Guided Self-Organization in Indirectly Encoded and Evolving Topographic Maps

In: *Proceedings of the Genetic and Evolutionary Computation Conference* (GECCO 2014). New York, NY: ACM

Sebastian Risi
Center for Computer Games Research
IT University of Copenhagen
Copenhagen, Denmark
sebr@itu.dk

Kenneth O. Stanley
Department of EECS
University of Central Florida
Orlando, FL 32816, USA
kstanley@eecs.ucf.edu

ABSTRACT

An important phenomenon seen in many areas of biological brains and recently in deep learning architectures is a process known as self-organization. For example, in the primary visual cortex, color and orientation maps develop based on lateral inhibitory connectivity patterns and Hebbian learning dynamics. These *topographic maps*, which are found in all sensory systems, are thought to be a key factor in enabling abstract cognitive representations. This paper shows for the first time that the Hypercube-based NeuroEvolution of Augmenting Topologies (HyperNEAT) method can be seeded to begin evolution with such lateral connectivity, enabling genuine self-organizing dynamics. The proposed approach draws on HyperNEAT's ability to generate a pattern of weights across the connectivity of an artificial neural network (ANN) based on a function of its geometry. Validating this approach, the afferent weights of an ANN self-organize in this paper to form a genuine topographic map of the input space for a simple line orientation task. Most interestingly, this seed can then be evolved further, providing a method to guide the self-organization of weights in a specific way, much as evolution likely guided the self-organizing trajectories of biological brains.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning – connectionism and neural nets

General Terms

Algorithms

Keywords

Self-Organization; Topographic Maps; Plastic Neural Networks; Adaptation; Learning; Neuroevolution; CPPNs; HyperNEAT

1. INTRODUCTION

While *neuroevolution* (NE), i.e. evolving artificial neural networks (ANNs) through evolutionary algorithms, has produced significant results in a variety of different domains [7, 28, 33], the ANNs evolved through NE still often do not include some of the important features seen in natural brains. One example of such a feature is neural plasticity, which allows a plastic ANN to change its internal connection weights during its lifetime based on Hebbian learning. This synaptic plasticity underlies the ability of real brains to learn and to adapt to sudden environmental changes. While there has been progress in incorporating plasticity into NE algorithms [6, 14, 16, 19, 23], so far these approaches are only applied to simple toy problems and do not exhibit many of the important plasticity-enabled features seen in natural brains.

One such "hallmark feature of vertebrate brains" [31] is topographic maps, which are areas of the brain that are spatially organized based on some feature of sensory input from the environment. In topographic maps neurons that respond to related sensory input are located near each other. For example the primary visual cortex (like many other areas in the neocortex) includes a topographic map [30], which means that it is organized such that adjacent neurons respond to adjacent regions of the visual field. Thivierge and Marcus [31] suggest that a key function of topographic maps is to enable abstract cognitive representations.

The topographic maps in the brain are themselves the result of a self-organizing process of synaptic connection weights, in which lateral inhibitory synapses and competitive learning play an important role [13]. As Miikkulainen [13] showed, a biologically realistic *hand-designed* ANN with short range excitatory and long range inhibitory connectivity allows the synaptic weights to self-organize to form a topographic map of the sensory stimuli. The lateral connections between neurons help to focus the initial activation pattern into a localized (winner-take-all) response on the map after which the connection weight are modified based on Hebbian learning.

Because evolving such plastic ANNs from scratch is likely a highly deceptive task [19], the new idea introduced in this paper is that the Hypercube-based NeuroEvolution of Augmenting Topologies (HyperNEAT) method [5, 8, 29] can be seeded to directly begin with the necessary lateral connectivity. This advance is enabled by HyperNEAT's ability to generate a pattern of weights across the connectivity of an evolving neural network based on a function of its geometry.

HyperNEAT employs an indirect encoding called compositional pattern producing networks (CPPNs) [25], which can compactly encode connectivity patterns with regularities such as symmetry, repetition, and repetition with variation. It also showed that giving neurons actual *locations* within the coordinate space of the evolving neural network allows evolution to exploit *topography* (i.e. not just topology), thereby allowing the geometry of sensors to align sensibly with the geometry of neurons in the brain. This type of geometric placement of neurons is missing from many ANN models, yet it is known to be essential to the organization and function of biological brains [24]. Because of this insight, HyperNEAT is able to evolve high-dimensional ANNs with regularities appropriate to the problem domain [4, 8, 9, 29], which is in the case of this paper a self-organizing topographic map.

More importantly, a seeded such structure can than be evolved further, allowing HyperNEAT (1) to guide the self-organization of weights towards a particular target configuration and (2) to accelerate the self-organizing process compared to a uniform weight baseline approach. To demonstrate the potential of this method, experiments in a simple line orientation task are performed, revealing that the seeded HyperNEAT approach is able to form a genuine topographic map of the input space. The main results is that seeding HyperNEAT with a connectivity pattern that encourages the emergence of topographic maps could allow NE approaches to expand the scope of neural structures that evolution can discover to dynamic structures heretofore only seen in neuroscience.

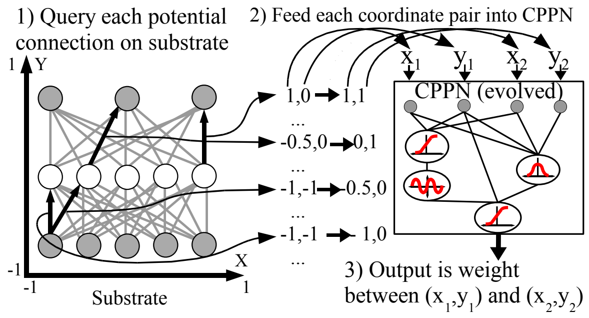

1) Query each potential connection on substrate
2) Feed each coordinate pair into CPPN
1
Y
-1
-1
X
1
Substrate
1,0 → 1,1
...
-0.5,0 → 0,1
...
-1,-1 → -0.5,0
...
-1,-1 → 1,0
...
x1
y1
x2
y2
CPPN (evolved)
3) Output is weight between (x1,y1) and (x2,y2)

Figure 1: HyperNEAT Geometric Connectivity Pattern Interpretation. A collection of nodes, called the *substrate*, is assigned coordinates that range from -1 to 1 in all dimensions. (1) Every potential connection in the substrate is queried to determine its presence and weight; the dark directed lines in the substrate depicted in the figure represent a sample of connections that are queried. (2) Internally, the CPPN (which is evolved) is a graph that determines which activation 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. For each query, the CPPN takes as input the positions of the two endpoints and (3) outputs the weight of the connection between them. Thus, CPPNs can produce regular patterns of connections in space.

2. BACKGROUND AND RELATED WORK

This section reviews the Neuroevolution of Augmenting Topologies (NEAT) and HyperNEAT methods behind the new approach presented in this paper.

2.1 Neuroevolution of Augmenting Topologies (NEAT)

The HyperNEAT method is itself an extension of the original NEAT algorithm for evolving ANNs. NEAT starts with a population of simple neural networks and then *adds complexity* over generations by adding new nodes and connections through mutations. By evolving networks in this way, the topology of the network does not need to be known a priori; NEAT searches through increasingly complex networks to find a suitable level of complexity. Because it starts simply and gradually adds complexity, it tends to find a solution network close to the minimal necessary size. However, as explained next, it turns out that directly representing connections and nodes as explicit genes in the genome cannot scale up to very large networks. For a complete overview of NEAT see Stanley and Miikkulainen [26].

2.2 HyperNEAT

NEAT is called a *direct encoding* because each part of the solution's representation (i.e. each connection weight in the genome) maps to a single piece of structure in the final solution network [7]. The significant disadvantage of this approach is that even when different parts of the solution are similar, they must be encoded and therefore discovered separately. Thus HyperNEAT introduces an *indirect* encoding instead, which means that the description of the solution is compressed such that information can be reused, allowing the final solution to contain more components than the description itself. Indirect encodings allow solutions to be represented as a *pattern* of parameters, rather than requiring each parameter to be represented individually [1, 9, 25, 27]. HyperNEAT, reviewed in this section, is an indirect encoding extension of NEAT that is proven in a number of challenging domains that require discovering regularities [3, 8, 9, 29]. For a full description see Stanley et al. [29] and Gauci and Stanley [9].

In HyperNEAT, NEAT is altered to evolve an indirect encoding called *compositional pattern producing networks* (CPPNs; [25]) *instead* of ANNs. The CPPN in HyperNEAT plays the role of DNA in nature, but at a much higher level of abstraction; in effect it encodes a pattern of weights that is painted across the geometry of a network. Yet the convenient trick in HyperNEAT is that this encoding is *itself* a network, which means that CPPNs can be evolved by NEAT. A CPPN is a *compositions of functions*, wherein each function loosely corresponds to a useful regularity. For example, a Gaussian function induces symmetry and a periodic function such as sine creates segmentation through repetition. In effect, the indirect CPPN encoding can compactly encode patterns with regularities such as symmetry, repetition, and repetition with variation [20, 25]. Thus as the CPPN increases in complexity, it encodes increasingly complex amalgamations of regularities and symmetries that are projected across the connectivity of a network [8, 9].

Unlike in many common neural network formalisms, in HyperNEAT neurons exist at *locations* in space. That way, connectivity is expressed across a geometry, like in a natural brain. Formally, CPPNs are *functions* that input the locations of nodes (i.e. the *geometry* of a network) and output weights between those locations. That way, when queried for many pairs of nodes situated in n dimensions, the result is a topographic connectivity pattern in that space. Consider a CPPN that takes four inputs labeled x_1, y_1, x_2, and y_2; this point in four-dimensional space *also* denotes 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 (figure 1). Because the connections are produced by a function of their endpoints, the final structure is produced with *knowledge* of its geometry. In effect, the CPPN is painting a pattern on the inside of a four-dimensional hypercube that is interpreted as the isomorphic connectivity pattern, which explains the origin of the name *hypercube-based NEAT* (HyperNEAT). Connectivity patterns produced by a CPPN in this way are called *substrates* so that they can be verbally distinguished from the CPPN itself, which is also a network.

Each queried point in the substrate is a node in an ANN. The experimenter defines both the location and role (i.e. 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 [3, 29]. That way, the connectivity of the substrate is a function of the task structure. For example, the sensors of a robot can be placed from left to right on the substrate in the same order that they exist on the robot. Outputs for moving left or right can also be placed in the same order, allowing HyperNEAT to understand from the outset the correlation of sensors to effectors. In this way, knowledge about the problem geometry can be injected into the search and HyperNEAT can exploit the regularities of a problem that are invisible to traditional neural network encodings.

In summary, the HyperNEAT method evolves the internal topology and weights of the CPPN that compactly *encodes* ANN weight patterns. The capabilities of HyperNEAT are important for the approach presented in this paper because the lateral connectivity patterns necessary for the self-organization of connection weights can also be expressed as a function of geometry. As explained next, an extension to HyperNEAT called adaptive HyperNEAT [16], allows not only patterns of weights across the connectivity of an ANN to be generated by a function of its geometry, but also patterns of local learning rules.

2.3 Adaptive HyperNEAT

Unlike traditional static ANNs in neuroevolution whose weights do not change during their lifetime, plastic ANNs [6, 14, 23] that play an important role in the formation of topographic maps [13, 22] can *adapt* by changing their internal connection strengths according to a Hebbian learning rule that modifies synaptic weights based on pre- and postsynaptic neuron activity.

The main idea in adaptive HyperNEAT [16] is that CPPNs can not only encode connectivity patterns but also *patterns of plasticity rules*. As in the brain, different *regions* of the ANN should be more or less plastic and employ different learning rules, which HyperNEAT can facilitate because it sees the ANN geometry. This idea is based on the insight that if a CPPN can paint a pattern of connection weights across network geometry, then it should also be able to paint a pattern of varying *plasticity rules* across the same geometry. That way, it does not need to encode each rule separately (i.e. because the CPPN is an indirect encoding). To facilitate this capability, instead of outputting only a *weight*, the Adaptive HyperNEAT variant in paper has one additional output that encodes the learning rate η of the Hebbian Rule:

$$\Delta w_{ij} = \eta \cdot o_i o_j \ . \qquad (1)$$

In this way, the CPPN computes an entire pattern of Hebbian rules with different learning rates as a function of the neural geometry in a regular pattern across the network. In this paper, adaptive HyperNEAT will be compared to a HyperNEAT variant that assigns learning rate η for every afferent connection uniformly. This comparison is designed to investigate the minimal sufficient adaptive dynamics that allow the automatic self-organization of a topographic map.

3. APPROACH: SEEDING HYPERNEAT WITH LATERAL INHIBITORY CONNECTIVITY PATTERNS

The new idea in this paper is that because the lateral connectivity patterns in topographic maps are related to their geometry, evolution in HyperNEAT can be seeded with geometric principles that bias the pattern of connectivity in a similar way. The hope is that the weights of the resulting ANNs should also self-organize, forming a genuine topographic map of the input stimuli. Additionally, this seed can then be evolved further, providing a method to guide the self-organization of weights in a specific way, much as evolution likely guided the self-organizing trajectories of biological brains (i.e. part of the specific mechanisms for structure formation in biological brains is genetically determined and shaped by evolution [10]).

For example, Miikkulainen [13] modeled the lateral connectivity patterns of a self-organizing ANN, also called self-organizing map (SOM), in form of a "Mexican hat", in which the connections to the closest neurons are excitatory and inhibitory to neurons farther away. Such a connectivity pattern can easily also be expressed by a CPPN, by also including a Mexican hat function to the set of activation functions for HyperNEAT. Because the Mexican hat function peaks when its input is 0.0, inputting a difference between coordinates, e.g. Δx, achieves the highest value when the coordinates are the same and negative values for coordinates further apart than some threshold. The Mexican hat function employed in this paper is described as follows:

$$\gamma(x) = \begin{cases} 0.1, & |x| < 0.6 \\ -0.0125, & |x| > 0.6 \text{ and } |x| < 1.8 \\ 0.0, & \text{otherwise.} \end{cases} \qquad (2)$$

Figure 2 shows an example of such a *SOM-generating CPPN seed* together with the HyperNEAT substrate employed in this paper. The substrate contains a two-dimensional input layer (A) and an output layer (B). The two CPPNs are depictions of the same CPPN being queried to determine the weights and learning rates of two different substrate connections. The CPPN at the bottom receives as input the x and y coordinates of a node in A and a node in B and returns the weight of that connection from its AB output node and the corresponding Hebbian learning rate η of that connection. Note that only the afferent connection weights are adapted during the training of the network. The CPPN on top is the *SOM seed* (i.e. the lateral weights within the SOM in the first generation of evolution), which describes the lateral connectivity pattern of the hidden layer with short range excitatory and long range inhibitory connections. This setup in effect means that the CPPN encodes both the initial starting weights and the temporal dynamics of the plastic component of the network (i.e. the AB connections in Figure 2, yielding the ability to evolve the entire temporal dynamics and starting point for a self-organizing system.

3.1 ANN Activation and Weight Adaptation

The plastic ANN encoded by HyperNEAT is activated for each training input and, following Miikkulainen [13], allowed to settle until it reaches a stable state. The number of activations for the experiments in this paper is set to 15. After the initial network activation the afferent weights are updated based on the Hebbian learning rule (Equation 1) and then normalized to keep the sum of weights constant [2].

The next section introduces the experiments in this paper designed to test HyperNEAT's ability to produce topographic map-like structures that develop through a process of self-organization of connection weights. Additionally, a property common to most abstract implementations of SOMs [12] and more biologically plausible computational models [13, 22] is that the afferent connection weights are normally initialized to small random numbers (such an example is shown in Figure 3a). Therefore, an interesting unexplored question is whether this self-organization could be guided and potentially accelerated by indirectly encoding the initial afferent neural weight patterns. This intriguing possibility is now enabled by HyperNEAT's ability to describe large-scale ANN connectivity patterns as a function of neural geometry.

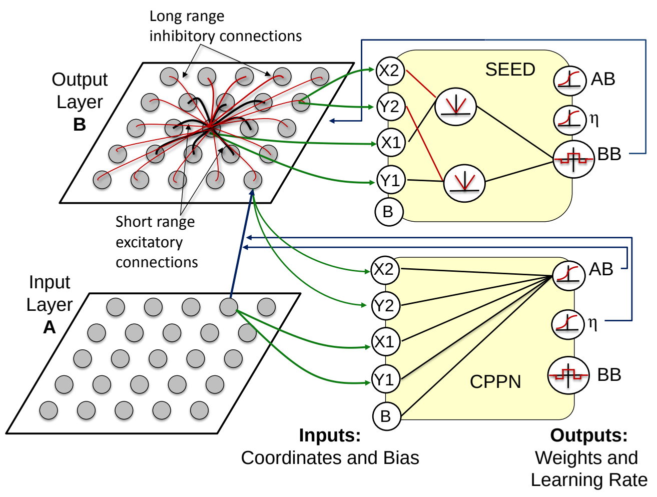

Long range inhibitory connections
Output Layer B
Short range excitatory connections
Input Layer A
SEED
X2
Y2
X1
Y1
B
AB
η
BB
CPPN
Inputs: Coordinates and Bias
Outputs: Weights and Learning Rate

Figure 2: HyperNEAT Substrate and Lateral Connectivity CPPN Seed. The substrate contains a two-dimensional input layer (A) and an output layer (B). The two CPPNs are depictions of the same CPPN being queried to determine the weights and learning rates of two different substrate connections. The CPPN at the bottom receives as input the x and y coordinates of a node in A and a node in B and returns the weight of that connection from its AB output node, as well as the corresponding Hebbian learning rate η of that connection from its η output. The CPPN at the top is being queried simply for the static weight of a connection entirely within the B layer, i.e. a BB connection.

4. EXPERIMENTS

The experiment in this paper tests the ability of the HyperNEAT approach to guide self-organization of connection weights in a line orientation task. In this task, differently-oriented lines are projected onto the ANN retina of 7×7 input neurons. The ANN output layer has the same number and layout of neurons. The substrate structure corresponds to the setup shown in Figure 2 (though the resolution shown in the figure is lower for clarity). At each learning step, one out of eight differently-orientated lines are selected in a *random order* and projected on the retina. This process is repeated for a total of 250 input presentations (following the network activation described in Section 3.1). Topographic maps of visual features such as line orientation are well studied in computational models of self-organization [21] and therefore provide a good first proof of concept for the ability of HyperNEAT to evolve a guided self-organizing process.

To asses the potential advantage of further evolving the initial SOM-generating CPPN seed shown in Figure 2, it is important first to test the self-organization capabilities of the initial seed itself. Figure 3 shows an example of randomly initialized weight patterns from the input neurons to the computational layer (before self-organization) and the weight pattern after training (i.e. 250 input presentations) for the seed-CPPN. The afferent connection weights self-organized such that neurons near each other respond to similar features of the sensory input (i.e. some neurons respond more strongly to particular line orientations than other neurons). The line orientation to which each neuron maximally responds is also shown before and after training with varying orientation increments (Figure 4). This figure shows how the self-organizing process responds to different orientation increments and consequently different number of training samples (eight compared to 30). The weight vectors

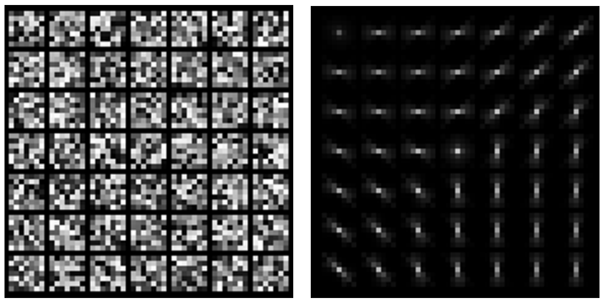

(a) Before Training (b) After Training

Figure 3: Weight Pattern Before and After Self-organization. This figure shows the weight patterns from each input node to all the 7×7 nodes in the output layer before (a) and (b) after self-organization . For example, the square to the top left depicts the connection weights from all 49 (7×7) input nodes to this particular output node.

form a topographic map that correctly reflects the distribution of input stimuli.

Now that the viability of the CPPN seed is established the question in this paper is whether evolution can provide a boost beyond the self-organizing functionality that the seed already provides. To test the hypothesis that HyperNEAT can guide the self-organizing process through indirectly encoded afferent connection weights, the evolved CPPNs are tested on their ability to create ANNs that can self-organize into a *particular* weight configuration *independently of the order of presented input samples*. It is important to note that the aim of this study is not to suggest that the particular final configuration is inherently important in its own right, but rather to make the point that evolution can shape the self-organizing process to push it in a particular direction. In effect, choosing an explicit final target configuration in this experiment makes it easier to show definitively and explicitly that such shaping is possible. In the future this principle can then be applied to more subtle and less explicit endpoints for self-organizing processes. The fitness of a network for this purpose is determined based on:

$$\Phi = 1.0 - \frac{1}{N}\sum_{i=0}^{N}|w_i - c(t)_i|, \qquad (3)$$

where **w** is the weight vector produced by the CPPN seed (as seen in Figure 3b), **c** is the current weight vector at time t and N are the total number of connections in the ANN. This function is evaluated every ten training steps and the resulting values are summed to determine the fitness for one trial. To encourage solutions that match as closely as possible to the final target pattern, a value of 5.0 is added to the fitness each time step that the pattern is within a margin of 0.1% to the target pattern. This fitness function encourages self-organization of connectivity weights into the final configuration as fast as possible. To encourage robust solutions that do not depend on a particular order of input samples, each network is evaluated on five independent trials and the final fitness is the average performance across these trials.

Five different approaches are compared to determine the role of different initial afferent weight patterns and indirectly encoded learning rates on the self-organizing progress:

- In the **Seeded HyperNEAT** approach the initial weights from the input to the output layer of the ANN are determined by the HyperNEAT approach described in Section 2.3 in which

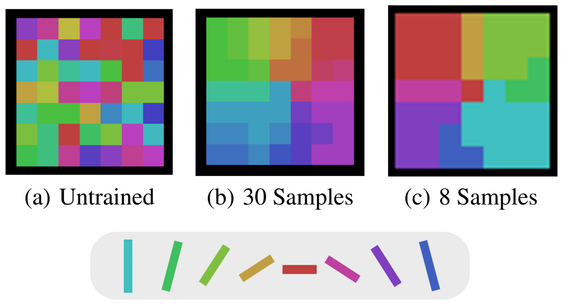


Figure 4: Neuron Orientation Sensitivity. Line orientations (d) are shown to which the neurons in the output layer maximally respond before the self-organization of afferent connection (a), and after the training with different orientation increments and therefore different-sized training samples (30 compared to eight) in (b) and (c). The corresponding self-organized afferent weight patterns for (c) are shown in Figure 3b. The main results is that the weight vectors form a topographic map that correctly reflects the distribution of input stimuli.

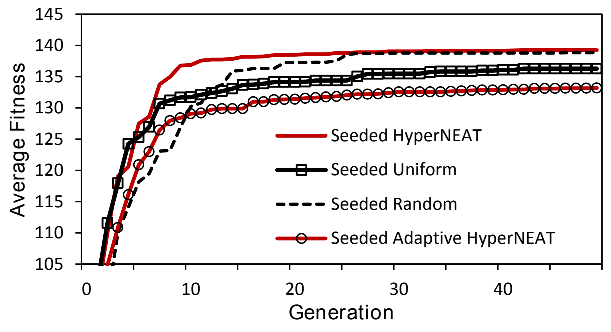


Figure 5: Training Performance. The average best fitness over generations is shown for the different HyperNEAT variants, which are averaged over 30 runs. The main result is that Seeded HyperNEAT and Seeded Random reach a significantly higher average fitness than all other approaches.

evolution is seeded with a lateral connectivity pattern (Figure 2). The Hebbian learning rate for the weights from the input to the outputs nodes are uniformly set to 1.0.

- The **Seeded Uniform** approach follows the Seeeded HyperNEAT approach but the weights from the input to the output layer are all set to an intermediate inform value of 0.5 instead of being determined by HyperNEAT.
- **Seeded Random** tests the influence of starting with random weights in the range [0, 1] from the inputs to the output nodes. This approach reflects the conventional approach initializing the afferent weights in SOMs [12, 13].
- The **Seeded Adaptive HyperNEAT** approach follows the Seeded HyperNEAT approach but in addition to the weights the learning rates are also generated by HyperNEAT (Section 2.3). This approach tests the hypothesis that distributing the learning rules in a pattern could provide an additional advantage.
- In the **Unseeded HyperNEAT** approach in this paper evolution is not seeded (i.e. the initial CPPNs do not contain any hidden nodes). This variant is designed to test the influence of seeding HyperNEAT with a SOM-like connectivity pattern. It is important to note that HyperNEAT can theoretically discover a CPPN similar to the seed CPPN by itself (e.g. a Mexican Hat function can be added during node addition) that would encode a lateral connectivity pattern necessary for self-organization.

4.1 Experimental Parameters

The size of each HyperNEAT population was 50 with 10% elitism and a termination criterion of 50 generations. Sexual offspring (75%) did not undergo mutation. Asexual offspring (25%) had 0.6 probability of weight mutation, 0.06 chance of link addition, and 0.005 chance of node addition. The available CPPN activation functions were sigmoid, Gaussian, absolute value, cosine, Mexican hat, and sine, all with equal probability of being added.

5. RESULTS

Each of the HyperNEAT approaches is evaluated over 30 independent runs consisting of 50 generations of evolution. Figure 5 shows the training performance over generations for the different HyperNEAT variants. While the seeded HyperNEAT and Seeded Random approaches reach a similar average maximum fitness of 139.24 ($\sigma = 0.49$) and 138.82 ($\sigma = 0.77$), respectively, the Uniform setup and seeded Adaptive HyperNEAT reach significantly lower average maximum fitness values ($p < 0.05$ according to the Student's t-test). The Unseeded HyperNEAT fails to achieve high average performance and only reaches a fitness of 24.16 ($\sigma = 0.01$) (data not shown).

It is important to note that while the differences in average fitness between the investigated HyperNEAT methods appear relatively small, the number of generations it took them on average to find a solution paints a clearer picture (Figure 6). A network counts as a solution if the error between its final weight pattern (after self-organization) and the training weight pattern is less than 0.01%. Seeded HyperNEAT finds a solution in 7.5 generations on average ($\sigma = 3.01$), which is significantly faster ($p < 0.05$) than any of the other approaches (Figure 6). In fact, the unseeded version fails to find a solution in all 30 runs, confirming the necessity of the initial CPPN seed and the deceptiveness of rediscovering the necessary lateral connectivity pattern through evolution alone. It is interesting to note that Seeded Adaptive HyperNEAT performs significantly worse than Seeded HyperNEAT alone ($p < 0.001$), suggesting that a uniform learning rate is already optimal for this particular task.

Because one motivation for this paper is to investigate the robustness of evolved networks in self-organizing into a particular target configuration, a generalization test was devised to measure how well an evolved solution would perform under different random orders of training samples. The generalization test consists of 30 trials and measures whether solutions get within a margin of 1% and 0.1% to the final weight target configuration shown in Figure 3b. The score on the generalization test reflects the probability that a solution evolved by a particular HyperNEAT variant self-organizes correctly (Figure 7). Seeded HyperNEAT, Seeded Adaptive HyperNEAT and the Uniform approach do not perform significantly different and are able to get within a margin of 1% to the target pattern in 50%, 46% and 56% of the time, respectively. However, all three methods perform significantly better than the Random approach ($p < 0.05$), which only solves the task 35% of the time. The

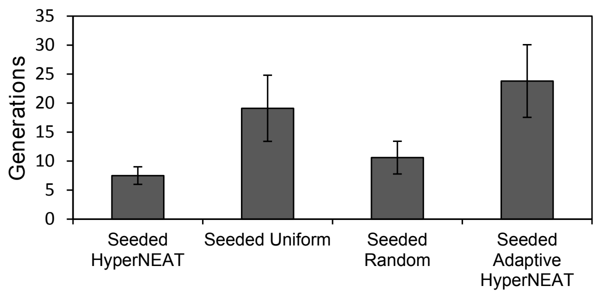


Figure 6: Average Generations to Solution (lower is better). The average number of generations over 30 runs that it took the different HyperNEAT variants to reach the final target configuration (Figure 3b) is shown. The unseeded version fails to find a solution in all 30 runs (data not shown). The main result is that Seeded HyperNEAT finds a solution significantly faster than all the other approaches.

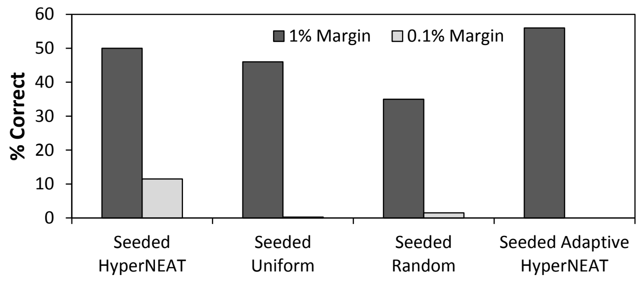


Figure 7: Generalization Test Results. This chart depicts how often the final champions from each of the 30 runs are able to reach the target configuration averaged over 30 independent trials (with different orders of training examples). Higher values are better.

final performance decreases significantly for all approaches if solutions are required get within a margin of 0.1% to the target pattern ($p < 0.001$). Seeded HyperNEAT only passes this stricter generalization test 15% of the times, which is however significantly higher than for the Uniform and Random approach ($p < 0.001$). Interestingly, these results indicate that the Random approach is less reliable in producing the final target configuration and that determining the initial afferent weights through HyperNEAT allows it to self-organize into the target configuration with a greater accuracy.

5.1 Self-Organization Analysis

Because both the Seeded HyperNEAT and the Seeded Uniform approach are able to reach the correct target pattern – within a margin of 1% error – about the same number of times how can their significant difference in final average performance (Figure 5) be explained? A closer look at typical learning curves for both approaches during self-organization of the map (Figure 8) suggest that while the seeded HyperNEAT and Uniform approach ultimately reach a similar accuracy, HyperNEAT reaches it more quickly (and therefore reaches a higher fitness). One way to analyze the reason behind these different progressions is to observe the development of the orientation sensitivity of the output neurons. From Figure 8 it becomes clear that the Seeded HyperNEAT approach is able to establish the correct orientation pattern more quickly than the Uniform approach. Initial uniform weights seem to hinder the self-organizing progress, which could potentially be explained by the increased local competition between neurons (i.e. by starting

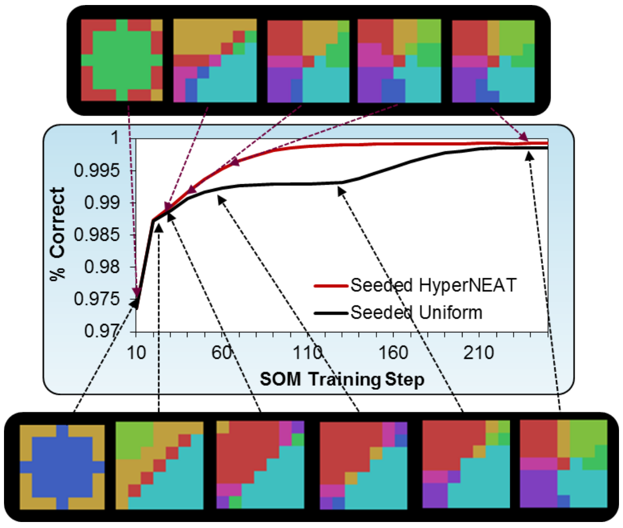


Figure 8: Development of Orientation Preferences. The development of the orientation preferences of the output neurons over time is shown for the Seeded HyperNEAT approach (top) and the Uniform approach (bottom). The initial non-uniform weights (Figure 9) allow the Seeded HyperNEAT approach to self-organize into the target configuration in fewer training steps.

Table 1: Training and Generalization Results. The Seeded HyperNEAT approach often performs significantly better than the other HyperNEAT variants and never significantly worse.

Method	Fitness	Generations	1% Margin
HyperNEAT	**139.2** ($\sigma = 0.49$)	**7.5** ($\sigma = 3.01$)	50%
Random	138.8 ($\sigma = 0.77$)	10.6 ($\sigma = 5.9$)	34%
Uniform	136.2 ($\sigma = 1.70$)	19.1 ($\sigma = 11.4$)	46%
Adaptive	133.1 ($\sigma = 1.82$)	23.8 ($\sigma = 12.5$)	**56%**
Unseeded	24.1 ($\sigma = 0.03$)	-	-

with uniform weights each neuron will initially respond similarly to the same input stimuli). The Seeded HyperNEAT approach on the other hand creates a smooth initial weight pattern (Figure 9) that likely reduces the initial competition among neurons (similarly to a random initialization).

The main result is that while HyperNEAT and the Random setup reach a similar final performance, HyperNEAT is able to reproduce the final target weight pattern significantly more often. Additionally, the Uniform setup reaches an average lower maximum fitness because the self-organization process is delayed when all the weights are initially the same. Overall then the Seeded HyperNEAT approach combines the speed of the Random setup with the reproducibility of the Uniform setup. The results are summarized in Table 1.

6. DISCUSSION AND FUTURE WORK

While the domain presented in this paper is simple, it helps to demonstrate that it is indeed possible to seed HyperNEAT with lateral connectivity that allows the network to self-organize into a genuine topographic map of the input space. Also importantly, self-organization can be guided and accelerated by indirectly encoding the initial afferent neural weight patterns with the HyperNEAT method. While this study does not suggest that a particular final configuration of weights is inherently important in its own

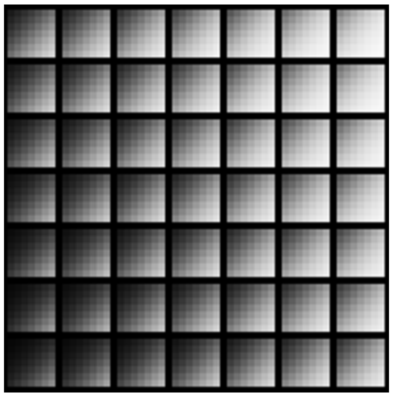


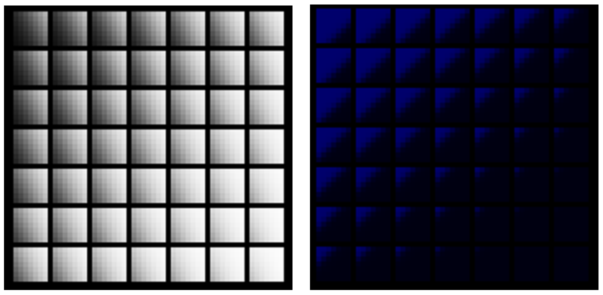
Figure 9: Evolved Initial Weight Pattern for Seeded HyperNEAT. The initial weight pattern from the input nodes to the output nodes is shown that was determined by the Seeded HyperNEAT approach. This pattern provides the self-organizing process with a bias towards a particular target configuration and accelerates the self-organizing process compared to a uniform initialization of connection weights.

right, it nevertheless establishes that such guided self-organization is possible. In effect evolving towards a final target facilitated the analysis of the role of different initial afferent weight patterns and indirectly encoded learning rates on the self-organizing progress.

Interestingly, the poor performance of the unseeded version (Figure 6) suggests that the path through the search space to encoding such lateral connectivity patterns necessary for self-organizing dynamics to emerge is inherently deceptive. Similar results indicating that discovering particular kinds of structures is deceptive were also observed by Verbancsics and Stanley [32] and Risi and Stanley [17] on seeding HyperNEAT with the concept of locality. This growing body of evidence suggests that such deception might be common when geometric principles such as lateral connectivity or locality are essential to achieving the desired behavior but fitness does not reward the intermediate stepping stones that lead to that final objective. Thus manually constructing an appropriate starting seed may prove an important tool to avert deception in many domains. In the future it might also be possible to automatically *compile* a seed that directly satisfies some user-imposed properties [15].

Furthermore, while prior results in a T-Maze learning task suggest the benefit of indirectly encoded learning rules [16], Seeded Adaptive HyperNEAT's performance in this domain is less consistent. It reaches a significantly lower maximum fitness than Seeded HyperNEAT (Figure 5), which could suggest that a uniform learning rate is already optimal for this particular task. However, it performs better on the generalization test (1% margin) than Seeded HyperNEAT (though not significantly), and interestingly, the three most general solutions that solve the task in 80% of the time are all evolved by the Seeded Adaptive HyperNEAT approach. Figure 10 shows the initial weight and learning rate pattern from the most general evolved Adaptive HyperNEAT solution. Although a more detailed analysis is warranted in the future, this result indicates that (1) both learning rate *and* weight pattern can work together to robustly guide the self-organizational process and (2) finding such a solution is harder because the right learning rate pattern also has to be discovered.

In the future it might be possible to combine a SOM-generating seed with a recent HyperNEAT extension called adaptive evolvable-substrate HyperNEAT [17, 18]. Adaptive ES-HyperNEAT can automatically determine the placement, density and plasticity of neurons in the HyperNEAT substrate. Thus seeding the adaptive ES-HyperNEAT approach with the right lateral connectivity could in principle allow structures reminiscent of deep learning architectures [11] to evolve, in which the internal layers self-organize into topographic maps that in each successive layer form higher and higher abstract representations of the input space.



(a) Weight Pattern (b) Learning Rate Pattern

Figure 10: Evolved Initial Weight and Learning Rate Pattern for Seeded Adaptive HyperNEAT. The initial weight and learning rate pattern for the afferent connections are shown that were determined by the Seeded Adaptive HyperNEAT approach. The very robust performance of this evolved solution on the generalization test suggests that both weights *and* learning rate patterns can work together to bias the self-organizing process.

7. CONCLUSION

This paper introduced the idea that the indirect HyperNEAT encoding can be seeded with a lateral inhibitory connectivity pattern, which then allows the weights from the inputs to the hidden layer to self-organize to form a genuine topographic map of the input space. The benefit of this new approach is that the initial seed can be evolved further to accelerate the self-organizing process and to bias it towards a specific target configuration. An interesting implication is that enabling HyperNEAT to capture this key feature of natural systems might now allow us to evolve architectures resembling deep learning networks that also rely on unsupervised learning processes.

References

[1] J. C. Bongard. Evolving modular genetic regulatory networks. In *Proc. of the 2002 Congress on Evolutionary Computation*, 2002.

[2] Y. Choe, J. Sirosh, and R. Miikkulainen. Laterally interconnected self-organizing maps in hand-written digit recognition. *Advances in neural information processing systems*, pages 736–742, 1996.

[3] J. Clune, B. E. Beckmann, C. Ofria, and R. T. Pennock. Evolving coordinated quadruped gaits with the HyperNEAT generative encoding. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC-2009) Special Session on Evolutionary Robotics*, Piscataway, NJ, USA, 2009. IEEE Press.

[4] J. Clune, K. O. Stanley, R. T. Pennock, and C. Ofria. On the performance of indirect encoding across the continuum of regularity. *Evolutionary Computation, IEEE Transactions on*, 15(3):346–367, 2011.

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

[6] D. Floreano and J. Urzelai. Evolutionary robots with online self-organization and behavioral fitness. *Neural Networks*, 13: 431–443, 2000.

[7] D. Floreano, P. Dürr, and C. Mattiussi. Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1(1):47–62, 2008.

[8] J. Gauci and K. O. Stanley. A case study on the critical role of geometric regularity in machine learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI-2008)*, Menlo Park, CA, 2008. AAAI Press.

[9] J. Gauci and K. O. Stanley. Autonomous evolution of topographic regularities in artificial neural networks. *Neural Computation*, 22:1860–1898, 2010.

[10] C. S. Goodman and C. J. Shatz. Developmental mechanisms that generate precise patterns of neuronal connectivity. *Cell*, 72:77–98, 1993.

[11] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7): 1527–1554, 2006.

[12] T. Kohonen. *Self-organizing maps*, volume 30. Springer, 2001.

[13] R. Miikkulainen. Self-organizing process based on lateral inhibition and synaptic resource redistribution. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 415–420, 1991.

[14] Y. Niv, D. Joel, I. Meilijson, and E. Ruppin. Evolution of reinforcement learning in uncertain environments: A simple explanation for complex foraging behaviors. *Adaptive Behavior*, 10(1):5–24, 2002. ISSN 1059-7123.

[15] S. Risi. A Compiler for CPPNs: Transforming Phenotypic Descriptions Into Genotypic Representations. In *Proceedings of the 2013 AAAI Fall Symposium on How Should Intelligence be Abstracted in AI Research*, 2013.

[16] S. Risi and K. O. Stanley. Indirectly encoding neural plasticity as a pattern of local rules. In S. Doncieux, B. Girard, A. Guillot, J. Hallam, J.-A. Meyer, and J.-B. Mouret, editors, *From Animals to Animats 11*, volume 6226 of *Lecture Notes in Computer Science*, pages 533–543. Springer Berlin / Heidelberg, 2010.

[17] S. Risi and K. O. Stanley. An enhanced hypercube-based encoding for evolving the placement, density, and connectivity of neurons. *Artificial Life*, 18(4):331–363, 2012.

[18] S. Risi and K. O. Stanley. A unified approach to evolving plasticity and neural geometry. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*. IEEE, 2012.

[19] S. Risi, C. E. Hughes, and K. O. Stanley. Evolving plastic neural networks with novelty search. *Adaptive Behavior - Animals, Animats, Software Agents, Robots, Adaptive Systems*, 18:470–491, December 2010. ISSN 1059-7123.

[20] J. Secretan, N. Beato, D. B. D'Ambrosio, A. Rodriguez, A. Campbell, and K. O. Stanley. Picbreeder: Evolving pictures collaboratively online. In *CHI '08: Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1759–1768, New York, NY, USA, 2008. ACM.

[21] J. Sirosh and R. Miikkulainen. Cooperative self-organization of afferent and lateral connections in cortical maps. *Biological Cybernetics*, 71(1):65–78, 1994.

[22] J. Sirosh and R. Miikkulainen. Topographic receptive fields and patterned lateral interaction in a self-organizing model of the primary visual cortex. *Neural Computation*, 9(3):577–594, 1997.

[23] A. Soltoggio, J. A. Bullinaria, C. Mattiussi, P. Dürr, and D. Floreano. Evolutionary advantages of neuromodulated plasticity in dynamic, reward-based scenarios. In *Artificial Life XI*, pages 569–576, Cambridge, MA, 2008. MIT Press.

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

[25] K. O. Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines Special Issue on Developmental Systems*, 8(2):131–162, 2007.

[26] K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10:99–127, 2002.

[27] K. O. Stanley and R. Miikkulainen. A taxonomy for artificial embryogeny. *Artificial Life*, 9(2):93–130, 2003.

[28] K. O. Stanley and R. Miikkulainen. Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, 21:63–100, 2004.

[29] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci. A hypercube-based indirect encoding for evolving large-scale neural networks. *Artificial Life*, 15(2):185–212, 2009.

[30] N. Swindale. The development of topography in the visual cortex: A review of models. *Network: Computation in neural systems*, 7(2):161–247, 1996.

[31] J.-P. Thivierge and G. F. Marcus. The topographic brain: from neural connectivity to cognition. *Trends in neurosciences*, 30 (6):251–259, 2007.

[32] P. Verbancsics and K. O. Stanley. Constraining Connectivity to Encourage Modularity in HyperNEAT. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2011)*, New York, NY, 2011. ACM.

[33] X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9), September 1999.