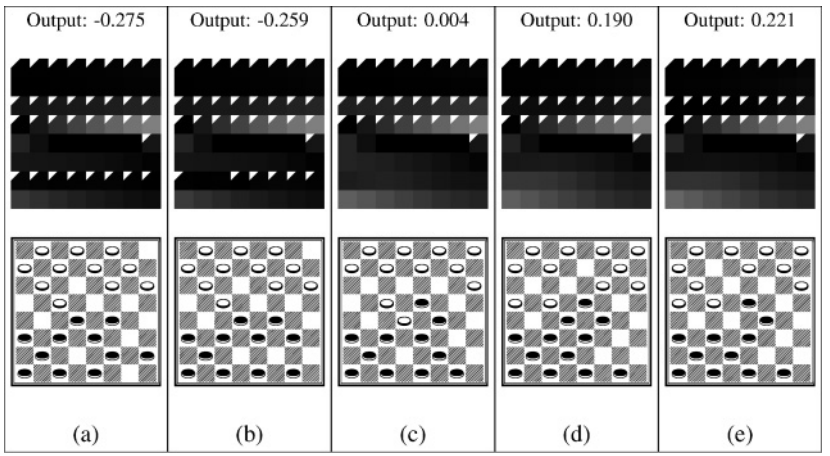


Figure 19: Board positions and associated hidden-layer activation patterns encountered by alpha-beta search for less general solution shown in Figure 13b. Moving from left (a) to right (e), black pieces assume more control of the center. However, this less general solution rates (d) highly, even though the white piece in the middle will double-jump the center black pieces on the next turn.

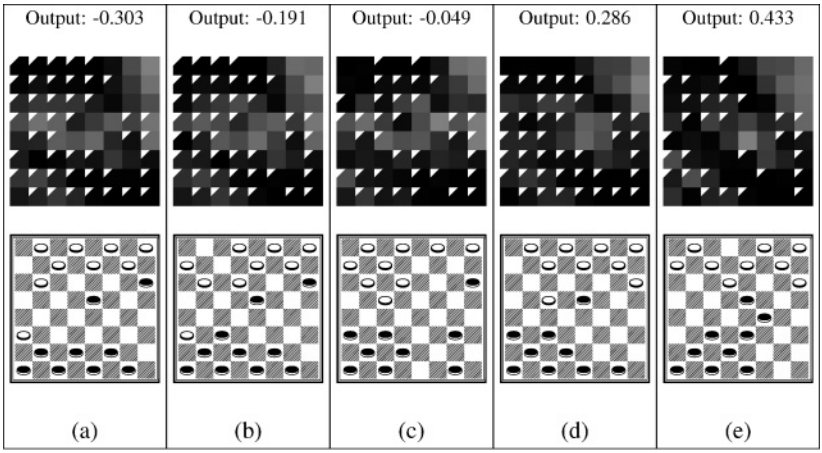


Figure 20: Board positions and associated hidden-layer activation patterns encountered by alpha-beta search for the less general solution shown in Figure 13c. (a) Two black pieces are far up the board, leaving little control of the center. As the evaluations improve in (b) through (e), black gains a stronger foothold on the center of the board and better support for pieces in white's territory. However, this less general solution favors (d) even though black's control of the center is prone to attack from white.

are tied inextricably to the geometry of the board itself. That is, areas of high activation are normally all adjacent and separated from areas of low activation, even if these areas appear in different locations for different solutions. Thus, the overall activation patterns, which are realized through the individual neural connectivity patterns in Figure 12, are also fundamentally geometric, combining the joint assessments of each individual receptive field.

Less general solutions (see Figures 18–21) yield activation patterns that are mainly jagged and discontinuous. These shapes do not resemble general solutions (see Figures 14–17), although they still exhibit patterns that are geometric. The jaggedness in the patterns suggests that the solutions are able to defeat the deterministic heuristic by memorizing certain states, and not by encoding a holistic pattern that describes the dynamics of checkers.

The general solutions (see Figures 14–17) typically favor a holistic strategy. For example, in Figures 14 and 15, keeping pieces in a tightly bound group at the back of the board is rewarded, although the lateral focus of density (left versus right) differs. In contrast, the substrates in Figures 15 and 16 favor solutions that are more aggressive and attempt to control of the center of the board. Nevertheless, the principle that unifies all these approaches is their generality; they are sensitive to relative concentrations of groupings of pieces.

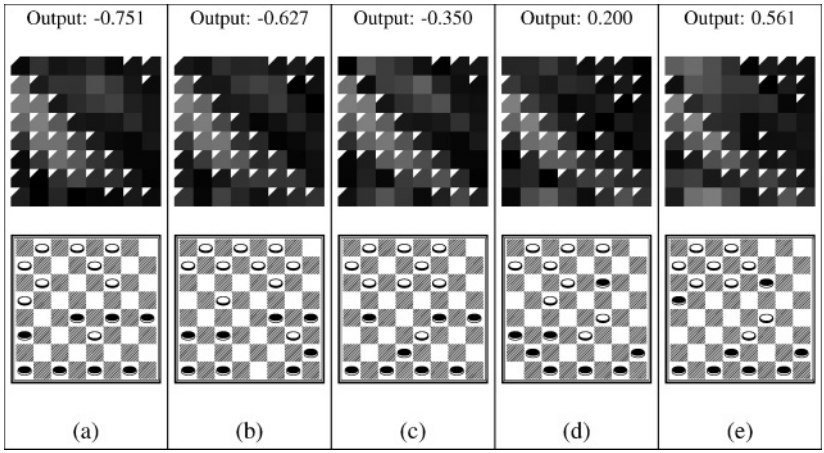


Figure 21: Board positions and associated hidden-layer activation patterns encountered by alpha-beta search for the less general solution shown in Figure 13d. Three black pieces in the center of (a) are hard to defend (they are spread out). Their position improves in (b). In (c), black also has a full back rank (all of the pieces on the back row are still in their starting configuration). One black piece is far up the board in (d), and in (e), black has both a piece far up the board and a full back rank. However, while this less general solution highly rewards (e), white is forced to capture black's most forward piece on the next turn.

Less general solutions (see Figures 18–21), while often reasonable, exhibit idiosyncratic holes in their approach that are reflected in their more piecemeal activation patterns. As described in Figures 18 to 21, such idiosyncrasies often yield specific evaluations that are fundamentally flawed. For example, position 19d is rated relatively highly yet leaves black open to a double-jump on the next turn, after black takes white's piece. Position 21e rewards black for advancing up the board, but does not account for the fact that white will capture the leading piece on the next turn. The fact that solutions that do not have a smooth geometry make such mistakes despite defeating the deterministic heuristic further suggests that generality is linked to smooth geometry.

The analysis in this section shows what it *means* to learn from geometry. In effect, learning from geometry means being able to correlate topographic maps to the geometry of the world. This ability affords smooth regular connectivity patterns, which this section showed are often correlated to the more general checkers players. The next section explores the deeper implications of this discovery and what it means for artificial evolution in general.

6 Discussion and Future Work

A major difference between traditional multilayer perceptrons (MLPs; McClelland, Rumelhart, & Hinton, 1986) and biological brains is that real

brains profusely exploit topographic relationships (Sporns, 2002). Some artificial neural models such as self-organizing maps (SOMS; Kohonen, 1981) and cortical models (Bednar and Miikkulainen, 2006; Bednar et al., 2002; Swindale, 1996) exhibit geometric structure, but the connectivity and topographic correlations in such models are not evolved. A key contribution of this letter is to show that it is possible for an evolutionary algorithm to actually evolve its own topographic maps that are domain appropriate. This development is intriguing because it means that neuroevolution algorithms can now produce structures more reminiscent of biological brains.

Furthermore, an important result is to show *why* evolving such structures is advantageous. In particular, at least in checkers, visualizing artificially evolved influence maps and receptive fields suggests an intimate connection between generalization and geometry. The connectivity patterns that exhibit smoothness, a qualitative assessment of the gradient across the hypercube of connection weights, were shown to be correlated to generalization, a quantitative assessment of the substrate against new opponents.

HyperNEAT is biased toward creating general players because the low complexity of the initial population of CPPNs tends to start evolution with simple, smooth geometries. However, it is not guaranteed to produce general results; several runs yielded less general solutions. Nevertheless, it is important to note that even these less general solutions still generalize significantly better than NEAT-EI on average, suggesting that NEAT-EI cannot easily encode the same smooth regularities demonstrated in HyperNEAT. Even worse, no runs of regular NEAT or FT-NEAT were able to defeat the *deterministic* heuristic in training, illustrating the necessity of capturing at least some geometric regularities, whether through an indirect encoding such as HyperNEAT or an engineered topology such as NEAT-EI. While engineering geometry into the network connectivity (as with NEAT-EI) provides NEAT a necessary advantage, it is not able to outperform HyperNEAT's ability to learn from geometry.

HyperNEAT solutions generalize significantly better than NEAT-EI solutions even though both methods trained against (and eventually defeated) the *same* heuristic in training. This difference is explained by HyperNEAT's indirect encoding: because HyperNEAT CPPNs initially are much smaller than NEAT-EI genomes, they are biased toward representing substrates that are highly regular but also successful in the domain of checkers. Because the direct encoding must learn each link weight individually, it searches through a comparatively high-dimensional space of neural networks, while the indirect encoding searches through a compressed (and hence lower-dimensional) space of solutions by leveraging its more powerful representation. HyperNEAT's representation naturally describes the geometric regularities of the problem domain. This capability helps in checkers because the domain (like many others) is inherently geometric. For example, the same rules generally apply to each piece at any position. Thus, the

domain of checkers helps to illustrate the advantage of an indirect encoding based on geometry such as in HyperNEAT.

However, the scope of domains that are inherently geometric is not limited to checkers and other board games. For example, Clune et al. (2008; Clune, Beckmann, Ofria, & Pennock, 2009; Clune, Ofria, & Pennock, 2009); D'Ambrosio and Stanley (2007, 2008), Gauci and Stanley (2007), and Stanley et al. (2009) show that visual discrimination and robot control domains can also benefit from indirect encoding through geometry. The inspirations for such domains are the vision and control systems of the human brain. In fact, topographic maps, which often have a geometry isomorphic to the external environment, are studied in the context of biological brains (Chklovskii & Koulakov, 2004; Churchland, 1986; Goodhill & Carreira-Perpinn, 2002; Kandel, Schwartz, & Jessell, 2000; Sporns, 2002). For example, the somatotopic representation of the human body in the brain exhibits a similar geometry to the body itself (Nakamura et al., 1998; Yang, Gallen, Schwartz, & Bloom, 1993). Interestingly, ANNs evolved with HyperNEAT have receptive fields and influence maps that can be visualized much like such topographic maps in biological brains. Thus, topographic maps in ANNs evolved by HyperNEAT are reminiscent of their more sophisticated biological counterparts, suggesting the start of an intriguing new direction of research in artificially evolved substrates. In addition, not only do such maps arise, but in this letter, their analysis helped to establish a connection between the smoothness and regularity of the geometry of such a map and its generalization. As a result, the surprising insight is that a qualitative assessment of evolved topographic maps translates directly to quantitative performance results in the generalization test.

Interestingly, any domain that exists in a space of multiple dimensions contains at least an implicit geometry that can be potentially exploited through an indirect encoding based on geometry. Clune et al. (2008) demonstrate that even when the geometry of an ANN that controls a robot is scrambled, HyperNEAT is able to find regularities within the scrambled geometry. Thus, future work for this approach will explore other challenging domains. The board game Go is one such appealing candidate for further research because its geometry is similar to that of checkers. Beyond board games, promising avenues include continuing work in vision (Gauci & Stanley, 2007) and robot control (D'Ambrosio & Stanley, 2007). While early such work focused on relatively simple problems, it is not known how close evolved indirect encodings can approach the complexity of biological brains, which are clearly suited for such tasks. Even if approaches such as HyperNEAT do not reach such ambitious scale, lessons learned along the way, such as the connection between smooth geometry and generalization, promise to be illuminating.

For example, an interesting question is whether ANNs evolved by HyperNEAT for visual tasks might resemble features in V1 or other parts of the biological visual processing hierarchy (Bednar & Miikkulainen,

2006; Hubel, 1988). While the human primary visual cortex contains about 140 million neurons (Leuba & Kraftsik, 1994), HyperNEAT has evolved functional networks with millions of connections (Stanley et al., 2009). Furthermore, while biological brains (including the visual cortex) exhibit synaptic plasticity (Hubel, Wiesel, & LeVay, 1977), ANNs with plastic synapses have been evolved in the past (Floreano & Urzelai 2000, 2001; Floreano & Mondada, 1996; Risi et al., 2009; Soltoggio et al., 2008), and in principle, HyperNEAT can potentially evolve the *geometry of the learning rules*, taking it another step closer to biological plausibility. That is, HyperNEAT can potentially assign plasticity roles to connections in a geometric pattern, which is necessary if plastic structures with millions of connections are to be evolved. Thus, while the evolved maps in this letter are static, in principle the capability to encode such maps suggests that evolving plastic maps with similar properties (e.g., geometry) is plausible.

Stepping back to the role of geometry in learning algorithms today, the results and analysis in this letter suggest that an important prerequisite to exploiting geometry in learning is to be *aware* of it. That the CPPNs in HyperNEAT literally see the positions of the nodes being connected affords the ability to exploit the domain geometry by creating smooth, semiregular patterns. To date, this ability to *see* the geometry of the substrate is unique, yet it portends the importance of endowing future algorithms with a similar capability if they are to exploit domain geometry effectively. Simply imagining trying to learn tic tac toe, a simple game, on a scrambled board illustrates the urgency of this consideration. Once the capability to perceive geometry is made available, an exciting new research direction with interesting biologically parallels opens up.

7 Conclusion

This letter argued that representing evolved ANNs as indirect functions of their geometry evolves structures that are closer to structures seen in biological brains than those evolved by prior NE approaches. In addition, such a method is able to exploit the underlying geometric regularities in a problem to quickly find elegant solutions to complex problems, provided that geometry plays a role in the problem domain.

The role of geometry was shown to be potentially useful to machine learning performance in the domain of checkers. Regular NEAT and FT-NEAT were not able to defeat the deterministic heuristic in a single run of training, while NEAT-EI and HyperNEAT were able to defeat the heuristic in all 20 runs. HyperNEAT was able to find solutions relatively quickly by searching through the low-dimensional space of CPPNs, while NEAT-EI took significantly longer, searching through the high-dimensional space of ANNs. In addition, solutions produced by HyperNEAT generalized significantly better than solutions produced by NEAT-EI, suggesting a link between HyperNEAT's perception of geometry and generality.

This link was confirmed through a visual study of general and less general HyperNEAT solutions and their performance in training. A correlation was drawn between the smoothness and continuity of connectivity patterns across the layers of solutions and their generalization performance, suggesting that general solutions encode ANNs that are smooth and regular, while less general solutions encode ANNs that are jagged and discontinuous. The CPPNs that encoded jagged ANNs were specialized for the specific games of checkers seen in training, while the CPPNs that encoded smooth ANNs were more general checkers players.

These results suggest that NE methods should ideally see the geometry of the domain and be able to encode and represent geometry in a way that creates smooth and regular ANNs. In this way, the ANNs produced by NE can more closely resemble neural networks seen in natural brains. In the future, the results of such artificial evolutionary approaches will offer increasing relevance to researchers beyond the field of NE.

Acknowledgments

Special thanks to Martin Fierz for providing the Simplech checkers engine. Special thanks also to our anonymous reviewers for helping to improve this work significantly.

References

Aaltonen, T., Adelman, J., Akimoto, T., Albrow, M. G., Alvarez Gonzalez, B., Almerio, S., et al. (2009). Measurement of the top quark mass with dilepton events selected using neuroevolution at CDF. *Physical Review Letters*, *102*, 152001–152008.

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.

Andrade, M. A., Muro, E. M., & Morán, F. (2001). Simulation of plasticity in the adult visual cortex. *Biological Cybernetics*, *84*(6), 445–451.

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.

Astor, J. S., & Adami, C. (2000). A developmental model for the evolution of artificial neural networks. *Artificial Life*, *6*(3), 189–218.

Bednar, J. A., Kelkar, A., & Miikkulainen, R. (2002). Modeling large cortical networks with growing self-organizing maps. *Neurocomputing*, 44–46, 315–321.

Bednar, J. A., & Miikkulainen, R. (2006). Joint maps for orientation, eye, and direction preference in a self-organizing model of V1. *Neurocomputing*, 69(10–12), 1272–1276.

Beer, R. D. (2000). Dynamical approaches to cognitive science. *Trends in Cognitive Sciences*, *4*, 91–99.

Beer, R., & Gallagher, J. (1992). Evolving dynamical neural networks for adaptive behavior. *Adaptive Behavior*, *1*(1), 91.

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: Morgan Kaufmann.

Bongard, J. C. (2002). Evolving modular genetic regulatory networks. In *Proceedings of the 2002 Congress on Evolutionary Computation*. San Mateo, CA: IEEE Computer Society.

Braun, H., & Weisbrod, J. (1993). Evolving feedforward neural networks. In *Proceedings of ANNGA93, International Conference on Artificial Neural Networks and Genetic Algorithms*. Berlin: Springer.

Chellapilla, K., & Fogel, D. B. (2001). Evolving an expert checkers playing program without using human expertise. *IEEE Trans. Evolutionary Computation*, *5*(4), 422–428.

Chklovskii, D. B., & Koulakov, A. A. (2004). Maps in the brain: What can we learn from them? *Annual Review of Neuroscience*, *27*, 369–392.

Choe, Y., & Miikkulainen, R. (2004). Contour integration and segmentation in a self-organizing map of spiking neurons. *Biological Cybernetics*, *90*, 75–88.

Churchland, P. M. (1986). Some reductive strategies in cognitive neurobiology. *Mind*, *95*, 279–309.

Clune, J., Beckmann, B., Ofria, C., & Pennock, R. (2009). Evolving coordinated quadruped gaits with the hyperneat generative encoding. In *IEEE Congress on Evolutionary Computation*. Piscataway, NJ: IEEE Press.

Clune, J., Ofria, C., & Pennock, R. T. (2008). How a generative encoding fares as problem-regularity decreases. In *Proceedings of the 10th International Conference on Parallel Problem Solving from Nature* (pp. 358–367). Berlin: Springer-Verlag.

Clune, J., Ofria, C., & Pennock, R. (2009). The sensitivity of hyperneat to different geometric representations of a problem. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation* (pp. 675–682). New York: ACM.

D'Ambrosio, D., & Stanley, K. O. (2007). A novel generative encoding for exploiting neural network sensor and output geometry. In *Proceedings of the Genetic and Evolutionary Computation Conference*. New York: ACM Press.

D'Ambrosio, D. B., & Stanley, K. O. (2008). Generative encoding for multiagent learning. In *Proceedings of the Genetic and Evolutionary Computation Conference*. New York: ACM Press.

Dellaert, F., & Beer, R. D. (1994). *Co-evolving body and brain in autonomous agents using a developmental model* (Tech. Rep. CES-94-16). Cleveland, OH: Department of Computer Engineering and Science, Case Western Reserve University.

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.

Fierz, M. (2002). *Xcheckers, Cliche, and GPL-Cake*. Available online at Simplech: http://arton.cunst.net/xcheckers/.

Floreano, D., Dürr, P., & Mattiussi, C. (2008). Neuroevolution: From architectures to learning. *Evolutionary Intelligence*, *1*(1), 47–62.

Floreano, D., & Mondada, F. (1996). Evolution of plastic neurocontrollers for situated agents. *IEEE Transactions on Systems, Man, and Cybernetics*, *26*(3), 396–407.

Floreano, D., & Urzelai, J. (2000). Evolutionary robots with online self-organization and behavioral fitness. *Neural Networks*, *13*, 431–443.

Floreano, D., & Urzelai, J. (2001). Evolution of plastic control networks. *Autonomous Robots*, *11*(3), 311–317.

Fogel, D. (1992). *Evolving artificial intelligence*. Unpublished doctoral dissertation, University of California, San Diego.

Fogel, D. (1993). Using evolutionary programing to create neural networks that are capable of playing tic-tac-toe. In *IEEE International Conference on Neural Networks, 1993* (pp. 875–880). Piscataway, NJ: IEEE.

Fogel, D. B. (2002). *Blondie24: Playing at the edge of AI*. San Francisco: Morgan Kaufmann.

Fogel, D., Fogel, L., & Porto, V. (1990). Evolving neural networks. *Biological Cybernetics*, *63*(6), 487–493.

Fogel, D., Hays, T., Hahn, S., & Quon, J. (2004). A self-learning evolutionary chess program. *Proceedings of the IEEE*, *92*(12), 1947–1954.

Gauci, J., & Stanley, K. O. (2007). Generating large-scale neural networks through discovering geometric regularities. In *Proceedings of the Genetic and Evolutionary Computation Conference*. New York: ACM Press.

Gomez, F., & Miikkulainen, R. (1997). Incremental evolution of complex general behavior. *Adaptive Behavior*, *5*, 317–342.

Gomez, F., Schmidhuber, J., & Miikkulainen, R. (2008). Accelerated neural evolution through cooperatively coevolved synapses. *J. Mach. Learn. Res.*, *9*, 937–965.

Goodhill, G. J., & Carreira-Perpinn, M. A. (2002). Cortical columns. In L. Nadel (Ed.), *Encyclopedia of cognitive science* (vol. 1, pp. 845–851). New York: Macmillan.

Gruau, F. (1995). Automatic definition of modular neural networks. *Adaptive Behaviour*, *3*(2), 151–183.

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.

Harp, S. A., Samad, T., & Guha, A. (1989). Towards the genetic synthesis of neural network. In *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 360–369). San Francisco: Morgan Kaufmann.

Hornby, G. (2004). Shortcomings with tree-structured edge encodings for neural networks. In *Proceedings of the Genetic and Evolutionary Computation Conference*. Berlin: Springer-Verlag.

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*. Piscataway, NJ: IEEE Press.

Hornby, G. S., & Pollack, J. B. (2002). Creating high-level components with a generative representation for body-brain evolution. *Artificial Life*, *8*(3), 600–607.

Hubel, D. H. (1988). *Eye, brain, and vision*. New York: Freeman.

Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology*, *160*, 106–154.

Hubel, D. H., & Wiesel, T. N. (1968). Receptive fields and functional architecture of monkey striate cortex. *Journal of Physiology*, *195*, 215–243.

Hubel, D. H., Wiesel, T. N., & LeVay, S. (1977). Plasticity of ocular dominance columns in monkey striate cortex. *Philosophical Transactions of the Royal Society of London, Series B, Biological Sciences*, *278*, 377–409.

Husbands, P., Smith, T., Jakobi, N., & Shea, M. (1998). Better living through chemistry: Evolving GasNets for robot control. *Connection Science*, *10*(3), 185–210.

Kampfner, R., & Conrad, M. (1983). Computational modeling of evolutionary learning processes in the brain. *Bulletin of Mathematical Biology*, *45*(6), 931–968.

Kandel, E. R., Schwartz, J. H., & Jessell, T. M. (2000). *Principles of neural science* (4th ed.). New York: McGraw-Hill.

Kitano, H. (1990). Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, *4*, 461–476.

Knuth, D., & Moore, R. (1975). An analysis of alpha-beta pruning. *Artificial Intelligence*, *6*(4), 293–326.

Kohonen, T. (1981). Automatic formation of topological maps of patterns in a self-organizing system. In *Proceedings of the 2nd Scandinavian Conference on Image Analysis* (pp. 214–220). Espoo, Finland: Pattern Recognition Society of Finland.

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

Koza, J. R., & Rice, J. P. (1991). Genetic generation of both the weights and architecture for a neural network. In *International Joint Conference on Neural Networks* (pp. 397–404). Piscataway, NJ: IEEE.

Leuba, G., & Kraftsik, R. (1994). *Changes in volume, surface estimate, three-dimensional shape and total number of neurons of the human primary visual cortex from midgestation until old age*. Berlin: Springer.

Lindenmayer, A. (1974). Adding continuous components to L-systems. In G. Rozenberg & A. Salomaa (Eds.), *L Systems*. Berlin: Springer-Verlag.

Lipson, H., & Pollack, J. B. (2000). Automatic design and manufacture of robotic lifeforms. *Nature*, *406*, 974–978.

Mattiussi, C., & Floreano, D. (2007). Analog genetic encoding for the evolution of circuits and networks. *IEEE Transactions on Evolutionary Computation*, *11*(5), 596–607.

McClelland, J. L., Rumelhart, D. E., & Hinton, G. E. (1986). The appeal of parallel distributed processing. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition* (Vol. 1, *foundations*, pp. 3–44). Cambridge, MA: MIT Press.

McHale, G., & Husbands, P. (2004). Quadrupedal locomotion: GasNets, CTRNNs and hybrid CTRNN/PNNs compared. In *Artificial life IX: Proceedings of the Ninth International Conference on the Simulation and Synthesis of Artificial Life* (p. 106). Cambridge, MA: MIT Press.

Miikkulainen, R., Bednar, J. A., Choe, Y., & Sirosh, J. (2005). *Computational maps in the visual cortex*. Berlin: Springer.

Miikkulainen, R., & Dyer, M. G. (1989). Encoding input/output representations in connectionist cognitive systems. In D. S. Touretzky, G. E. Hinton, & T. J. Sejnowski (Eds.), *Proceedings of the 1988 Connectionist Models Summer School* (pp. 347–356). San Francisco: Morgan Kaufmann.

Miller, G. F., Todd, P. M., & Hegde, S. U. (1989). Designing neural networks using genetic algorithms. In *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 379–384). San Francisco: Morgan Kaufmann.

Montana, D. J., & Davis, L. (1989). Training feedforward neural networks using genetic algorithms. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence* (pp. 762–767). San Francisco: Morgan Kaufmann.

Moriarty, D. E., & Miikkulainen, R. (1996). Efficient reinforcement learning through symbiotic evolution. In L. P. Kaelbling (Ed.), *Recent advances in reinforcement learning*. Dordrecht: Kluwer.

Nakamura, A., Yamada, T., Goto, A., Kato, T., Ito, K., Abe, Y., et al. (1998). Somatosensory homunculus as drawn by MEG. *Neuroimage*, *7*(4), 377–386.

Opitz, D. W., & Shavlik, J. W. (1997). Connectionist theory refinement: Genetically searching the space of network topologies. *Journal of Artificial Intelligence Research*, *6*, 177–209.

Pujol, J. C. F., & Poli, R. (1998). Evolving the topology and the weights of neural networks using a dual representation. *Applied Intelligence Journal*, *8*(1), 73–84.

Risi, S., Vanderbleek, S. D., Hughes, C. E., & Stanley, K. O. (2009). How novelty search escapes the deceptive trap of learning to learn. In *Proceedings of the Genetic and Evolutionary Computation Conference*. New York: ACM Press.

Rosen, Y., & Lenkinski, R. E. (2007). Recent advances in magnetic resonance neurospectroscopy. *Neurotherapeutics*, 330–345.

Russell, S., Norvig, P., Canny, J., Malik, J., & Edwards, D. (1995). *Artificial intelligence: A modern approach*. Englewood Cliffs, NJ: Prentice Hall.

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.

Soltoggio, A., Bullinaria, J. A., Mattiussi, C., Dürr, P., & Floreano, D. (2008). Evolutionary advantages of neuromodulated plasticity in dynamic, reward-based scenarios. *Artificial Life, 11*, pp. 569–576.

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

Srinivas, M., & Paranaik, L. (1991). Learning neural network weights using genetic algorithms-improving performance by search-space reduction. *IEEE International Joint Conference on Neural Networks*, *3*, 2331–2336.

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

Stanley, K. O., Bryant, B. D., & Miikkulainen, R. (2005a). Evolving neural network agents in the NERO video game. In *Proceedings of the IEEE 2005 Symposium on Computational Intelligence and Games*. Piscataway, NJ: IEEE.

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

Stanley, K. O., D'Ambrosio, D. B., & Gauci, J. (2009). A hypercube-based encoding for evolving large-scale neural networks. *Artificial Life*, *15*, 185–212.

Stanley, K. O., Kohl, N., & Miikkulainen, R. (2005). Neuroevolution of an automobile crash warning system. In *Proceedings of the Genetic and Evolutionary Computation Conference*. New York: ACM.

Stanley, K. O., & Miikkulainen, R. (2002a). Efficient evolution of neural network topologies. In *Proceedings of the 2002 Congress on Evolutionary Computation*. Piscataway, NJ: IEEE.

Stanley, K. O., & Miikkulainen, R. (2002b). Efficient reinforcement learning through evolving neural network topologies. In *Proceedings of the Genetic and Evolutionary Computation Conference*. San Francisco: Morgan Kaufmann.

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

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

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

Swindale, N. V. (1996). The development of topography in the visual cortex: A review of models. *Network: Computation in Neural Systems*, *7*, 161–247.

Taylor, M. E., Whiteson, S., & Stone, P. (2006). Comparing evolutionary and temporal difference methods in a reinforcement learning domain. In *GECCO 2006: Proceedings of the Genetic and Evolutionary Computation Conference* (pp. 1321–1328). New York: ACM.

Togelius, J., & Lucas, S. M. (2005). Forcing neurocontrollers to exploit sensory symmetry through hard-wired modularity in the game of cellz. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games* (pp. 37–43). Piscataway, NJ: IEEE.

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

Weliky, M., Bosking, W. H., & Fitzpatrick, D. (1996). A systematic map of direction preference in primary visual cortex. *Nature*, *379*, 725–728.

Whiteson, S., Kohl, N., Miikkulainen, R., & Stone, P. (2003). Evolving RoboCup keepaway players through task decomposition. In *Proceedings of the Genetic and Evolutionary Computation Conference* (pp. 356–368). San Francisco: Morgan Kaufmann.

Whitley, D. (1995). Genetic algorithms and neural networks. In J. Periaux, M. Galan, & P. Cuesta (Eds.), *Genetic algorithms in engineering and computer science* (pp. 203–216). Hoboken, NJ: Wiley.

Wieland, A. (1991). Evolving neural network controllers for unstable systems. In *Proceedings of the International Joint Conference on Neural Networks* (pp. 667–673). Piscataway, NJ: IEEE.

Yang, T., Gallen, C., Schwartz, B., & Bloom, F. (1993). Noninvasive somatosensory homunculus mapping in humans by using a large-array biomagnetometer. *Proceedings of the National Academy of Sciences*, *90*(7), 3098–3102.

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

Zhang, B.-T., & Muhlenbein, H. (1993). Evolving optimal neural networks using genetic algorithms with Occam's razor. *Complex Systems*, *7*, 199–220.

Received June 24, 2009; accepted November 21, 2009.