On the Cross-Domain Reusability of Neural Modules for General Video Game Playing

Alex Braylan Mark Hollenbeck Elliot Meyerson Risto Miikkulainen
{braylan,mhollen,ekm,risto}@cs.utexas.edu
The University of Texas at Austin

Abstract

We consider a general approach to knowledge transfer in which an agent learning with a neural network adapts how it reuses existing networks as it learns in a new domain. Networks trained for a new domain are able to improve performance by selectively routing activation through previously learned neural structure, regardless of how or for what it was learned. We present a neuroevolution implementation of the approach with application to reinforcement learning domains. This approach is more general than previous approaches to transfer for reinforcement learning. It is domain-agnostic and requires no prior assumptions about the nature of task relatedness or mappings. We analyze the method's performance and applicability in high-dimensional Atari 2600 general video game playing.

1 Introduction

The ability to generally apply any and all available previously learned knowledge to new tasks is a hallmark of general intelligence. *Transfer learning* is the process of reusing knowledge from previously learned *source* tasks to bootstrap learning of *target* tasks. For reinforcement learning (RL) agents, transfer is particularly important, as previous experience can help to efficiently explore new environments. Knowledge acquired during previous tasks also contains information about the agent's task-independent decision making and learning dynamics, and thus can be useful even if the tasks seem completely unrelated.

Existing approaches to transfer learning for reinforcement learning have successfully demonstrated transfer of varying kinds of knowledge [Taylor and Stone, 2009], but they tend to make two fundamental assumptions that restrict their generality: (1) some sort of a priori human-defined understanding of how tasks are related, (2) separability of knowledge extraction and target learning. The first assumption limits the applicability of the approach by restricting its use only to cases where the agent has been provided with this additional relational knowledge, or, if it can be learned, cases where task mappings are useful. The second assumption implies further expectations about what knowledge will be useful and how it should be incorporated *before* learning on the target task begins, preventing the agent from adapting the way it uses source knowledge as it gains information about the target domain.

We consider General Reuse Of Modules (GReuseOM), a general neural network approach to transfer learning that avoids both of these assumptions, by augmenting the learning process to allow learning networks to selectively route through existing neural modules (source networks) as they simultaneously develop new structure for the target task. Unlike previous work, which has dealt with mapping task variables between source and target, GReuseOM is task-independent, in that no knowledge about the structure of the source task or even knowledge about where the network came from is required for it to be reused. Instead of using mappings between task-spaces to facilitate transfer, it searches directly for mappings in the solution space, that is, connections between existing source networks and the target network. GReuseOM is motivated by studies that have shown in both naturally occurring complex networks [Milo *et al.*, 2002] and artificial neural networks [Swarup and Ray, 2006] that certain network structures repeat and can be useful across domains, without any context for how exactly this structure should be used. We are further motivated by the idea that neural resources in the human brain are reused for countless purposes in varying complex ways [Anderson, 2010].

In this paper, we present an implementation of GReuseOM based on the Enforced Subpopulations (ESP) neuroevolution framework [Gomez and Miikkulainen, 1997]. We validate our approach first in a simple boolean logic domain, then scale up to the Atari 2600 general game playing domain. In both domains, we find that GReuseOM-ESP improves learning overall, and tends to be most useful when source and target networks are more complex. In the Atari domain, we show that the effectiveness of transfer coincides with an intuitive high-level understanding of game dynamics. This demonstrates that even without traditional transfer learning assumptions, successful knowledge transfer via general reuse of existing neural modules is possible and useful for RL. In principle, our approach and implementation naturally scale to transfer from an arbitrary number of source tasks, which points towards a future class of GReuseOM agents that accumulate and reuse knowledge throughout their lifetimes across a variety of diverse domains.

The remainder of this paper is organized as follows: Section 2 provides background on transfer learning and related work, Section 3 describes our approach in detail, Section 4 analyzes results from experiments we have run with this approach, and Section 5 discusses the implications of these results and motivations for future work.

2 Background

Transfer learning encompasses machine learning techniques that involve reusing knowledge across different domains and tasks. In this section we review existing transfer learning methodologies and discuss their advantages and shortcomings to motivate our approach. We take the following two definitions from [Sinno and Yang, 2010]. A *domain* is an environment in which learning takes place, characterized by the input and output space. A *task* is a particular function from input to output to be learned. In sequential-decision domains, a task is characterized by the values of sensory-action sequences corresponding to the pursuit of a given goal. A taxonomy of types of knowledge that may be transferred are also enumerated in [Sinno and Yang, 2010]. As our approach reuses the structure of existing neural networks, it falls under 'feature representation transfer'.

2.1 Transfer Learning for RL

Reinforcement learning (RL) domains are often formulated as Markov decision processes in which the state space comprises all possible observations, and the probability of an observation depends on the previous observation and an action taken by a learning agent. However, many real world RL domains are non-Markovian, including many Atari 2600 games.

Five dimensions for characterizing the generality and autonomy of algorithms for transfer learning in RL are given in [Taylor and Stone, 2009]: (1) restrictions on how source and target task can differ; (2) whether or not *mappings* between source and target state and action variables are available to assist transfer; (3) the form of the knowledge transferred; (4) restrictions on what classes of learning algorithms can be used in the source and/or target tasks; (5) whether or not the algorithm autonomously selects which sources to reuse.

Some of the most general existing approaches to transfer for RL automatically learn task mappings, so they need not be provided beforehand, e.g., [Taylor *et al.*, 2007; 2008; Talvitie and Singh, 2007]. These approaches are general enough to apply to any reinforcement learning task, but as the state and action spaces become large they become intractable due to combinatorial blowup in the number of possible mappings. These approaches also rely on the assumption that knowledge for transfer can be extracted based on mappings between state and action variables, which may miss useful internal structure these mappings cannot capture.

2.2 General Neural Structure Transfer

There are existing algorithms similar to our approach in that they enable general reuse of existing neural structure. They can apply to a wide range of domains and tasks in that they automatically select source knowledge and avoid inter-task mappings. Knowledge-Based Cascade Correlation [Shultz and Rivest, 2001] uses a technique based on cascade correlation to build increasingly complex networks by inserting source networks chosen by how much they reduce error. Knowledge Based Cascade Correlation is restricted in that it is only designed for supervised learning, as the source selection depends heavily on an immediate error calculation. Also, connectivity between source and target networks is limited to the input and output layer of the source. Subgraph Mining with Structured Representations [Swarup and Ray, 2006] creates sparse networks out of primitives, or commonly used sub-networks, mined from a library of source networks. The subgraph mining approach depends on a computationally expensive graph mining algorithm, and it tends to favor exploitation over innovation and small primitives rather than larger networks as sources.

Our approach is more general in that it can be applied to unsupervised and reinforcement learning tasks, and makes fewer a priori assumptions about what kind of sources and mappings should work best. Although we only consider an evolutionary approach in this paper, GReuseOM should be extensible to any neural network-based learning algorithm.

3 Approach

This section introduces the general idea behind GReuseOM, then provides an overview of the ESP neuroevolution framework, before describing our particular implementation: GReuseOM-ESP.

3.1 General Reuse Of Modules (GReuseOM)

The underlying idea is that an agent learning a neural network for a target task can selectively reuse knowledge from existing neural modules (source networks) while simultaneously developing new structure unique to a target task. This attempts to balance reuse and innovation in an integrated architecture. Both source networks and new hidden nodes are termed *recruits*. Recruits are added to the target network during the learning process. Recruits are adaptively incorporated into the target network as it learns connection parameters from the target to the recruit and from the recruit to the target. All internal structure of source networks is *frozen* to allow learning of connection parameters to remain consistent across recruits. This forces the target network to transfer learned knowledge, rather than simply overwrite it. Connections to and from source networks can, in the most general case, connect to any nodes in the source and target, minimizing assumptions about what knowledge will be useful.

A GReuseOM network or *reuse network* is a 3-tuple $\mathcal{G} = (M, S, T)$ where M is a traditional neural network (feedforward or recurrent) containing the new nodes and connections unique to the target task, with input and output nodes corresponding to inputs and outputs defined by the target domain; S is a (possibly empty) set of pointers to recruited source networks $\mathcal{S}_1, ..., \mathcal{S}_k$; and T is a set of weighted *transfer connections* between nodes in M and nodes in source networks, that is, for any connection $((u, v), w) \in T$, $(u \in M \wedge v \in \mathcal{S}_i) \vee (u \in \mathcal{S}_i \wedge v \in M)$ for some $0 \leq i \leq k$. This construction strictly extends traditional neural networks so that each $\mathcal{S}_i$ can be a traditional neural network or a reuse network of its own. When $\mathcal{G}$ is evaluated, we evaluate only the network induced by directed paths from inputs of M to outputs of M, including those which pass through some $\mathcal{S}_i$ via connections in T. Before each evaluation of $\mathcal{G}$, all recruited source network inputs are set to 0, since at any given time the agent is focused only on performing the current target task. The parameters to be learned are the usual parameters of M, along with the contents of S and T. The internal parameters of each $\mathcal{S}_i$ are frozen in that they cannot be rewritten through $\mathcal{G}$.

The motivation for this architecture is that if the solution to a source task contains *any* information relevant to solving a target task, then the neural network constructed for the source task will contain *some* structure (subnetwork or module) that will be useful for a target network. This has been shown to be true in naturally occurring complex networks [Anderson, 2010], as well as cross-domain artificial neural networks [Swarup and Ray, 2006]. Unlike in the subgraph mining approach to neural structure transfer [Swarup and Ray, 2006], this general formalism makes no assumptions as to what subnetworks actually will be useful. One perspective that can be taken with this approach is that a lifelong learning agent maintains a system of interconnected neural modules that it can potentially reuse at any time for a new task. Even if existing modules are unlabeled, they may still be useful, simply due to the fact that they contain knowledge of how the agent can successfully learn. Furthermore, recent advances in reservoir computing [Lukoševičius and Jaeger, 2009] have demonstrated the power of using large amounts of frozen neural structure to facilitate learning of complex and chaotic tasks.

The above formalism is general enough to allow for an arbitrary number of source networks and arbitrary connectivity between source and target. In this paper, to validate the approach and simplify analysis, we use at most one source network and only allow connections from target input to source hidden layer and source output layer to target output. This is sufficient to show that the implementation can successfully reuse hidden source features, and analyze the cases in which transfer is most useful. Future refinements are discussed in Section 5. The current implementation, described below, is a neuroevolution approach based on ESP.

3.2 Enforced Subpopulations (ESP)

Enforced Sub-Populations (ESP) [Gomez and Miikkulainen, 1997] is a neuroevolution technique in which different elements of a neural network are evolved in separate *subpopulations* rather than evolving the whole network in a single population. ESP has been shown to perform well an a variety of reinforcement learning tasks, e.g., [Gomez and Miikkulainen, 1997; 2003; Gomez and Schmidhuber, 2005; Miikkulainen *et al.*, 2012; Schmidhuber *et al.*, 2007]. In standard ESP, each hidden neuron is evolved in its own subpopulation. Recombination occurs only between members of the same subpopulation, and mutants in a subpopulation derive only from members of that subpopulation. The genome of each individual in a subpopulation is a vector of weights corresponding to the weights of connections from and to that neuron, including node bias. In each generation, networks to be evaluated are randomly constructed by inserting one neuron from each subpopulation. Each individual that participated in the network receives the fitness achieved by that network.

When fitness converges, i.e., does not improve over several consecutive generations, ESP makes use of *burst phases*. In initial burst phases each subpopulation is repopulated by mutations of the single best neuron ever occuring in that subpopulation, so that it reverts to searching a δ-neighborhood around the best solution found so far. If a second consecutive burst phase is reached, i.e., no improvements were made since the previous burst phase, a new neuron with a new subpopulation may be added [Gomez, 2003].

3.3 GReuseOM-ESP

We extend the idea of enforced sub-populations to transfer learning via GReuseOM networks. For each reused source network $\mathcal{S}_i$ the transfer connections in T between $\mathcal{S}_i$ and M evolve in a distinct subpopulation. At the same time new hidden nodes can be added to M and evolve within their own subpopulations in the manner of standard ESP. In this way, the integrated evolutionary process simultaneously searches the space for how to reuse each potential source network and how to innovate with each new node. Specifically, the GReuseOM-ESP architecture (Figure 1) is composed of the following elements:

- A pool of potential source networks. In the experiments in this paper, each target network reuses at most one source at a time.
- *Transfer genomes* defining a list of transfer connections between the source and target networks. Each potential source network in the pool has its own subpopulation for evolving transfer genomes between it and the target network. Each connection in T is contained in some transfer genome. In our experiments, the transfer connections included are those such that the target's inputs are fully connected to the source's hidden layer, and the source's outputs are fully connected into the target's outputs. Therefore, the transfer genome only encodes the weights of these cross-network connections.
- A burst mechanism that determines when innovation is necessary based on a recent history of performance improvement. New hidden recruits (source networks or new single nodes) added during the burst phase evolve within their own subpopulations in the manner of classic ESP.

All hidden and output neurons use a hyperbolic tangent activation function. Networks include a single hidden layer, and can include self loops on hidden nodes; they are otherwise feedforward. The particulars of the genetic algorithm in our implementation used to evolve each subpopulation mirror those described in [Gomez, 2003]. This algorithm has been shown to work well within the ESP framework, though any evolutionary algorithm could potentially be substituted in its place.

4 Experiments

We evaluate GReuseOM-ESP on two domains: a simple n-bit parity domain mirroring that used to evaluate knowledge

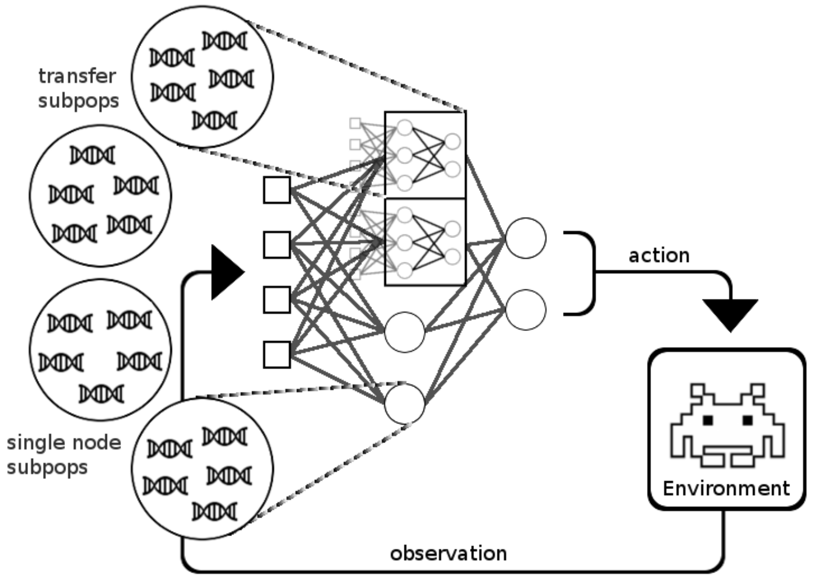


Figure 1: The GReuseOM-ESP architecture. Reused subnetworks of sources are boxed. Edges between input and source and between source and output denote full connectivity between these layers. The genome in each subpopulation encodes weight information for the connections from and to the corresponding recruit.

transfer in [Swarup and Ray, 2006], and the more complex Atari 2600 video game playing domain. In both domains, we first train *scratch* networks that do not reuse existing networks, that is, S is the empty set. We then reuse each scratch network in training GReuseOM networks for different tasks. We compare performance between scratch and transfer, and between source-target setups. Results demonstrate the ability of GReusOM-ESP to selectively reuse source structure.

4.1 N-Bit Parity

GReuseOM-ESP was initially evaluated under the boolean logic domain using N-bit parity. The N-bit parity problem has a long-standing history serving as a benchmark for basic neural network performance. The N-bit parity function is the mapping defined on N-bit binary vectors that returns 1 if the sum of the N binary values in the vector is odd, and 0 otherwise. This function is deceptively difficult for neural networks to learn since a change in any single input bit will alter the output. Although N-bit parity is not fully *cross-domain* in the stronger sense for which our approach applies, the input feature space does differ as N differs, and it is useful for validation of the approach and connection with previous work.

Performance is measured in number of generations to find a network that solves N-bit parity within $\epsilon = 0.1$ mean squared error. In this experiment, networks were trained from scratch with ESP for $N = [2, 3, 4]$. Then, each of these networks was used as a source network for each N-bit parity target domain with $N = [3, 4, 5]$. ESP, without transfer, was used as a control condition for each target task. A total of 10 trials were completed for each condition.

In this experiment, transfer learning was able to outperform learning from scratch for all three target tasks when using some source task. For 3-bit and 4-bit parity, transfer

	Source	None	2-bit par	3-bit	4-bit
Target	3-bit par	309.5	202	167.5	**158**
	4-bit par	339	**192.5**	308	311
	5-bit par	626	780	720.5	**542**

Table 1: Median number of generations for task completion for all N-bit parity source-target setups.

learning always outperformed learning from scratch for all three possible sources. For the more complex 5-bit parity target task, transfer from the 4-bit network outperformed learning from scratch, while transfer from the simpler tasks did not. This may be due to the significantly greater complexity required for 5-bit parity over 2- or 3-bit parity. The limited frozen structure may become a burden to innovation after the initial stages of evolution. The more complex 4-bit parity networks have more structure to select from, and thus may assist in innovation over a longer time frame.

4.2 Atari 2600 Game Playing

Our next experiment evaluated game playing performance in the Atari 2600 game platform using the Arcade Learning Environment (ALE) simulator [Bellemare *et al.*, 2013]. This domain is particularly popular for evaluating RL techniques, as it exhibits sufficient complexity to challenge modern approaches, contains non-markovian properties, and entertained a generation of human video game players. We used GReuseOM-ESP to train agents to play eight games (Asterix, Bowling, Boxing, Breakout, Freeway, Pong, Space Invaders, and Seaquest) both from scratch and using transferred knowledge from existing game-playing source networks. Neuroevolution techniques are quite competitive in the Atari 2600 domain [Hausknecht *et al.*, 2013], and ESP in particular has yielded state-of-the-art performance for several games [Braylan *et al.*, 2015].

Each source network was trained from scratch on a game using standard ESP (GReuseOM-ESP with an empty reuse set). Each source network was then used by a target network for an evolutionary run for each other game. Each run lasted 200 generations with 100 evaluations per generation. Each individual i achieves some score $i(g)$ in its game g. Let $min(g)$ be the min over all max scores achieved in a single generation by any run of g. Let the *fitness* of i be $i(g) - min(g)$. This ensures that fitness is always positive (in both boxing and pong, raw scores can be negative). The fitness of an evolutionary run at a given generation is the highest fitness achieved by an individual by that generation.

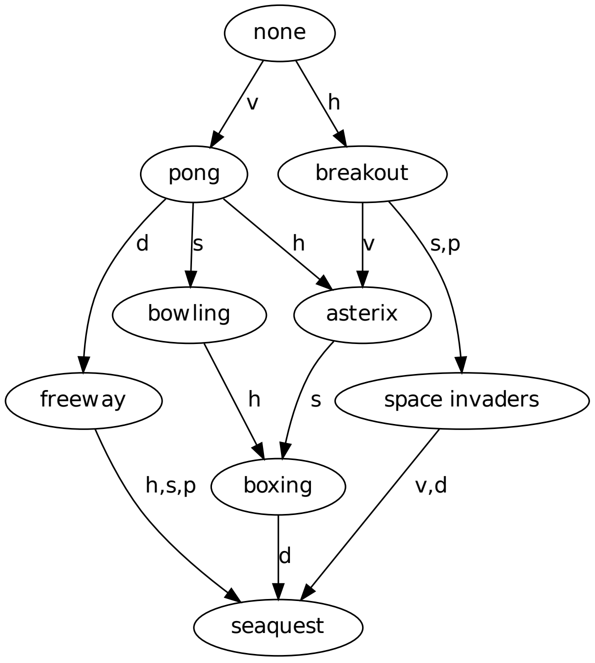
We ran a total of 176 trials split across all possible setups: training using each other game as a source, and training from scratch. We use the ϵ-repeat action approach as suggested in [Hausknecht and Stone, 2015] to make the environment stochastic in order to disable the algorithm from finding loopholes in the deterministic nature of the simulator. We use the recommended $\epsilon = 0.25$ [1]. Parameters were selected based on their success with standard ESP.

To interface with ALE, the output layer of each network consists of a 3x3 substrate representing the 9 directional movements of the Atari joystick in addition to a single node representing the Fire button. The input layer consists of a series of object representations manually generated as previously described in [Hausknecht *et al.*, 2013], where the location of each object on the screen is represented in an 8x10 input substrate corresponding to the object's class. The number of object classes for the games used in our experiments vary between one and four. Although object representations are used in these experiments, pixel-level vision could also be learned from scratch below the neuroevolution process, e.g., via convolutional networks, as in [Koutník *et al.*, 2014].

Domain Characterization

Each game can be characterized by generic binary features that determine the requirements for successful game play, in order to place the games within a unified framework. We use binary features based on the existence of the following: (1) horizontal movement (joystick left/right), (2) vertical movement (joystick up/down), (3) shooting (fire button); (4) delayed rewards; and (5) the requirement of long-term planning. Intuitively, more complex games will possess more of these qualities. The partial ordering of games by complexity defined by these features is shown in Figure 2. The assignment of features (1), (2) and (3) is completely defined based on game interface [Bellemare *et al.*, 2013]. Freeway and Seaquest are said to have *delayed rewards* because a high score can only be achieved by long sequences of rewardless behavior. Only Space Invaders and Seaquest were deemed to require long-term planning [Mnih *et al.*, 2013], since the long-range dynamics of these games penalize reflexive strategies, and as such, agents in these games can perform well with a low decision-making frequency [Braylan *et al.*, 2015]. Aside from their intuitiveness, these features are validated below based on their ability to characterize games by complexity and predict transferibility. For a simple metric of complexity, let $cmplx(g)$ be the number of the above features game g exhibits.

[1] https://github.com/mgbellemare/Arcade-Learning-Environment/tree/dev



	vert	horiz	shoot	delay	plan
pong	■				
breakout		■			
asterix	■	■			
bowling	■		■		
freeway	■			■	
boxing	■	■	■		
space invaders		■	■		■
seaquest	■	■	■	■	■

Figure 2: Vector of features for each game (indicated in black) and lattice of games ordered by features. Every path from *none* to g contains along its edges each complexity feature of g a exactly once. Features: v = vertical movement, h = horizontal movement, s = shooting, d = delayed reward, p = long-term planning.

Atari 2600 Results

There are many possible approaches to evaluating success of transfer [Taylor and Stone, 2009]. For comparing performance *across* games, we focus on *time to threshold*. To minimize threshold bias, for each game we chose the threshold to be the min of the max fitness achieved across all trials. Given this threshold, the average time to threshold in terms of generations may be vastly different, depending on the average learning curve of each game. These learning curves are quite irregular, as illustrated in Figure 4. For each game we measure time in terms of percent of average time to threshold, and the *success rate* is the proportion of trials that have achieved the threshold by that time.

Figure 3 plots success over time for different groups of trials. The leftmost plot compares the success rate of all transfer

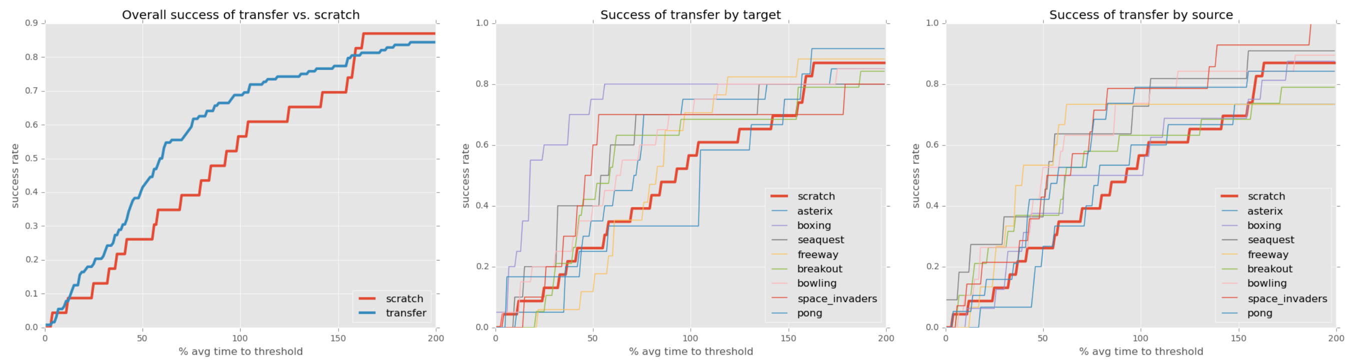


Figure 3: Success rate (proportion of trials that have reached the target threshold) by percent of average number of generations to threshold of target game with trials (allowing comparisons across games with different average times to threshold) grouped by (1) scratch vs. transfer, (2) target game, (3) source game.

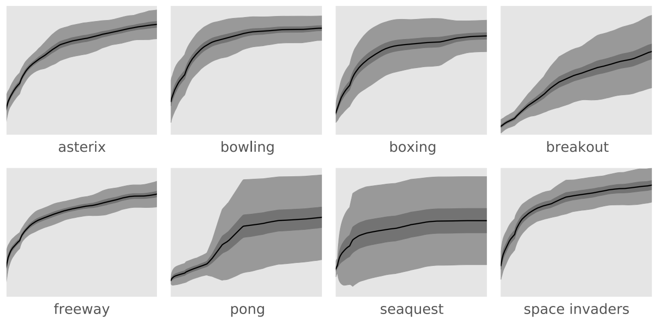


Figure 4: Distributions of fitness for each game by generation over all trials. Mean (black), standard error (dark gray) and standard deviation (light gray) are shown at each generation.

game g	cmplx(g)	deg(g)	deg$^-(g)$	deg$^+(g)$
seaquest	5	8	4	4
space invaders	3	7	4	3
boxing	3	6	4	2
bowling	2	5	3	2
asterix	2	5	2	3
freeway	2	4	2	2
pong	1	4	1	3
breakout	1	1	0	1

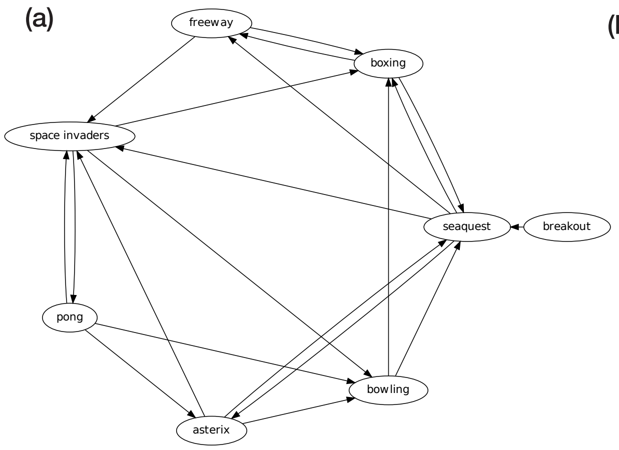
Table 2: A total ordering of games by complexity score and degree (total, in (-), and out (+)) in the transferability digraph with edge cutoff 0.5 (Figure 5(a)).

trials to scratch trials. It shows what we would expect from transfer overall: networks that reuse frozen structure from previous games are able to take advantage of that structure to bootstrap learning. This works initially, but eventually scratch catches up, as it becomes more difficult to innovate with a single frozen structure. When trials are grouped by target (middle pane), we can see that some games are better targets for transfer than others. As demonstrated in Figure 3, more complex games (with respect to our game features) are generally better targets than less complex. It is less clear what we can draw from grouping trials by source (right pane). There is a tighter spread than with targets, though there may still be a tendency towards more complex games being better sources. This may be counter-intuitive, as we might expect simpler games to be easier to reuse. However, more complex games have networks with more complex structure from which a target network can, through the evolutionary process, select some useful subnetwork that fits its needs. Similarly, a complex domain will be more likely to be a good target, since it requires a wider variety of structure to be successful, so sources have a higher chance of satisfying *some* of that requirement.

For comparing performance *within* a target game, we need not resort to threshold normalization, and can instead focus on raw max fitness. For reference, average and best fitness for both transfer and scratch are given in Table 3. Note that previously published approaches to Atari game-playing use fully deterministic environments, making direct score comparisons difficult (see [Braylan *et al.*, 2015] for a comparison of ESP to other approaches in deterministic environments).

The *transfer effectiveness* of a source-target setup is the log ratio between its average max fitness and the average max fitness of that game from scratch. The digraphs in Figure 5 each contain the directed edge from g_1 to g_2 only when transfer effectiveness is above a specified threshold. These graphs indicate that the more complex games serve a more useful role in transfer than less complex. Consider the total ordering of games by $cmplx(g)$ given in Table 2. This ordering corresponds exactly to that induced by the degree sequence (by both total degree and in-degree) [Diestel, 2005] of the graph with edge cutoff 0.5. However, for out-degree, the correlation with respect to the ordering is less clear. This reflects Figure 3, in which there is more spread in success when grouped by target (in-degree) vs. source (out-degree).

We see that we can predict the transfer effectiveness by the feature characterizations we provided. The feature characterizations allow us to consider all trials in the same feature space. A linear regression model trained on a random half of the setups yielded weight coefficients for the source and target features that successfully predicted the transfer effectiveness of setups in the test set (Figure 6). The slope was found to be statistically significant with a p-value of 0.01. The most



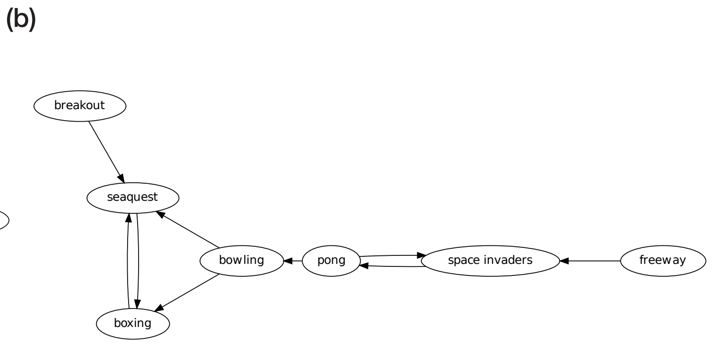


Figure 5: Transferability graphs illustrating the most successful source-target pairs. Each graph includes a directed edge from g_1 to g_2 $\iff$ the transfer effectiveness (defined above) for g_2 reusing g_1 is greater than (a) 0.5, and (b) 1.0, respectively.

game	$min(g)$	best$_t$	best$_s$	avg$_t$	avg$_s$
seaquest	160	1510	300	475.0	262.0
space invaders	310	1520	1320	1076.0	1160.0
boxing	-12	111	107	98.6	104.1
bowling	30	237	231	219.9	201.9
asterix	650	3030	2150	1989.0	2016.7
freeway	21	13	11	10.7	10.7
pong	-21	42	42	21.8	20.3
breakout	0	51	37	25.4	31.3

Table 3: For both transfer (t) and scratch (s) runs, average fitness and best fitness of GReusOM-ESP.

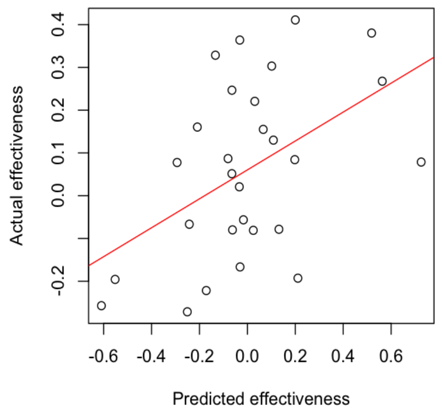


Figure 6: Feature-based linear prediction versus actual transfer effectiveness on out-of-sample setups

significant features were vertical movement and long-term planning in the source domain, with respective coefficients of 0.73 and 0.89. The ability to use the game features to predict transfer effectiveness can be used to inform source selection. It is also encouraging that the effectiveness of transfer with GReuseOM-ESP correlates with a high-level intuition of inter-game dynamics.

5 Discussion

Our results show that GReuseOM-ESP, an evolutionary algorithm for general transfer of neural network structure, can improve learning in both boolean logic and Atari game playing by reusing previously developed knowledge. However, we find that the improvement in learning performance in the target domain depends heavily on the source network. Some source-target pairs do not consistently outperform training from scratch, indicating negative transfer from that source. This highlights the importance of source selection in transfer learning.

Specifically with the Atari game playing domain, we observe an issue of source knowledge *quality*. Some of the source networks that were trained from scratch do relatively well on games whereas others do not. One problem is that the measure of knowledge in source networks is ill-defined. As alluded to in [Taylor *et al.*, 2007], there could be an optimal point in a source's training at which to transfer knowledge to a target, after which the source network has encoded knowledge too specific to its own task, which does not generalize as well to other tasks, and makes useful knowledge difficult to extract. Future analysis will investigate topological regularities of source networks and transfer connections, to further address what and how knowledge is successfully reused.

Another future area of work will involve increasing the flexibility in the combined architecture by 1) relaxing the requirement for all transfer connections to be input-to-hidden and output-to-output, and 2) allowing deeper architectures for the source and target networks. This will promote reuse of subnetworks of varying depth and flexible positioning of modules. However, as networks become large and plentiful, maintaining full connectivity between layers will become intractable, and enforcing sparsity will be necessary.

Having shown that our algorithm works with certain target-source pairs, a next step will involve pooling multiple candidate sources and testing GReuseOM-ESP's ability to exploit the most useful ones. GReuseOM-ESP extends naturally to learning transfer connections for multiple sources simultaneously. By starting with limited connectivity and adding connections to sources that show promise (while removing connections from ones that are not helping), adaptive multi-source selection may be integrated into the evolutionary process. Methods for adapting this connectivity online have yet to be developed.

Although our initial experiments only scratched the surface, they are encouraging in that they show general transfer of neural structure is possible and useful. They have also helped us characterize the conditions under which transfer may be useful. It will be interesting to investigate whether the same principles extend to other general video game playing domains, such as [Perez *et al.*, 2015; Schaul, 2013]. This should help us better understand how subsymbolic knowledge can be recycled indefinitely across diverse domains.

6 Conclusion

We consider a framework for general transfer learning using neural networks. This approach minimizes a priori assumptions of task relatedness and enables a flexible approach to adaptive learning across many domains. In both the Atari 2600 and N-bit parity domains, we show that a specific implementation, GReuseOM-ESP is able to successfully boost learning by reusing neural structure across disparate tasks. The success of transfer is shown to correlate with intuitive notions of task dynamics and complexity. Our results indicate that general neural reuse – a staple of biological systems – can effectively assist agents in increasingly complex environments.

Acknowledgments This research was supported in part by NSF grant DBI-0939454, NIH grant R01-GM105042, and an NPSC fellowship sponsored by NSA.

References

[Anderson, 2010] M. L. Anderson. Neural reuse: A fundamental organizational principle of the brain. *Behavioral and Brain Sciences*, 33:245–266, 2010.

[Bellemare *et al.*, 2013] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *JAIR*, 47:253–279, 2013.

[Braylan *et al.*, 2015] A. Braylan, M. Hollenbeck, E. Meyerson, and R. Miikkulainen. Frame skip is a powerful parameter for learning to play atari. In *AAAI Workshop on Learning for General Competency in Video Games*, 2015.

[Diestel, 2005] R. Diestel. *Graph Theory*. Springer, 2005. p. 278.

[Gomez and Miikkulainen, 1997] F. J. Gomez and R. Miikkulainen. Incremental evolution of complex general behavior. *Adaptive Behavior*, (5):317–342, 1997.

[Gomez and Miikkulainen, 2003] F. J. Gomez and R. Miikkulainen. Active guidance for a finless rocket using neuroevolution. In *Proc. of GECCO '03*, pages 2084–2095, 2003.

[Gomez and Schmidhuber, 2005] F. J. Gomez and J. Schmidhuber. Co-evolving recurrent neurons learn deep memory pomdps. In *Proc. of GECCO '05*, pages 491–498, 2005.

[Gomez, 2003] F. J. Gomez. Robust non-linear control through neuroevolution. Technical report, UT Austin, 2003.

[Hausknecht and Stone, 2015] M. Hausknecht and P. Stone. The impact of determinism on learning atari 2600 games. In *AAAI Workshop on Learning for General Competency in Video Games*, 2015.

[Hausknecht *et al.*, 2013] M. Hausknecht, J. Lehman, R. Miikkulainen, and P. Stone. A neuroevolution approach to general atari game playing. In *Comp. Intel. and AI in Games*, 2013.

[Koutník *et al.*, 2014] J. Koutník, J. Schmidhuber, and F. J. Gomez. Online evolution of deep convolutional network for vision-based reinforcement learning. In *From Animals to Animats 13*. 2014.

[Lukoševičius and Jaeger, 2009] M. Lukoševičius and H. Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.

[Miikkulainen *et al.*, 2012] R. Miikkulainen, E. Feasley, L. Johnson, I. Karpov, P. Rajagopalan, A. Rawal, and W. Tansey. Multiagent learning through neuroevolution. In *Advances in Computational Intelligence*, pages 24–46. 2012.

[Milo *et al.*, 2002] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

[Mnih *et al.*, 2013] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. arXiv:1312.5602, 2013.

[Perez *et al.*, 2015] D. Perez, S. Samothrakis, J. Togelius, T. Schaul, S. Lucas, A. Couetoux, J. Lee, C. Lim, and T. Thompson. The 2014 general video game playing competition. *IEEE Trans. on Computational Intelligence and AI in Games*, (99), 2015.

[Schaul, 2013] T. Schaul. A video game description language for model-based or interactive learning. In *Proc. of IEEE Computational Intelligence in Games (CIG '13)*, pages 1–8, Aug 2013.

[Schmidhuber *et al.*, 2007] J. Schmidhuber, D. Wierstra, M. Gagliolo, and F. J. Gomez. Training recurrent networks by evolino. *Neural computation*, 19(3):757–779, 2007.

[Shultz and Rivest, 2001] T. R. Shultz and F. Rivest. Knowledge-based cascade-correlation: Using knowledge to speed learning. *Connection Science*, 13(1):43–72, 2001.

[Sinno and Yang, 2010] P. J. Sinno and Q. Yang. A survey on transfer learning. *Knowl. and Data Eng.*, (10):1345–1359, 2010.

[Swarup and Ray, 2006] S. Swarup and S. R. Ray. Cross-domain knowledge transfer using structured representations. In *Proc. of AAAI*, pages 506–511, 2006.

[Talvitie and Singh, 2007] E. Talvitie and S. Singh. An experts algorithm for transfer learning. In *Proc. of IJCAI '07*, pages 1065–1070, 2007.

[Taylor and Stone, 2009] M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, pages 1633–1685, 2009.

[Taylor *et al.*, 2007] M. E. Taylor, S. Whiteson, and P. Stone. Transfer via inter-task mappings in policy search reinforcement learning. In *Proc. of AAMAS '07*, pages 37:1–37:8, 2007.

[Taylor *et al.*, 2008] M. E. Taylor, G. Kuhlmann, and P. Stone. Autonomous transfer for reinforcement learning. In *Proc. of AAMAS '08*, pages 283–290, 2008.