

# A Unified Approach to Evolving Plasticity and Neural Geometry

In: *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2012)*. Piscataway, NJ: IEEE  
Winner of the Best Student Paper Award

Sebastian Risi  
Department of EECS  
University of Central Florida  
Orlando, FL 32816, USA  
Email: sebastian.risi@gmail.com

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

**Abstract**—An ambitious long-term goal for neuroevolution, which studies how artificial evolutionary processes can be driven to produce brain-like structures, is to evolve neurocontrollers with a high density of neurons and connections that can adapt and learn from past experience. Yet while neuroevolution has produced successful results in a variety of domains, the scale of natural brains remains far beyond reach. This paper unifies a set of advanced neuroevolution techniques into a new method called *adaptive evolvable-substrate HyperNEAT*, which is a step toward more biologically-plausible artificial neural networks (ANNs). The combined approach is able to fully determine the geometry, density, and plasticity of an evolving neuromodulated ANN. These complementary capabilities are demonstrated in a maze-learning task based on similar experiments with animals. The most interesting aspect of this investigation is that the emergent neural structures are beginning to acquire more natural properties, which means that neuroevolution can begin to pose new problems and answer deeper questions about how brains evolved that are ultimately relevant to the field of AI as a whole.

## I. INTRODUCTION

The human brain is the most complex system known to exist [17, 36]. It is composed of an astronomically high number of neurons and connections organized in precise, intricate motifs that repeat throughout it, often with variation on a theme, such as cortical columns [26]. Additionally, the brain is not a static entity but instead changing throughout life, enabling us to learn and modify our behavior based on past experience. These characteristics have so far eluded attempts to create artificial neural networks (ANNs) with the same capabilities.

One methodology that has shown promise in autonomously generating ANNs for complex control problems is *neuroevolution*, i.e. evolving ANNs through evolutionary algorithms (EAs) [12, 20, 31, 35]. To this end, as such algorithms are asked to evolve increasingly large and complex structures, interest has increased in recent years in indirect neural network encodings, wherein the description of the solution is compressed such that information can be reused [2, 4, 14, 16, 27, 29]. Such compression allows the final solution to contain more components than its description. Nevertheless, neuroevolution has historically produced networks with orders of magnitude fewer neurons and significantly less organization and regularity than natural brains [30, 35].

While past approaches to neuroevolution generally concentrated on deciding which node is connected to which (i.e.

neural *topology*) [12, 30, 35], the Hypercube-based NeuroEvolution of Augmenting Topologies (HyperNEAT) method [8, 13, 32] provided a new perspective on evolving ANNs by showing that the pattern of weights across the connectivity of an ANN can be generated as a function of its geometry. HyperNEAT employs an indirect encoding called compositional pattern producing networks (CPPNs) [27], which can compactly encode patterns with regularities such as symmetry, repetition, and repetition with variation.

HyperNEAT exposed the fact that neuroevolution benefits from neurons that exist at *locations* within the space of the brain and that by placing neurons at locations, evolution can exploit *topography* (as opposed to just topology), which makes it possible to correlate the geometry of sensors with the geometry of the brain. In other words, two ANNs with the same topology (i.e. the same connectivity pattern) can have different topographies (i.e. nodes existing at different locations in space). While lacking in many ANNs, such geometry is a critical facet of natural brains that is responsible for e.g. topographic maps and modular organization across space [26]. This insight allowed large ANNs with regularities in connectivity to evolve for high-dimensional problems [7, 13, 14, 32]. Yet a significant limitation is that the *positions* of the nodes connected through this approach must be decided a priori by the user. In other words, in the original HyperNEAT, the user must literally place nodes at locations within a two-dimensional or three-dimensional space called the substrate.

In a step towards more brain-like neurocontrollers, the recently introduced *evolvable-substrate HyperNEAT* (ES-HyperNEAT) approach [22] showed that the placement and density of the hidden nodes can in fact be determined solely based on implicit information in an infinite-resolution pattern of weights generated by HyperNEAT. The novel insight was that a representation that encodes the pattern of connectivity across a network (such as in HyperNEAT) automatically contains *implicit* clues on where the nodes should be placed to best capture the information stored in the connectivity pattern. In other words, there is no need for any new information or any new representational structure beyond the very same CPPN that already encodes network connectivity in the original HyperNEAT. This approach not only can evolve the location of every neuron in the network, but also can

represent regions of varying density, which means resolution can increase holistically over evolution.

Another common contrast with real brains is that the synaptic connections in ANNs produced by EAs are static in many implementations (i.e. they do not change their strength during the lifetime of the network). While some tasks do not require the network to change its behavior, many domains would benefit from *online adaptation*. In other words, whereas evolution produces phylogenetic adaptation, learning gives the individual the possibility to react much faster to environmental changes by modifying its behavior during its lifetime. For example, a robot that is physically damaged should be able to adapt to its new circumstances without the need to re-evolve its neurocontroller.

One way that agents controlled by ANNs can evolve the ability to adapt over their lifetime is by encoding *local learning rules* in the genome that determine how their synaptic connection strengths should change in response to changing activation levels in the neurons they connect [11, 18, 25]. This approach resembles the way organisms in nature, which possess plastic nervous systems, cope with changing and unpredictable environments. Although demonstrations of this approach have suggested the promise of evolving adaptive ANNs, a significant problem is that local learning rules for every connection in the network must be discovered separately. The recently introduced *adaptive HyperNEAT* [21] addressed this problem and showed that 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 neural learning rules*. The idea that learning rules can be distributed in a geometric pattern is new to neuroevolution but reflects the intuition that synaptic plasticity in biological brains is not encoded in DNA separately for every synapse in the brain.

While ES-HyperNEAT and adaptive HyperNEAT have shown promise, it is the combination of the two methods that should allow the evolution of complex regular plastic ANNs, which is a major goal for neuroevolution. *Adaptive ES-HyperNEAT*, introduced here, is able to fully determine the geometry, density, and plasticity of an evolving ANN. In this way, it unifies two insights that have recently extended the reach of HyperNEAT: First, the placement and density of neurons throughout the geometry of the network should reflect the complexity of the underlying functionality of its respective parts [22]. Second, neural plasticity, which allows adaptation, should be encoded as a function of neural geometry [21]. Also importantly, the approach has the potential to create networks from several dozen nodes up to several million, which will be necessary in the future to evolve more intelligent systems.

The main conclusion is that adaptive ES-HyperNEAT takes a step towards more biologically-plausible ANNs and expands the scope of neural structures that evolution can discover, as demonstrated by experiments in a T-Maze learning task in this paper. In this scenario, the reward location is a variable factor in the environment that the agent must learn to exploit. The most important aspect of this domain is that it is not the traditional discrete grid-world T-Maze that is popular in this

area [21, 25]; rather, the more realistic domain presented here requires the robot to develop both collision avoidance *and* the ability to learn during its lifetime in a continuous world [3, 9]. Both are afforded seamlessly by adaptive ES-HyperNEAT’s ability to evolve both neural density and plasticity together.

## II. BACKGROUND

This section reviews the sequence of cumulative abstractions of processes in natural evolution from the last decade that are combined in this paper.

### A. Neuroevolution of Augmenting Topologies (NEAT)

The approaches introduced in this paper are extensions of the original NEAT algorithm that evolves increasing large ANNs. It starts with a population of simple networks and then *adds complexity* over generations by adding new nodes and connections through mutations. By evolving ANNs 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 large brain-like networks. For a complete overview of NEAT see Stanley and Miikkulainen [28].

### B. Hypercube-based NEAT

In direct encodings like NEAT, each part of the solution’s representation (i.e. each *gene*) maps to a single piece of structure in the final solution [12]. 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 *patterns* of parameters, rather than representing each parameter individually [4, 14, 27, 29]. HyperNEAT, reviewed here, is an indirect encoding extension of NEAT that is proven in a number of challenging domains that require discovering regularities [6, 13, 14, 32, 34]. For a full description see Stanley et al. [32] and Gauci and Stanley [14].

In HyperNEAT, NEAT is altered to evolve an indirect encoding called *compositional pattern producing networks* (CPPNs [27]) *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 “DNA” encoding is *itself* a network, which means that CPPNs can be evolved by NEAT. Specifically, a CPPN is a *composition 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.

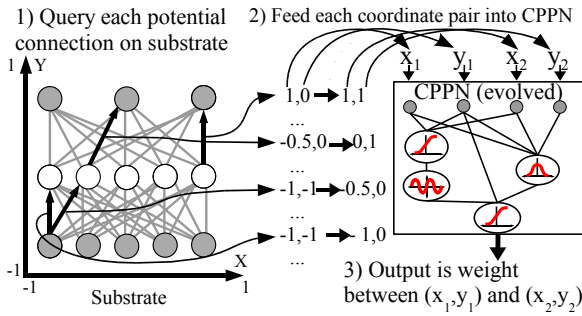


Fig. 1. **HyperNEAT Geometric Connectivity Pattern Interpretation.** A collection of nodes, called the *substrate*, is assigned coordinates that range from  $-1$  to  $1$  in both dimensions. (1) Every potential connection in the substrate is queried by the CPPN to determine its presence and weight; 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 by NEAT) 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.

In effect, the indirect CPPN encoding can compactly encode patterns with regularities such as symmetry, repetition, and repetition with variation [27]. Thus as the CPPN increases in complexity through NEAT, it encodes increasingly complex amalgamations of regularities and symmetries that are projected across the connectivity of an ANN [13, 14].

Unlike in many common ANN 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 (recall) is also a network.

Each queried point in the substrate is a node in an ANN. In the original HyperNEAT approach, 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 [6, 32]. 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 (e.g. adjacency, or symmetry) of a problem that are invisible to traditional encodings.

Because it encodes the network as a function of its geometry, HyperNEAT made it possible to evolve ANNs with large geometrically-situated output fields [34] and millions of connections [32], thereby advancing neuroevolution a step closer to nature. Yet a problem with the original HyperNEAT was that the experimenter is left to decide how many hidden nodes there should be and at what geometric locations to place them. The recent solution to this problem is the *evolvable substrate*, which is reviewed in the next section.

### C. Evolvable-Substrate HyperNEAT

While past approaches to neuroevolution generally concentrated on deciding which node is connected to which [12, 30, 35], HyperNEAT exposed the fact that neuroevolution benefits from neurons that exist at *locations* within the space of the brain. By placing neurons at locations, evolution can exploit *topography*, which makes it possible to correlate the geometry of sensors with the geometry of the brain. While lacking in many ANNs, such geometry is a critical facet of natural brains that is responsible for e.g. topographic maps and modular organization across space [26].

Thus when Risi et al. [24] introduced a method to automatically deduce node geometry and density *from CPPNs* instead of requiring a priori placement (as in original HyperNEAT), it significantly expanded the scope of neural structures that evolution can discover. This approach, called *evolvable-substrate HyperNEAT*, evolves not only the location of every neuron in the brain, but also can represent regions of varying density, which means resolution can increase holistically over evolution. This advance was enabled by the insight that both connectivity *and* hidden node placement can be automatically determined by information already inherent in the pattern encoded by the CPPN.

The key idea is to *search* the pattern within the four-dimensional hypercube encoded by the CPPN for areas of high information. The fundamental insight is that the geometric location of nodes in an ANN is ultimately a signification of where information is stored within the hypercube. That is, areas of uniform weight (which map to uniform connection weights in the substrate) ultimately encode very little information and therefore little of functional value. Thus connections (and hence the node locations that they connect) can be chosen to be expressed based on the variance within their region of the CPPN-encoded function in the hypercube.

The implementation of this idea is based on a quadtree-like decomposition of weight-space that searches for areas of high variance. Interestingly, such a decomposition will continue to drill down to higher and higher resolutions as long as variance

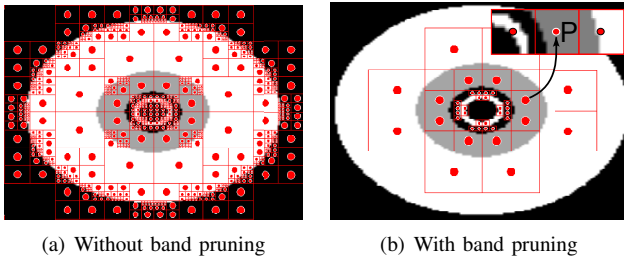


Fig. 2. **Example point selection in two dimensions.** Chosen points without band pruning are shown in (a). Points that still remain after band pruning (e.g. point  $P$ , whose neighbors at the same resolution have different CPPN activation levels) are shown in (b). The resulting point distribution reflects the information inherent in the image.

is detected. In this way, the density of nodes is automatically determined and effectively unbounded. Thus substrates of unbounded density can be evolved and determined without any additional representation beyond the original CPPN in HyperNEAT. This result is elegant because it corresponds to a philosophical view of neural organization that the locations of useful information within the hypercube are where the nodes should be placed. That way, the *size* of the brain is roughly correlated to its *complexity*. The full ES-HyperNEAT approach is detailed in Risi and Stanley [22].

Figure 2a shows an example of the points chosen by the quadtree algorithm for a two-dimensional pattern, which resembles typical quadtree image decompositions [33]. To understand how this pattern (which would be encoded by a CPPN) relates to HyperNEAT, think of it as an infinite set of candidate connection weights from which ES-HyperNEAT must choose those actually to include in the network. The variance is high at the borders of the circles, which results in a high density of expressed points at those locations. However, for the purpose of identifying connections to include in a neural topography, the raw pattern output by the quadtree algorithm can be improved further. If we think of the pattern output by the CPPN as a kind of *language* for specifying the locations of expressed connections, then it makes sense additionally to prune the points around borders so that it is easy for the CPPN to encode points definitively within one region or another.

Thus a more parsimonious “language” for describing density patterns would ignore the edges and focus on the inner region of *bands*, which are points that are enclosed by at least two neighbors on opposite sides (e.g. left and right) with different CPPN activation levels (figure 2b). Furthermore, narrower bands can be interpreted as requests for more points, giving the CPPN an explicit mechanism for affecting density.

Thus, to facilitate banding, a pruning stage is added that removes points that are not in a band. Membership in a band for a square with center  $(x, y)$  and width  $\omega$  is determined by band level  $b = \max(\min(d_{\text{top}}, d_{\text{bottom}}), \min(d_{\text{left}}, d_{\text{right}}))$ , where  $d_{\text{left}}$  is the difference in CPPN activation levels between the point  $(x, y)$  and its left neighbor at  $(x - \omega, y)$ . The other values,  $d_{\text{right}}$ ,  $d_{\text{bottom}}$ , and  $d_{\text{top}}$ , are calculated accordingly. If the band level  $b$  is below a given threshold  $b_t$  then the

corresponding point is not expressed. Figure 2b shows the resulting point selections with band pruning. This approach also naturally enables the CPPN to increase the density of points chosen by creating more bands or making them thinner.

While the approach so far identifies which connections to express from within a field of potential connections, the problem still remains of ensuring that the inputs and outputs that the user places on the substrate ultimately connect into the network. ES-HyperNEAT thus completes the network by iteratively discovering the placement of the hidden neurons starting from the inputs of the ANN [22]. This approach focuses the search within the hypercube on discovering *functional networks* in which every hidden node contributes to the ANN output and receives information (at least indirectly) from at least one input neuron, while ignoring parts of the hypercube that are disconnected.

The idea behind this completion algorithm is depicted in figure 3. Instead of searching directly in the four-dimensional hypercube space (recall that it takes four dimensions to represent a two-dimensional connectivity pattern), the algorithm analyzes a sequence of two-dimensional cross-sections of the hypercube, one at a time, to discover which connections to include in the ANN.

For example, given an input node at  $(0, -1)$  the quadtree point-choosing approach is applied only to the two-dimensional *outgoing* connectivity patterns from that single node (figure 3a) described by the function  $CPPN(0, -1, x, y)$  with  $x$  and  $y$  ranging from -1 to 1. This process can be iteratively applied to the discovered hidden nodes until a user-defined maximum iteration level is reached or no more information is discovered in the hypercube (figure 3b). Note that an iteration level greater than zero enables the algorithm also to create recurrent ANNs. To tie the network into the outputs, the approach chooses connections based on each output’s *incoming* connectivity patterns (figure 3c). Once all hidden neurons are discovered, only those are kept that have a path to an input *and* output neuron (figure 3d).

Because ES-HyperNEAT can automatically deduce node geometry and density instead of requiring a priori placement (as in original HyperNEAT), it significantly expands the scope of neural structures that evolution can discover. The approach does not only evolve the location of every neuron in the brain, but also can represent regions of varying density, which means resolution can increase holistically over evolution. The main insight is that the connectivity and hidden node placement can be automatically determined by information already inherent in the pattern encoded by the CPPN. In this way, the density of nodes is effectively unbounded. Thus substrates of unbounded density can be evolved without any additional representation beyond the original CPPN in HyperNEAT.

#### D. Evolving Plastic ANNs with HyperNEAT

Unlike traditional static ANNs in neuroevolution whose weights do not change during their lifetime, plastic ANNs can *learn during their lifetime* by changing their internal synaptic connection strengths following a Hebbian learning rule that

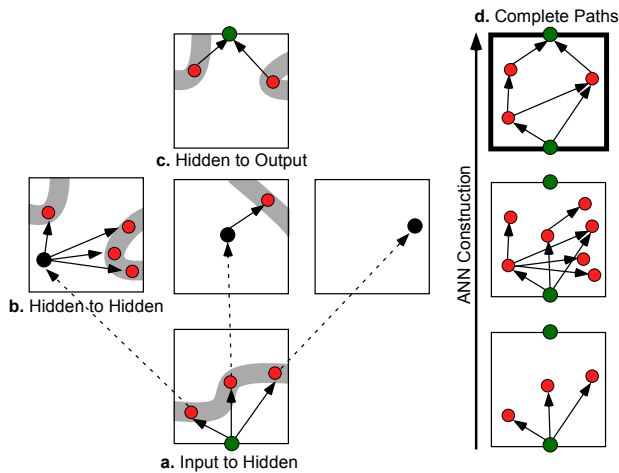


Fig. 3. **Iterated Network Completion.** The algorithm starts by iteratively discovering the placement of the hidden neurons from the inputs (a) and then ties the network into the outputs (c). The two-dimensional motif in (a) represent *outgoing* connectivity patterns from a single input node whereas the motif in (c) represent *incoming* connectivity pattern for a single output node. The target nodes discovered (through the quadtree algorithm) are those that reside within bands in the hypercube. In this way regions of high variance are sought only in the two-dimensional cross-section of the hypercube containing the source or target node. The algorithm can be iteratively applied to the discovered hidden nodes (b). Only those nodes are kept that have a path to an input *and* output neuron (d). That way, the search through the hypercube is restricted to functional ANN topologies.

modifies synaptic weights based on pre- and postsynaptic neuron activity. The generalized Hebbian plasticity rule [18] takes the form:  $\Delta w_{ji} = \eta \cdot [A o_j o_i + B o_j + C o_i + D]$ , where  $\eta$  is the learning rate,  $o_j$  and  $o_i$  are the activation levels of the presynaptic and postsynaptic neurons and  $A$ – $D$  are the correlation term, presynaptic term, postsynaptic term, and constant, respectively. However, recent studies [18, 25] suggest that more elaborate forms of learning require mechanisms beyond Hebbian plasticity, in particular *neuromodulation*.

In a neuromodulated network, other neurons can change the degree of potential plasticity between separate pre- and postsynaptic neurons based on their activation levels. The benefit of adding neuromodulation is that it allows the ANN to change the level of plasticity on specific neurons at specific times. This property seems to play a critical role in regulating learning behavior in animals [5] and neuromodulated networks have a clear advantage in more complex dynamic, reward-based scenarios: Soltoggio et al. [25] showed that networks with neuromodulated plasticity and the same evolved Hebbian rule at each synapse significantly outperform fixed-weight recurrent and traditional adaptive ANNs without neuromodulation in a discrete T-Maze domain.

However, the experiment in this paper evolves controllers for a significantly more challenging *continuous T-Maze*. This advance is enabled by an enhancement called *adaptive HyperNEAT* that can evolve patterns of *heterogeneous rules* within the same network [21]. It 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) and can thus escape from the trap of evolving only a single rule. To facilitate this capability, instead of outputting only a *weight*, the CPPN also outputs the coefficients for the generalized Hebbian plasticity rule. In effect, the CPPN computes an entire pattern of rules as a function of the neural geometry, allowing it to distribute heterogeneous rules in a regular pattern across the network. Risi and Stanley [21] showed that this approach can solve a discrete T-Maze with every parameter for the entire evolved learning network compactly encoded by the CPPN.

### III. UNIFIED APPROACH: ADAPTIVE ES-HYPERNEAT

To date no method in neuroevolution unifies the ability to indirectly encode connectivity through geometry, simultaneously encode the density and placement of nodes in space, and generate patterns of heterogeneous plasticity. While such a unified approach risks being ad hoc, HyperNEAT offers a unifying principle that naturally expresses both node placement and plasticity. That is, both can be encoded by the CPPN as a pattern situated within the geometry of the substrate. Thus this section introduces for the first time the complete *adaptive ES-HyperNEAT* approach that can evolve increasingly complex neural geometry, density, and plasticity.

Adaptive ES-HyperNEAT augments the four-dimensional CPPN that normally encodes connectivity patterns with six *additional* outputs beyond the usual weight output: learning rate  $\eta$ , correlation term  $A$ , presynaptic term  $B$ , postsynaptic term  $C$ , constant  $D$ , and modulation parameter  $M$ . The pattern produced by parameter  $M$  encodes the modulatory connections, as explained in the next paragraph. When the CPPN is initially queried, these parameters are permanently stored for each connection, which allows the weights to be modified during the lifetime of the ANN. In addition to its standard activation, each neuron  $i$  also computes its modulatory activation  $m_i$  based on the types of its incoming connections:  $m_i = \sum_{w_{ji} \in Mod} w_{ji} \cdot o_j$ . The weight of a connection between neurons  $i$  and  $j$  then changes following the  $m_i$ -modulated plasticity rule:  $\Delta w_{ji} = \tanh(m_i/2) \cdot \eta \cdot [A o_j o_i + B o_j + C o_i + D]$ . Thus HyperNEAT allows connection-mediated neuromodulation (i.e. connections *instead* of neurons are either standard or neuromodulatory) because that way HyperNEAT can efficiently encode plasticity as a pattern of local rules.

The other important component of adaptive ES-HyperNEAT is that it determines the placement and density of nodes from implicit information in the hypercube encoded by the CPPN, as in ES-HyperNEAT alone [22]. Among the seven CPPN outputs, the node placement and connectivity is determined by searching only through the patterns produced by the weight output  $W$  (to discover regular connections) and modulatory output  $M$  (to discover modulatory connections). These patterns in effect represents a priori information at the “birth” of the ANN, which by convention then determines node placement. As noted earlier, the node-placement algorithm

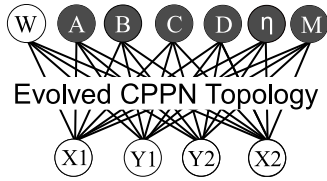


Fig. 4. **Adaptive ES-HyperNEAT CPPN.** This CPPN is only activated once for each queried connection to determine its weight and learning parameters. This model completely determines weights, plasticity, and modulation through a single vector function encoded by the CPPN. Node placement is further determined by implicit information in the pattern within the hypercube, following the ES-HyperNEAT approach.

searches for areas of high variances within the hypercube, in effect giving the CPPN a *language* to express density patterns [22]. Once all hidden neurons are discovered, only those are kept that have a path to an input *and* output neuron.

Thus adaptive ES-HyperNEAT is able to fully determine the geometry, density, and plasticity of an evolving ANN based on a function of its geometry. While past work on ES-HyperNEAT [22] has shown the benefits of evolving the HyperNEAT substrate, the next section presents an experiment designed to demonstrate the advantages of augmenting the model with synaptic plasticity.

#### IV. CONTINUOUS T-MAZE DOMAIN

The domain investigated in this paper is the T-Maze, which is often studied in the context of operant conditioning of animals; it is also studied to assess the ability of plastic ANNs [3, 25]. Thus it makes a good test of the ability to evolve adaptive neural structures. The domain consists of two arms that either contain a high or low reward (figure 5a). The simulated robot begins at the bottom of the maze and its goal is to navigate to the reward position. This procedure is repeated many times during the robot’s lifetime. When the position of the high reward sometimes changes, the robot should alter its strategy accordingly to explore the other arm of the maze in the next trial and *remember* the new position in the future (requiring adaptation). The goal of the robot is to maximize the amount of reward collected over four independent deployments with varying initial reward positions and switching times. Both the abilities to evolve neural density and to adapt should come into play.

The most important aspect of this domain is that it is not the traditional discrete grid-world T-Maze that is popular in this area [25]; rather, the more realistic domain presented here requires the robot to develop both collision avoidance and the ability to learn during its lifetime in a continuous world. Similar experiments with neuromodulated ANNs were performed by Dürr et al. [9] but their robot relied on additional ANN inputs (e.g. an input that signaled the end of the maze and an input that was activated once the T-Maze junction came into view) to solve the task. Additionally, to facilitate the task, their robots had only two infrared sensors whose values were merged into one sensory input. The increased number of rangefinder inputs (five) in the present experiment

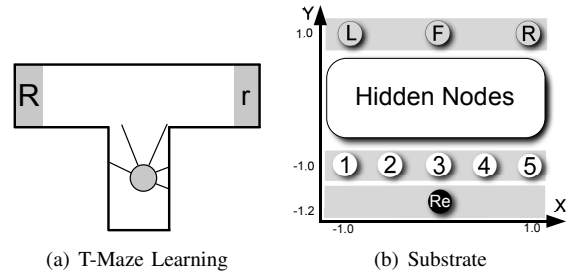


Fig. 5. **Domain and Substrate Configuration.** (a) The challenge for the robot in the T-Maze domain is to remember the location of the high reward from one trial to the next. (b) The simulated robot is equipped with five distance sensors (white) and one reward sensor (black).

gives the simulated robot a finer resolution without the need for any sensor preprocessing. Blynel and Floreano [3] also solved a version of the continuous task, but with CTRNNs instead of plastic networks, which often required incremental evolution. Therefore, this challenging version of the T-Maze domain should test adaptive ES-HyperNEAT’s ability to directly evolve more sophisticated plastic ANNs.

##### A. Experimental Setup

To generate a controller for the T-Maze, the evolved CPPNs query the substrate shown in figure 5b. The placement of input and output nodes on the substrate is designed to geometrically correlate sensors and effectors (e.g. seeing something on the left and turning left). Thus the CPPN can exploit the geometry of the robot. The simulated robot is equipped with five rangefinder sensors that are scaled into the range  $[0, 1]$ . The *Reward* input remains zero during navigation and is set to the amount of reward collected at the maze end (e.g. 0.1 or 1.0). At each discrete moment of time, the number of units moved by the robot is  $20F$ , where  $F$  is the forward effector output. The robot also turns by  $17(R - L)$  degrees, where  $R$  is the right effector output and  $L$  is the left effector output.

Because prior research has shown that only measuring performance based on the amount of collected reward is not a good indicator of the adaptive capabilities of the agent [23], the fitness function in this paper rewards consistently collecting the same reward, with a score of 1.0 for each low reward and a score of 2.0 for each high reward collected in sequence.

##### B. Experimental Parameters

All experiments were run with a modified version of the public domain SharpNEAT package (Green [15]). More information on ES-HyperNEAT and its source code can be found at: <http://eplex.cs.ucf.edu/ESHHyperNEAT>. The size of each population was 300 with 10% elitism and a termination criterion of 1,000 generations. Sexual offspring (50%) did not undergo mutation. Asexual offspring (50%) had 0.94 probability of weight mutation, 0.03 chance of CPPN link addition, and 0.02 chance of CPPN node addition. The available CPPN activation functions were sigmoid, Gaussian, absolute value, and sine, all with equal probability of being added. As in previous work [32] all CPPNs received the length of the queried connection as an additional input.

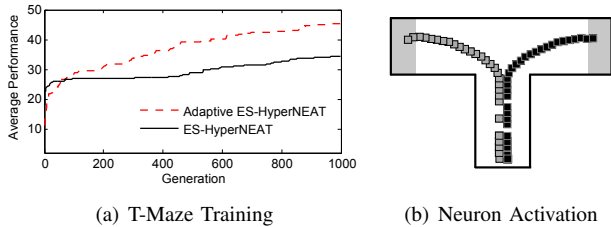


Fig. 6. **Average Performance.** The average best fitness over generations is shown for the T-Maze domain (a). The main result is that adaptive ES-HyperNEAT significantly outperforms ES-HyperNEAT because it allows the robot to more easily adapt during its lifetime. The activation of two hidden neurons from a T-Maze solution ANN is shown in (b). Only on trials with the high reward to the right does one of the two neurons show a positive activation (black) while the robot passes through the maze and turns onto the right choice arm. The other neuron only shows a positive activation (gray) on trials with the reward to the left, indicating an episodic-like encoding of events [10].

## V. RESULTS

Whereas ES-HyperNEAT alone could only solve the T-Maze task in one out of 30 runs (by storing state information through recurrent connections), adaptive ES-HyperNEAT found a solution in 19 out of 30 runs, in 426 generations on average ( $\sigma = 257$ ) when successful (figure 6a). This result suggests that augmenting ES-HyperNEAT with the ability also to encode plastic ANNs is important for tasks that require the agent to adapt. Evolved solutions always collect the maximum-possible reward without colliding and only require a minimum amount of exploration (i.e. collecting a low reward) at the beginning of their lifetime and when the reward changes position. Unlike in Dürr et al. [9], the ANNs have no special sensors to designate key locations and they also take raw rangefinder input.

Figure 6b also shows that the neural dynamics start to resemble particular dynamics found in nature. Similarly to cells in the rat’s hippocampus [10], individual hidden neurons in a typical ANN solution only fire when the robot is in a left or right-side trial, indicating an episodic-like coding of events (i.e. each trial is encoded separately).

Figure 7 shows an example ANN solution and its underlying CPPN. The ANN has 149 connections and 32 hidden nodes. With seven parameters per connection, the network has a total of 1,043 parameters. In contrast, the CPPN that encodes this network has only 54 connections and 6 hidden nodes; thus it is 19 times smaller than the network it encodes. In this way, HyperNEAT is able to explore a significantly smaller search space (i.e. CPPNs) while still creating complex structures (i.e. substrates). The ANN includes both neuromodulatory and plastic connections, showing that the full heterogeneous plasticity of an adaptive network can be encoded by a single compact CPPN. Different regions of the substrate show varying levels of node density and the majority of the modulatory connections originate from the right side of the substrate. Similar neural structures can also be found in the brain, where groups of neurons, called diffuse modulatory systems, project to numerous other areas [1].

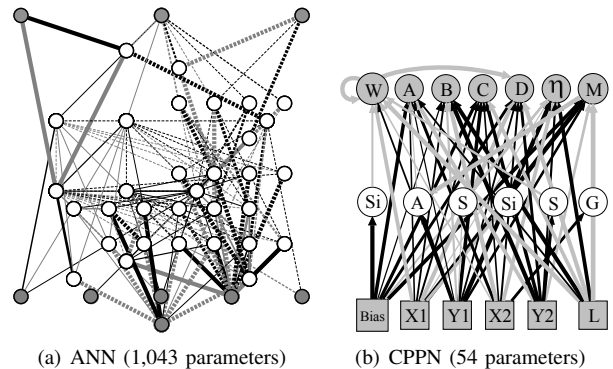


Fig. 7. **Example solution ANN created by Adaptive ES-HyperNEAT and its underlying CPPN.** Positive connections are dark whereas negative connections are light. Modulatory connections are dotted and line width corresponds to connection strength. The algorithm discovered this ANN solution (a) by extracting the information inherent in the simpler underlying CPPN (b). CPPN activation functions are denoted by  $G$  for Gaussian,  $S$  for sigmoid,  $Si$  for sine, and  $A$  for absolute value. The CPPN receives the length of the queried connection  $L$  as an additional input.

## VI. DISCUSSION AND FUTURE WORK

The main results signify progress in the field of neuroevolution. Evolved ANNs are beginning to assume more natural properties such as topography, regularity, heterogeneous plasticity, and neuromodulation. Furthermore, the results in the continuous T-Maze demonstrate that these capabilities combine effectively in a task that requires adaptation. Perhaps the most important point is that progress in this area has been cumulative: rather than separate methods that depend on entirely different architectures, the techniques combined in this study are built one upon another. In effect, a new research trajectory is coming into focus. For the field of AI, the idea that we are beginning to be able to reproduce some of the phenomena produced through natural evolution at a high level of abstraction is important because the evolution of brains ultimately produced the seat of intelligence in nature.

For example, adaptive ES-HyperNEAT can potentially later be applied to a neural-inspired navigation domain that requires map-based reasoning. An interesting question in such a domain is whether an architecture reminiscent of *place cells*, which are currently the object of intense study in neuroscience [19], might evolve. If they did, or if an alternative structure emerged, it would provide insight into possible neural organizations for intelligent thought.

## VII. CONCLUSION

This paper tied several strands of research in neuroevolution into a unified approach. For the first time, the geometry, density, and plasticity of ANNs was evolved based on a function of their neural geometry. Results in a maze-learning domain demonstrated that the presented approach can compactly encode and discover neural geometry and plasticity that exhibit more natural features than traditional ANNs. The conclusion is that it is becoming possible to produce more natural neural structures through artificial evolution, which

increasingly opens up opportunities for discovery relevant to the broader field of AI.

## VIII. ACKNOWLEDGMENTS

This material is based upon work supported by the US Army Research Office under Award No. W911NF-11-1-0489 and the DARPA Computer Science Study Group (CSSG Phase 3) Grant No. N11AP20003. It does not necessarily reflect the position or policy of the government, and no official endorsement should be inferred.

## REFERENCES

- [1] M. Bear, B. Connors, and M. Paradiso. *Neuroscience: Exploring the brain*. Lippincott Williams & Wilkins, 2007.
- [2] P. J. Bentley and S. Kumar. The ways to grow designs: A comparison of embryogenies for an evolutionary design problem. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999)*, pages 35–43, San Francisco, 1999. Kaufmann.
- [3] J. Blynel and D. Floreano. Exploring the T-Maze: Evolving learning-like robot behaviors using CTRNNs. In *2nd European Workshop on Evolutionary Robotics (EvoRob 2003)*, Lecture Notes in Computer Science, 2003.
- [4] J. C. Bongard. Evolving modular genetic regulatory networks. In *Proceedings of the 2002 Congress on Evolutionary Computation*, 2002.
- [5] T. Carew, E. Walters, and E. Kandel. Classical conditioning in a simple withdrawal reflex in *Aplysia californica*. *The Journal of Neuroscience*, 1(12):1426–1437, 1981.
- [6] 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.
- [7] J. Clune, K. Stanley, R. 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.
- [8] 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.
- [9] P. Dürri, C. Mattiussi, and D. Floreano. Genetic representation and evolvability of modular neural controllers. *IEEE Computational Intelligence Magazine*, 2010.
- [10] H. Eichenbaum. A cortical–hippocampal system for declarative memory. *Nature Reviews Neuroscience*, 1(1):41–50, 2000.
- [11] D. Floreano and J. Urzelai. Evolutionary robots with online self-organization and behavioral fitness. *Neural Networks*, 13: 431–443, 2000.
- [12] D. Floreano, P. Dürri, and C. Mattiussi. Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1(1):47–62, 2008.
- [13] 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.
- [14] J. Gauci and K. O. Stanley. Autonomous evolution of topographic regularities in artificial neural networks. *Neural Comput.*, 22:1860–1898, 2010.
- [15] C. Green. SharpNEAT homepage. <http://sharpneat.sourceforge.net/>, 2003–2006.
- [16] G. S. Hornby and J. B. Pollack. Creating high-level components with a generative representation for body-brain evolution. *Artificial Life*, 8(3), 2002.
- [17] E. R. Kandel, J. H. Schwartz, and T. M. Jessell, editors. *Principles of Neural Science*. Elsevier, Amsterdam, third edition, 1991.
- [18] Y. Niv, D. Joel, I. Meilijson, and E. Ruppín. Evolution of reinforcement learning in uncertain environments: A simple explanation for complex foraging behaviors. *Adaptive Behavior*, 10(1):5–24, 2002. ISSN 1059-7123.
- [19] J. O’Keefe and J. Dostrovsky. The hippocampus as a spatial map. Preliminary evidence from unit activity in the freely-moving rat. *Brain research*, 34(1):171, 1971.
- [20] T. Reil and P. Husbands. Evolution of central pattern generators for bipedal walking in a real-time physics environment. *IEEE Trans. Evolutionary Computation*, 6(2):159–168, 2002.
- [21] 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 Animals 11*, volume 6226 of *Lecture Notes in Computer Science*, pages 533–543. Springer Berlin / Heidelberg, 2010.
- [22] S. Risi and K. O. Stanley. Enhancing ES-HyperNEAT to evolve more complex regular neural networks. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2011)*, pages 1539–1546, New York, NY, USA, 2011. ACM.
- [23] S. Risi, C. E. Hughes, and K. O. Stanley. Evolving plastic neural networks with novelty search. *Adaptive Behavior - Animals, Animals, Software Agents, Robots, Adaptive Systems*, 18:470–491, December 2010. ISSN 1059-7123.
- [24] S. Risi, J. Lehman, and K. O. Stanley. Evolving the placement and density of neurons in the hyperneat substrate. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2010)*, pages 563–570, 2010.
- [25] A. Soltoggio, J. A. Bullinaria, C. Mattiussi, P. Dürri, 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.
- [26] O. Sporns. Network analysis, complexity, and brain function. *Complexity*, 8(1):56–60, 2002.
- [27] 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.
- [28] K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10: 99–127, 2002.
- [29] K. O. Stanley and R. Miikkulainen. A taxonomy for artificial embryogeny. *Artificial Life*, 9(2):93–130, 2003.
- [30] K. O. Stanley and R. Miikkulainen. Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, 21:63–100, 2004.
- [31] K. O. Stanley, B. D. Bryant, and R. Miikkulainen. Real-time neuroevolution in the NERO video game. *IEEE Transactions on Evolutionary Computation*, 9(6):653–668, December 2005.
- [32] 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.
- [33] P. Strobach. Quadtree-structured recursive plane decomposition coding of images. *Signal Processing*, 39:1380–1397, 1991.
- [34] P. Verbancsics and K. O. Stanley. Evolving static representations for task transfer. *Journal of Machine Learning Research*, 99: 1737–1769, August 2010.
- [35] X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.
- [36] M. J. Zigmond, F. E. Bloom, S. C. Landis, J. L. Roberts, and L. R. Squire, editors. *Fundamental Neuroscience*. Academic Press, London, 1999.