Transfer Learning through Indirect Encoding

In: *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2010).* New York, NY: ACM, 2010 (8 pages)

Phillip Verbancsics
School of EECS
University of Central Florida
Orlando, FL 32816, USA
verb@eecs.ucf.edu

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

ABSTRACT

An important goal for the generative and developmental systems (GDS) community is to show that GDS approaches can compete with more mainstream approaches in machine learning (ML). One popular ML domain is RoboCup and its subtasks (e.g. Keepaway). This paper shows how a GDS approach called HyperNEAT competes with the best results to date in Keepaway. Furthermore, a significant advantage of GDS is shown to be in transfer learning. For example, playing Keepaway should contribute to learning the full game of soccer. Previous approaches to transfer have focused on transforming the original representation to fit the new task. In contrast, this paper explores transfer with a representation designed to be the *same* even across different tasks. A *bird's eye view* (BEV) representation is introduced that can represent different tasks on the same two-dimensional map. Yet the problem is that a raw two-dimensional map is high-dimensional and unstructured. The problem is addressed naturally by *indirect encoding*, which compresses the representation in HyperNEAT by exploiting its geometry. The result is that the BEV learns a Keepaway policy that transfers from two different training domains *without further learning* or manipulation. The results in this paper thus show the power of GDS versus other ML methods.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Performance, Experimentation

Keywords

Generative and Developmental Systems, Artificial Neural Networks, Task Transfer, RoboCup Soccer

1. INTRODUCTION

As generative and developmental systems (GDS) matures as a field, the question of where it provides a practical advantage comes into greater focus. The ability to evolve large structures and regularities should prove advantageous in the right context. The idea in this paper is that these capabilities can be exploited in *task transfer*, i.e. bootstrapping learning on one task to facilitate learning another task [3, 26, 28]. To demonstrate the unique potential of GDS in transfer, a static state representation is introduced called a *bird's eye view* (BEV), which is a two-dimensional depiction of objects on the ground from above. Its advantage is that its dimensionality is constant no matter how many objects are in the environment. Thus even if the task is transferred to a version with more objects, the representation remains the same (i.e. static), simplifying task transfer.

The challenge for the BEV is that representing a high-resolution planar field requires a high-dimensional state space. For example, for an artificial neural network (ANN) to process such a representation at high resolution would require a high number of inputs. The resultant "curse of dimensionality" means a large state space must be learned from a small proportion of examples [1, 2]. While this problem might have prohibited such a representation in the past, *indirect encodings*, which compress the solution representation by reusing information, significantly expand the scope of possible representations. The indirect encoding in this paper, called a *compositional pattern producing network* (CPPN; [15]), represents ANN mappings between high-dimensional spaces by exploiting *regularities* in their geometry, which is well-suited to the BEV. An evolutionary algorithm called Hypercube-based NeuroEvolution of Augmenting Topologies (HyperNEAT; [8, 9, 16]) that is designed to evolve CPPNs is therefore able to learn effectively from the BEV.

The HyperNEAT BEV is tested in variants of the popular Robocup Keepaway soccer benchmark [20]. One interesting result is the longest holding time in the 3 vs. 2 Keepaway variant yet recorded, demonstrating the ability of GDS to compete with other ML methods on this task. Additionally, unlike any method so far, HyperNEAT can transfer from 3 vs. 2 to 4 vs. 3 Keepaway with no change in representation and no further learning. Finally, transfer from the significantly simpler benchmark task Knight Joust [28] to Keepaway results in enhanced learning, both instantaneously and upon further training. These results demonstrate the critical role of representation and the power of GDS in transfer.

2. BACKGROUND

The geometry-based methods that underlie the static representation are introduced first in this paper and task transfer is then discussed.

2.1 NeuroEvolution of Augmenting Topologies (NEAT)

This section briefly reviews the NEAT evolutionary algorithm [17, 19], a popular policy search method that evolves ANNs. NEAT evolves connection weights as well as adds new nodes and connections over generations, thereby increasing solution complexity. It has been proven to be effective in challenging control and decision making tasks [19, 29, 32, 33]. NEAT starts with a population of small, simple ANNs that increase their complexity over generations by adding new nodes and connections through mutation. That way, the topology of the network does not need to be known a priori; NEAT searches through increasingly complex networks as it evolves their connection weights to find a suitable level of complexity. The techniques that facilitate evolving a population of diverse and increasingly complex networks are described in detail in Stanley and Miikkulainen [17]; Stanley and Miikkulainen [19]; the important concept for the approach in this paper is that NEAT is a policy search method that discovers the right topology and weights of a network to maximize performance on a task. The next section reviews the extension of NEAT called HyperNEAT that allows it to exploit geometry through representation.

2.2 CPPNs and HyperNEAT

NEAT is a vehicle to study alternate representations because it easily extends to an *indirect encoding*, which means a *compressed* description of the solution network. Such compression makes the policy search practical even if the state space is high-dimensional. One effective indirect encoding is to compute the network structure as a function of geometry. This section describes such an extension of NEAT, called Hypercube-based NEAT (HyperNEAT; [8, 9, 16]), which enables high-dimensional static representations. The effectiveness of the geometry-based learning in HyperNEAT has been demonstrated in multiple domains and representations, such as checkers [8, 9], multi-agent predator prey [5, 6], visual discrimination [16], and quadruped locomotion [4]. A full description of HyperNEAT is in Gauci and Stanley [8, 9]; Stanley et al. [16].

The main idea in HyperNEAT is that geometric relationships are learned though an indirect encoding that describes how the *connectivity* of the ANN can be *generated* as a function of geometry. Unlike a *direct* representation, wherein every connection in the ANN is described individually, an indirect representation describes a pattern of parameters without explicitly enumerating each such parameter. That is, information is reused in such an encoding, which is a major focus in the field of GDS from which HyperNEAT originates [18, 31]. Such information reuse allows indirect encoding to search a compressed space. HyperNEAT discovers the *regularities* in the geometry and learns a policy based on them.

The indirect encoding in HyperNEAT is called a *compositional pattern producing network* (CPPN; [15]), which encodes the *connectivity pattern* of an ANN [8, 16]. The idea behind CPPNs is that geometric patterns can be encoded by a *composition of functions* that are chosen to represent common regularities. Given a function f and a function g, a composition is defined as $f \circ g(x) = f(g(x))$. Thus, a set of simple functions can be composed into more elaborate functions through hierarchical composition (e.g. $f \circ g(f(x) + g(x))$). For example, the Gaussian function is symmetric, so when it is composed with any other function, the result is a symmetric pattern. The internal structure of a CPPN is a weighted network, similar to an ANN, that denotes which functions are composed and in what order. The appeal of this encoding is that it can represent a pattern of connectivity, with regularities such as symmetry, repetition, and repetition with variation, through a network of simple functions (i.e. the CPPN), which means that instead of evolving ANNs, NEAT can evolve *CPPNs* that generate ANN connectivity patterns.

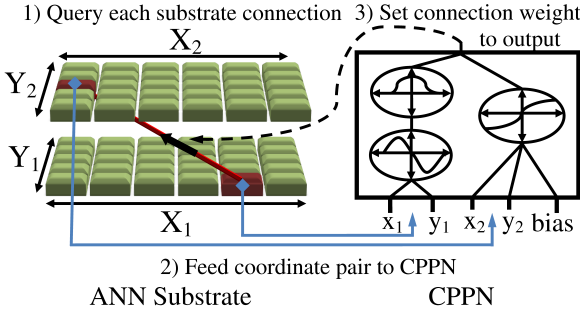


Figure 1: A CPPN Describes Connectivity. A grid of nodes, called the ANN *substrate*, is assigned coordinates. (1) Every connection between layers in the substrate is queried by the CPPN to determine its weight; the line connecting layers in the substrate represents a sample such connection. (2) For each such query, the CPPN inputs the coordinates of the two endpoints, which are highlighted on the input and output layers of the substrate. (3) The weight between them is output by the CPPN. Thus, CPPNs can generate regular patterns of connections.

Formally, CPPNs are *functions* of geometry (i.e. locations in space) that output connectivity patterns whose nodes are situated in n dimensions, where n is the number of dimensions in a Cartesian space. Consider a CPPN that takes four inputs labeled x_1, y_1, x_2, and y_2; this point in four-dimensional space can *also* denote the connection between the two-dimensional points (x_1, y_1) and (x_2, y_2). The output of the CPPN for that input thereby represents the weight of that connection (figure 1). By querying every pair of points in the space, the CPPN can produce an ANN, wherein each queried point is the position of a neuron. While CPPNs are themselves networks, the distinction in terminology between CPPN and ANN is important for explicative purposes because in HyperNEAT, CPPNs *encode* ANNs. Because the connection weights are produced as a function of their endpoints, the final structure is produced with *knowledge* of the domain geometry, which is literally depicted geometrically within the constellation of nodes.

In effect, the CPPN paints a pattern within a n-dimensional hypercube that is interpreted as an isomorphic connectivity pattern. Connectivity patterns produced by a CPPN in this way are called *substrates* so that they can be verbally distin-

```
Input: Substrate Configuration
Output: Solution CPPN
Initialize population of minimal CPPNs with random weights;
while Stopping criteria is not met do
    foreach CPPN in the population do
        foreach Possible connection in the substrate do
            Query the CPPN for weight w of connection;
            if Abs(w) > Threshold then
                Create connection with a weight scaled
                proportionally to w (figure 1);
            end
        end
        Run the substrate as an ANN in the task
        domain to ascertain fitness;
    end
    Reproduce CPPNs according to the NEAT method
    to produce the next generation;
end
Output the Champion CPPN;
```

Algorithm 1: Basic HyperNEAT Algorithm

guished from the CPPN itself. It is important to note that the structure of the substrate is independent of the structure of the CPPN. The substrate is an ANN whose nodes are situated in a coordinate system, while the CPPN defines the connectivity among the nodes of the ANN. The experimenter defines both the location and role (i.e. hidden, input, or output) of each node in the substrate.

As a rule of thumb, nodes are placed on the substrate to reflect the geometry of the domain (i.e. the state), making setup straightforward [4, 9, 16]. For example, a visual field can be laid out in two dimensions such that nodes that receive input from adjacent locations in the image are literally adjacent in the network geometry. This way, the connectivity becomes a direct function of the domain geometry, which means that knowledge about the problem can be injected into the search and HyperNEAT can exploit the regularities (e.g. adjacency, or symmetry, which the CPPN sees) of a problem that are invisible to traditional encodings. For example, geometric knowledge can be imparted e.g. by including a hidden node in the CPPN that computes $Gauss(x_2 - x_1)$, which encodes the concept of locality and symmetry on the x-axis, an idea employed in the implementation in this paper. This idea can be illustrated by considering three source points with x positions $-1, 0, 1$ and one target point at $x = 0$.

$$w_1 = Gauss(0 + 1) = 0.37$$

$$w_2 = Gauss(0 - 0) = 1$$

$$w_3 = Gauss(0 - 1) = 0.37$$

Thus, the output of this function is both symmetric about the x-axis and decreasing with distance from the target.

In summary, instead of evolving the ANN directly, HyperNEAT evolves the internal topology and weights of the CPPN that *encodes* it, which is significantly more compact. The process is shown in Algorithm 1. Next, task transfer is discussed.

2.3 Task Transfer

Task transfer means applying knowledge learned in one task to a new, related task [3, 26, 28]. This capability is becoming increasingly important as the tasks studied increase in complexity and learning must be applied to novel situations. *Transfer via inter-task mapping for policy search methods* (TVITM-PS; [28]) is one such leading transfer method and the only for which results are reported in Keepaway, which is also the benchmark domain in this paper.

In TVITM-PS, a transfer functional ρ is defined to transform the policy π for a source task into the policy for a target task, such that $\rho(\pi_{source}) = \pi_{target}$. This functional is often hand-coded based on domain knowledge, though learning it is possible. When there are novel state variables or actions, an *incomplete mapping* is defined from the source to the target. TVITM-PS can be adapted to multiple representations. For example, in an ANN, input or output nodes whose connections are not defined in the mapping (i.e. it is incomplete) are made fully connected to the existing hidden nodes with random weights. This incomplete mapping implies that further training is needed to optimize the policies with respect to the new state variables and actions. However, it makes it possible to begin in the target domain from a better starting point than from scratch. TVITM-PS is a milestone in task transfer because it introduced a formal approach to moving from one domain to another that defined how ambiguous variables in the target domain should be treated.

This paper adds to our understanding of task transfer by focusing on the role of representation. The next section explains how indirect encoding makes it possible to learn from a bird's eye view.

3. APPROACH: BIRD'S EYE VIEW

A challenge for state representation in reinforcement learning tasks is that specific state variables are often tied to individual objects, which makes it difficult to add more such objects without expanding the space [28]. This section proposes the bird's eye view (BEV) perspective, which scales to higher complexity states without altering the representation. The BEV representation is explained first, followed by its implementation, which is an ANN whose connectivity is trained by HyperNEAT.

3.1 Bird's Eye View

Humans often visualize data from a BEV. Examples include maps for navigation, construction blue prints, and sports play books. Key to these representations is that they remain the same (i.e. they are *static*) no matter how many objects are represented on them. For example, a city map does not change size or format when new buildings are constructed or new roads are created. Additionally, the physical geometry of such representations allow agents to understand spatial relationships among objects in the environment by placing them in the context of physical space. The BEV also implicitly represents its borders by excluding space outside them from its field of view. As suggested in Kuipers' Spatial Semantic Hierarchy, such *metrical* representation of the geometry of large-scale space is a critical component of human spatial reasoning [10].

A distinctive feature of the proposed representation is that the agent *also* requests *actions* within the same BEV perspective. For example, to request a pass the agent can indicate its target by simply highlighting it on a two-dimensional output array. That way, instead of making decisions blind to the geometry of physical space, it can be taken into account.

Egocentric data (figure 2a) can be mapped to an equivalent BEV by translating from local (relative) coordinates to global coordinates established by static points of reference (i.e. fiducials). The global coordinates mark the location of objects in the BEV (figure 2b). This translation allows the mapping of any number of objects into the static representation of the BEV. Importantly, the continuous coordinate system must be discretized so that each variable in the state representation corresponds to a single discrete location. This discretization allows the two-dimensional field to be represented with a finite set of parameters. The values of these parameters denote objects in their regions. However, because the solution is indirectly encoded, the discretization resolution can be arbitrarily increased. Interestingly, while the traditional egocentric Keepaway representation (figure 2a) is designed specifically to include the relationships that people believe are important for Keepaway, the BEV is entirely generic, i.e. it is just a two-dimensional field, and therefore could be applied to any domain that takes place in two dimensions.

Note that while the division of the field in figure 2b appears reminiscent of *tile coding* [25], that appearance is superficial because (1) a tile coding of the state variables in figure 2a would still be egocentric whereas the BEV is not, and (2) tile coding breaks the state representation into chunks that can be optimized separately whereas the HyperNEAT CPPN derives the connectivity of the policy network directly from the geometric relationships among the squares in figure 2b, as explained next.

3.2 HyperNEAT: Learning from the BEV

To understand the impact of learning from the true geometry of the domain, consider a two-dimensional field converted to a traditional vector of parameters, which removes the geometry (figure 2c). In contrast, HyperNEAT *sees* the task geometry, thereby exploiting geometric regularities and relationships, such as locality, which the BEV makes explicit.

For HyperNEAT to exploit patterns in a two-dimensional BEV (e.g. in soccer), the geometry of the input layer of the substrate is made two-dimensional, as in figure 3, which comes naturally from the domain. That way, CPPNs can compute the connectivity of the substrate as a function of that geometry. The x and y coordinates of each input unit are in the range $[-1, 1]$. Furthermore, the output layer of the substrate matches the dimensions of the BEV so that the CPPN can exploit the geometric relationship between the input space and output space as well (figure 3). That the *outputs* are themselves a discretized two-dimensional plane is another significant difference from tile coding. Each coordinate in this substrate represents a discretized region of the overhead view of physical space. A four-dimensional CPPN with inputs x_1, y_1, x_2, and y_2 determines the weights between coordinates in the two-dimensional input layer and the two-dimensional output layer, creating a pattern of connections between regions in the physical space. Thus, the internal connectivity of the ANN is completely defined by the pattern produced by the evolved CPPN. Connectivity may range from fully connected to sparsely connected and can include regularities based on properties such as locality. To represent world state, objects and agents are literally "drawn" onto the input substrate, like marking a map. The generated network then can make decisions based on the relationships of such features in physical space.

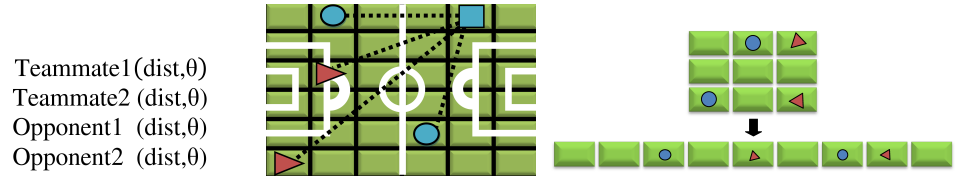
In this way, the BEV makes it possible to add new features (e.g. a new player) to the state space *without* the need to add new inputs. Instead, they can now simply be drawn onto the existing representation. That way, task transfer to different numbers of players is made simple. The next section introduces the experiment designed to demonstrate the benefits of this approach.

4. EXPERIMENTAL SETUP

The experiments in this paper are designed to investigate the benefit of indirect encoding in task transfer. This paper focuses on the idea that an effective representation for transfer is one that *does not need to change* from one task to the next. Indirect encoding facilitates such static representation by making possible high-dimensional geometric depictions of state. Because the representation is consistent, it has the potential to exhibit improved performance in the target domain immediately after transfer, without further learning. This section explains the domain, the methods that are compared, and the experiments.

The Robocup simulated soccer Keepaway domain [22] is suited to such an investigation because it is a popular RL benchmark and can be scaled to different numbers of agents to create new versions of the same task. Experiments are run on the Keepaway 0.6 player benchmark [20] and the Robocup Simulator Soccer Server v. 12.1.1 [13]. Robocup Keepaway is challenging because it includes a large state space, partially observable state, and noisy sensors and actuators. In Keepaway, *keepers* try to maintain possession of the ball within a fixed region and *takers* attempt to take it away. The number of agents and field size can be varied to make the task more or less difficult: The smaller the field and the more players in the game, the harder it becomes.

Each learning method in this paper is trained in the standard benchmark setup [23] of the three keepers versus two takers task on a 20m×20m field. In this setup, agents' sensors are noisy and their actions are nondeterministic. Takers follow static policies, wherein the first two takers go towards the ball and additional takers (i.e. when scaled to 4 vs. 3) attempt to block open keepers. The learner only controls the keeper who possesses the ball; its choices are to hold the ball or pass to a specific teammate. The keepers' reward is the length of time they hold ball. In the 3 vs. 2 task, 13 variables traditionally represent the agent's state [23]. These include each player's distance to the center of the field, the distance from the keeper with the ball to each other player, the distance from each other keeper to the closest taker, and the minimum angle between the other keepers and the takers. The three possible actions are holding the ball or passing to one of the other two keepers. The keeper with the ball remains stationary, while the other keepers follow a fixed



(a) Egocentric view (b) BEV (c) Losing Geometry

Figure 2: Alternative Representations of a Soccer Field and the Importance of Preserving Geometry. Several parameters (a) represent the agent's relationship with other agents on a soccer field (taken from a standard Robocup representation [13]). Each distance and angle pair represents a specific relationship of the agent to another agent. The BEV (b) represents the same relationships as paths in the geometric space. A square depicts the agent, circles depicts its teammates, and triangles its opponents. The overhead perspective also makes it possible to represent any number of agents without changing the representation. Interestingly, if a two-dimensional field is transformed into a traditional vector of parameters, it forfeits knowledge of the geometry of the domain (c).

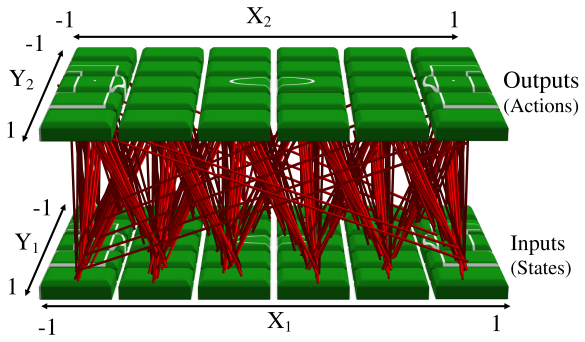


Figure 3: BEV Implemented in the Substrate. Each dimension ranges between $[-1, 1]$ and the input and output planes of the substrate are equivalently constructed to take advantage of geometric regularities between states and actions. Because CPPNs are an indirect encoding, the high dimensionality of the weights does not affect performance. (The CPPN is the search space.)

policy of moving toward the least congested position on the field, where *congestion* is defined as the number of players near that position.

To test the ability of the HyperNEAT BEV to learn this task, it is compared to both static policies [20] and the learning algorithms Sarsa [14], NEAT [19], and EANT [12]. The static benchmarks are Always-Hold, Random, and a Hand-Coded policy, which holds the ball if no takers are within 10m [21]. State Action Reward State Action (Sarsa; [14]) is an on-policy temporal difference RL method that learns the action-value function $Q(s, a)$. Each keeper separately learns which action to take in a given state to maximize the reward it receives [29]. Regular NEAT [17] evolves directly-encoded ANNs to maximize a fitness function. The ANNs receive 13 state inputs (like Sarsa) to define the state of the system and produce three outputs to select an action. EANT [12] is an additional directly-encoded neuroevolution algorithm based on NEAT that learned Keepaway. Though similar to NEAT, it distinguishes itself by more explicitly controlling the ratio of exploration to exploitation during evolution. The fitness in Robocup Keepaway is the average length of time that keepers can hold the ball over a number of trials [29].

Input: Substrate ANN, KeeperWithBall Position, KeeperWithoutBall Positions, Taker Positions
Reset all ANN input values to 0;
Set ANN inputs at coordinate of Keeper Positions to 1;
Set ANN inputs at coordinate of Taker Positions to -1;
foreach *KeeperWithoutBall Position* **do**
  Create Line from Keeper to KeeperWithBall;
  foreach *ANN input coordinate within Threshold distance of Line AND Between Keeper and KeeperWithBall* **do**
    Increment ANN input by 0.3;
  end
end
foreach *Taker Position* **do**
  Determine Line from Taker to KeeperWithBall;
  foreach *ANN input coordinate within Threshold distance of Line AND Between Taker and KeeperWithBall* **do**
    Decrement ANN input by 0.3;
  end
end

Algorithm 2: Drawing World State on the Substrate.

As described in Section 3, the HyperNEAT BEV transforms the traditional state representation to explicitly capture the geometry. The substrate is a 20×20 input layer connected to a 20×20 output layer and thus encompasses 160,000 potential connections. It is important to note that this dimensionality is far beyond the point after which direct encodings have been shown to struggle [16]. As with Sarsa in Stone and Sutton [21], this policy representation does not include a hidden layer. However, the CPPN that encodes its weights *does* evolve internal nodes. Thus the pattern of weights across the connections is itself potentially nonlinear. Each node in a substrate layer represents a $1m^2$ discrete chunk of Keepaway field. Each keeper's position is marked on the input layer with a positive value of 1.0 in its containing node and takers are similarly denoted by -1.0. Paths are literally drawn from the keeper with the ball to the other players (as in figure 2b and Algorithm 2). Positive values of 0.3 depict paths to other keepers and values of -0.3 depict paths to takers. These values for agents and paths are experimentally determined and robust to minor variation. Actions are selected from among the output

nodes (top layer of figure 3) that correspond to where the keepers are located: If the highest output is the node where the keeper with the ball is located, it holds the ball. Otherwise, it passes to the teammate with the highest output at its node. These action selections thus correspond exactly with the three actions available to Sarsa, NEAT, and EANT.

The population size in HyperNEAT is 100. Available CPPN activation functions are absolute value, bipolar sigmoid, Gaussian, linear, sine, and step. Signed activation is used, resulting in a node output range of $[-1, 1]$. By convention, a connection is not expressed if the magnitude of the corresponding CPPN output is below a minimal threshold of 0.2 [16]. The probability of adding a node to the CPPN is 0.05 and the probability of adding a connection is 0.18. These parameters were found to be robust to moderate variation in preliminary experimentation. Fitness is assigned according to the generated ANN's average ball possession time over at least 30 trials and up to 100 trials assigned to those above the mean, following Taylor et al. [29].

Task transfer is first evaluated by training a HyperNEAT BEV on the 3 vs. 2 task on a 25m×25m field (instead of the standard 20m×20m) and testing the trained BEVs on the 4 vs. 3 version of the task on the same field *without any further training*. The larger field accommodates the larger version of the task [30]. To switch from 3 vs. 2 to 4 vs. 3, the additional players and paths are simply drawn on the input layer as usual, with no transformation of the representation or further training. The resulting performance on 4 vs. 3 is compared to TVITM-PS ([30]; described in Section 2.3), which is the leading transfer method for this task. TVITM-PS results, which were obtained by Taylor et al. [30], are from policies represented by an ANN trained by NEAT [30]. Unlike the HyperNEAT BEV, TVITM-PS requires further training after transfer because ρ expands the ANN by adding new state variables.

Second, transfer is further evaluated by first training the HyperNEAT BEV on the simpler task of Knight Joust [27, 28]. In Knight Joust an agent and opponent are situated on opposite ends of a 20×20 grid world. The objective is that the agent reach the other side (i.e. the goal side) of the world without touching its opponent. The agent is allowed three types of moves: one square forward, one forward and two left, and one forward and two right. The opponent follows a static stochastic policy that attempts to intercept the agent. While Knight Joust is significantly different from Keepaway, a common aspect between them is that at each step the agent must make the decision that best avoids the opponent. Also, Knight Joust is simpler, eliminating such complexity as multiple agents, noise, and kicking a ball, making it more tractable.

Similarly to the change in representation in Keepaway, agent state is drawn onto the BEV as the equivalent paths. These paths are from the agent to the opponent and from the agent to the left and right corners of the goal side. Also, as in Keepaway, actions are selected by querying specific outputs: the square in front of the agent (move forward), the right goal side corner (right knight jump), and the left goal side corner (left knight jump). The best individuals from the end of such training seeds the beginning of training for Keepaway.

Method	Average Hold Time
HyperNEAT BEV	$15.4s$
EANT	$14.9s$
NEAT	$14.0s$
Sarsa	$12.5s$
Hand-tuned Benchmark	$8.3s$
Always Hold Benchmark	$7.5s$
Random Benchmark	$3.4s$

Table 1: Performance Results on the Benchmark Keepaway. Average best performance by method is shown. The HyperNEAT BEV holds the ball longer than previously reported best results for neuroevolution and temporal difference learning methods. Results are shown for Evolutionary Acquisition of Neural Topologies (EANT) from Metzen et al. [12], NeuroEvolution of Augmenting Topologies (NEAT) from Taylor et al. [29], and State Action Reward State Action (Sarsa) from Stone and Sutton [21].

5. RESULTS

In the Robocup Keepaway benchmark, performance is measured by the number of seconds that the keepers maintain possession [20, 21, 30]. After training, the champion of each epoch is tested over 1,000 trials. Performance results are averaged over five runs, each consisting of 50 generations of evolution, which takes approximately 1,000 hours of simulator time. In 3 vs. 2 Keepaway on the 20m×20m field, the best keepers from each of the five runs controlled by a BEV substrate trained by HyperNEAT maintain possession of the ball on average for 15.4 seconds ($sd = 1.31$), which significantly outperforms ($p < 0.05$) all static benchmarks (Table 1). Furthermore, assuming similar variance, this performance significantly exceeds ($p < 0.05$) the current best reported average results [22, 29] on this task for both temporal difference learning (12.5 seconds) and NEAT (14.0 seconds), and matches EANT (14.9 seconds; Table 1). This result establishes that the HyperNEAT BEV, an indirect encoding, is competitive with the top learning algorithms on a major benchmark task in machine learning.

In transfer learning the BEV is evaluated by testing individuals trained for 20 generations *only* on the 3 vs. 2 task on a 25m×25m field on *both* the 3 vs. 2 and 4 vs. 3 tasks for $1,000$ trials each. Note that this evaluation of transfer differs from Taylor et al. [30], in which teams trained on the smaller task are *further trained* on the larger task after the transfer because new parameters are added. Performance is averaged over five runs, following Taylor et al. [29]. Figure 4 shows the average test performance on *both* 3 vs. 2 (trained) and 4 vs. 3 (untrained; immediately after transfer) of each generation champion. Testing performance on the 3 vs. 2 task improves to 14.3 seconds on average over each run. At the same time, the test performance of these same individuals on the 4 vs. 3 task, which was not trained, improves from 6.6 seconds to 8.1 seconds on average. In contrast, the previous best approach to transfer learning in this domain required executing a transfer functional and additional training for between 50 and 200 hours (depending on the chosen

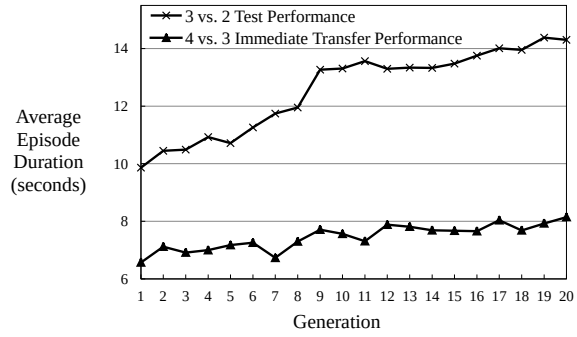


Figure 4: Transfer Results from 3 vs. 2 to 4 vs. 3 Keepaway. Transfer learning performance is shown. As the performance (averaged over five runs) of the champion on the 3 vs. 2 task improves, the transfer performance on the 4 vs. 3 task also consequently improves from 6.6 seconds to 8.1 seconds without *ever* training for it. The improvement is positively correlated ($r = 0.87$).

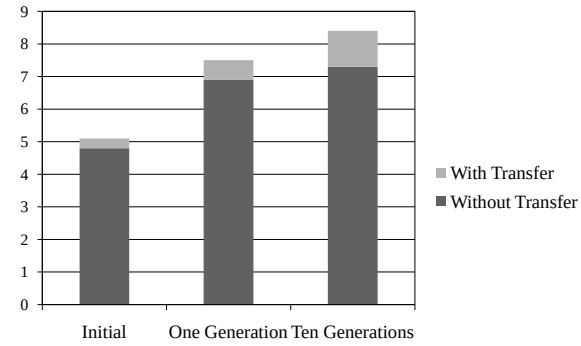


Figure 5: Transfer Results from Knight Joust to Keepaway. Direct transfer and further training performance averaged over 30 runs are shown. The initial champion performance from Knight Joust on Keepaway outperforms initial random individual by 0.3 seconds. After one generation, this advantage from transfer increased to 0.6 seconds and at 10 generations the advantage is 1.1 seconds. Thus, performance on Keepaway, both instantaneous and with further training, benefits from transfer from the Knight Joust domain with significance $p < 0.05$.

transfer function) *beyond* the initial bootstrap training in 3 vs. 2 to achieve 8.0 second episode duration [30]. Thus, with the BEV, transfer is instantaneous and requires no special adjustments to the representation to achieve the same result as many hours of *further* training with TVITM-PS.

Transfer is also evaluated from the simpler task of Knight Joust on a 20×20 grid to 3 vs. 2 Keepaway on a 20m×20m field. Evolution is run for 20 generations on the Knight Joust task and then the champions seed the beginning runs of the 3 vs. 2 Keepaway. Further training is then performed over ten additional generations of evolution. Performance of the champion genomes from Knight Joust is 0.3 seconds above the performance of initial random individuals. After one generation of evolution, the best individuals from transfer exceed the raw performance by 0.6 seconds. Finally, after ten further generations the best individuals with transfer hold the ball for 1.1 seconds longer than without transfer (figure 5). The results are averaged over 30 runs and are significant ($p < 0.05$). Thus even preliminary learning in a significantly different domain proved beneficial to the BEV. In contrast, previous transfer results from Knight Joust to Keepaway from Taylor and Stone [27] demonstrated an initial performance advantage, but after training for five simulator hours there was no performance difference between learning with transfer and without it.

6. DISCUSSION

TVITM-PS remains an important tool in task transfer for domains in which the representation must change with the task. However, the BEV shows that a carefully chosen representation with an *indirect* encoding can sometimes eliminate the need to change the representation. This property of the BEV is good news because, as a two-dimensional field, it is a highly generic representation that can apply to many tasks. In fact, it is precisely *because* it is so generic that it is able to remain static when transferring from one task to another. The deeper lesson is the critical role of representation in transfer and the consequent need for algorithms that can learn from relatively high-dimensional representations of task geometry and overcome the "curse of dimensionality" [1, 2, 7, 11, 24]. Indeed, the human eye contains *millions* of photoreceptors, which provide the same set of inputs to every visual task tackled by humans. Thus this paper provides initial evidence that the capabilities of GDS *do* indeed provide concrete advantages over more traditional approaches in some domains.

An exciting implication of this work is that the power of static transfer and indirect encoding can potentially bootstrap learning the complete game of soccer. After all, the key elements of soccer are present in Keepaway as well and even the most basic of geometric concepts present in Knight Joust contribute to learning Keepaway. RoboCup soccer is among the hottest applications in ML and the idea of static representation with indirect encoding in transfer can potentially contribute to scaling up to the full game in the future.

7. CONCLUSION

This paper demonstrated that GDS is competitive with leading methods in a popular benchmark domain, RoboCup, and introduced the idea that static representation facilitates transfer. With a static representation, no matter how many objects are in the domain, the size of the state representation remains the same. In contrast, in traditional representations, changing the number of players (e.g. in the Robocup Keepaway task) forces changes in the representation by adding dimensions to the state space. Importantly, the BEV, which is enabled by an indirect encoding, achieved transfer learning from 3 vs. 2 to 4 vs. 3 Keepaway *without* further training and demonstrated an advantage, both instantaneously and upon further training, in transfer learning from Knight Joust to Keepaway. These results highlight the critical role that representation plays in learning and transfer. Altering the representation simplifies transfer learning. Yet high-dimensional static representations require indirect

encodings that take advantage of their expressive power, such as in HyperNEAT. Such advanced representations in conjunction with indirect encoding can contribute to the transfer of learning to more challenging tasks, validating the practical importance of GDS as a scientific pursuit.

Acknowledgements

This research is supported in part by a Science, Mathematics, and Research for Transformation (SMART) fellowship from the American Society of Engineering Education (ASEE) and the Naval Postgraduate School.

8. REFERENCES

[1] Charu C. Aggarwal. On k-anonymity and the curse of dimensionality. In *Proc. of the 31st Int. Conf. on Very Large DB*, pages 901–909. VLDB Endowment, 2005.

[2] R. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[3] Rich Caruana. Multitask learning. In *Machine Learning*, pages 41–75, 1997.

[4] Jeff Clune, Benjamin E. Beckmann, Charles Ofria, and Robert T. Pennock. Evolving coordinated quadruped gaits with the hyperneat generative encoding. In *Proc. of the IEEE Congress on Ev. Comp. Special Section on Ev. Robotics*, Piscataway, NJ, USA, 2009. IEEE Press.

[5] David D'Ambroiso and Kenneth O. Stanley. Evolving policy geometry for scalable multiagent learning. In *Proc. of the 9th Int. Conf. on AAMAS*, page 8, New York, NY, USA, 2010. ACM Press.

[6] David B. D'Ambrosio and Kenneth O. Stanley. Generative encoding for multiagent learning. In *Proc. of the Genetic and Ev. Comp. Conf.*, New York, NY, 2008. ACM Press.

[7] Jerome H. Friedman. On bias, variance, 0/1—loss, and the curse-of-dimensionality. *Data Min. Knowl. Discov.*, 1(1):55–77, 1997.

[8] Jason Gauci and Kenneth O. Stanley. A case study on the critical role of geometric regularity in machine learning. In *Proc. of the 23rd AAAI Conf. on AI*, Menlo Park, CA, 2008. AAAI Press.

[9] Jason Gauci and Kenneth O. Stanley. Autonomous evolution of topographic regularities in artificial neural networks. *Neural Computation*, page 38, 2010. To appear.

[10] Benjamin Kuipers. The spatial semantic heirarchy. *Artifical Intelligence*, 119:191–233, 2000.

[11] Madhubanti Maitra and Amitava Chatterjee. A hybrid cooperative-comprehensive learning based pso algorithm for image segmentation using multilevel thresholding. *Expert Systems with Applications*, 34(2):1341 – 1350, 2008.

[12] Jan FH. Metzen, Mark Edgington, Yohannes Kassahun, and Frank Kirchner. Performance evaluation of EANT in the robocup keepaway benchmark. In *Proc. of the 6th Int. Conf. on ML and Apps.*, pages 342–347, Washington, DC, USA, 2007. IEEE Computer Society.

[13] Itsuki Noda, Hitoshi Matsubara, Kazuo Hiraki, and Ian Frank. Soccer server: A tool for research on multiagent systems. *Applied Artificial Intelligence*, 12:233–250, 1998.

[14] G. A. Rummery and M. Niranjan. On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG-RT 116, Cambridge Uni. Eng. Dept., 1994.

[15] Kenneth 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.

[16] Kenneth O. Stanley, David B. D'Ambrosio, and Jason Gauci. A hypercube-based indirect encoding for evolving large-scale neural networks. *Artificial Life*, 15, 2009.

[17] Kenneth O. Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10:99–127, 2002.

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

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

[20] Peter Stone, Gregory Kuhlmann, Matthew E. Taylor, and Yaxin Liu. Keepaway soccer: From machine learning testbed to benchmark. In *Robot Soccer World Cup IX*, pages 93–105. Springer Verlag, 2006.

[21] Peter Stone and Richard S. Sutton. Scaling reinforcement learning to robocup soccer. In *The 18th Int. Conf. on ML*, pages 537–544, New York, NY, June 2001, ACM.

[22] Peter Stone and Richard S. Sutton. Keepaway soccer: A machine learning testbed. In *Robot Soccer World Cup V*, pages 214–223, London, UK, 2002. Springer-Verlag.

[23] Peter Stone, Richard S. Sutton, and Gregory Kuhlmann. Reinforcement learning for RoboCup-soccer keepaway. *Adaptive Behavior*, 13(3):165–188, 2005.

[24] Peter Stone and Manuela Veloso. Layered learning. In Ramon López de Mántaras and Enric Plaza, editors, *Proc. of the 11th European Conf. on ML*, pages 369–381. Springer Verlag, Barcelona,Spain, May/June 2000.

[25] Richard S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 8*, pages 1038–1044. MIT Press, 1996.

[26] Erik Talvitie and Satinder Singh. An experts algorithm for transfer learning. In *Proc. of the 10th Int. Joint Conf. on AI*, pages 1065–1070, 2007.

[27] Matthew E. Taylor and Peter Stone. Cross-domain transfer for reinforcement learning. In *Proc. of the 24th Int. Conf. on ML*, pages 879–886, New York, NY, USA, 2007. ACM.

[28] Matthew E. Taylor, Peter Stone, and Yaxin Liu. Transfer learning vis inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, 1(8):2125–2167, September 2007.

[29] Matthew E. Taylor, Shimon Whiteson, and Peter Stone. Comparing evolutionary and temporal difference methods in a reinforcement learning domain. In *Proc. of the Genetic and Ev. Comp. Conf.*, pages 1321–1328, New York, NY, July 2006. ACM Press.

[30] Matthew E. Taylor, Shimone Whiteson, and Peter Stone. Transfer via intertask mappings in policy search reinforcement learning. In *Proc. of the AAMAS Conf.*, New York, NY, May 2007, ACM Press.

[31] A. M. Turing. The Chemical Basis of Morphogenesis. *Royal Society of London Philosophical Transactions Series B*, 237:37–72, August 1952.

[32] Shimon Whiteson. Improving reinforcement learning function approximators via neuroevolution. In *Proc. of the 4th Int. Joint Conf. on AAMAS*, pages 1386–1386, New York, NY, USA, 2005. ACM.

[33] Shimon Whiteson and Daniel Whiteson. Stochastic optimization for collision selection in high energy physics. In *Proc. of the 19th Annual Innov. Apps. of AI Conf.*, Vancouver, Canada, July 2007. AAAI Press.