Knowledge Transfer for Deep Reinforcement Agents in General Game Playing

Cameron McEwan and Michael Thielscher

UNSW Sydney, Sydney NSW 2052, Australia
`cameronpmcewan@gmail.com, mit@unsw.edu.au`

Abstract. Learning to master new games with nothing but the rules given is a hallmark of human intelligence. This ability has recently been successfully replicated in AI systems through a combination of Knowledge Representation, Monte Carlo Tree Search, and Deep Reinforcement Learning: Generalised AlphaZero [7] provides a method for building general game-playing agents that can learn any game describable in a formal specification language. We investigate how to boost the ability of deep reinforcement agents for general game playing by applying transfer learning for new game variants. Experiments show that transfer learning can significantly reduce the training time on variations of games that were previously learned, and our results further suggest that the most successful method is to train a source network that uses the guidance of multiple expert networks.

1 Introduction

In General Game Playing (GGP), agents are provided at runtime with the rules of a game they must learn [5]. Players are given formal specifications in the general Game Description Language (GDL), thereby addressing the problem that using single games for testing AI encourages niche programs that can only perform in one environment [6].

The most common algorithm employed by successful GGP agents is a variation of Monte Carlo Tree Search known as the *upper confidence bound on trees* (UCT) method [2]. UCT has been successfully combined with deep reinforcement learning in AlphaGo [16], to beat eighteen-time world champion Go player Lee Sedol, and later in AlphaZero [17], which learned to play three different board games from scratch to a superhuman level. The underlying principle of combining UCT with deep reinforcement learning was further developed to Generalised AlphaZero [7], where the two methods were further combined with knowledge representation and reasoning in order to build general game-playing agents that are capable of learning any game describable in GDL.

In both AlphaZero and Generalised AlphaZero, new games are learned from scratch. *Transfer learning* (TL) is a technique that aims to increase the efficiency of learning systems by transferring the knowledge gained from learning one task to a new but similar task [21]. Heretofore there has been limited success incorporating TL into agents for GGP [19,3,11], and no research that combines deep TL and GGP. Nevertheless, the method [3], which achieved positive transfer between completely different games, set the precedent that positive transfer in GGP is possible.

In this paper, we investigate and show how to boost the ability of deep reinforcement agents for general game playing by using transfer learning for new game variants. Our contribution is three-fold:

1. We develop a hierarchy of general game variations and strategies to best address each level by different transfer learning strategies.
2. We investigate the efficacy of two different methods of applying transfer learning in general game playing with deep reinforcement learning: using a network that has been previously trained on a single game versus a network that combines multiple expert networks, each previously trained on a different game.
3. Our analysis of the experimental results shows that guidance by multiple expert networks is the most successful of our methods to train a source network.

2 Background

General Game Playing. General game playing was developed to address the problem of niche game-playing AI programs that can only perform in certain environments [5,6]. The training process begins with players being provided with the rules of a game using the general Game Description Language GDL. Players have *startclock* seconds to perform initial training [5]. The International General Game Playing Competition typically has a 10 minute *startclock* [7]. In 2014, regular competition in the related field of *General Video Game Playing* (GVGAI) was established [13]. This competition shares GGP's basic premise that agents must be able to play previously unseen games. In contrast to GGP, GVGAI focuses on video games and may require agents to interpret states from an image feed and actions from a forward network to learn to play [13].

UCT and Reinforcement Learning. The most common algorithm employed by successful GGP agents is a variation on the *upper confidence bound on trees* (UCT) method [5]. This fundamental algorithm is a modification of *Monte Carlo Tree Search* (MCTS) to improve the trade-off between exploration and exploitation, by embedding an upper confidence bound of the expected reward in each node of the tree to guide search. UCT was successfully combined with *deep reinforcement learning* (RL) in AlphaGo to beat eighteen-time world champion Go player Lee Sedol [16]. RL is a field of machine learning that is concerned with agents learning to maximise reward. *Q-learning* is a common model-free RL algorithm. Its objective is to learn the function $Q(s, a)$ which represents the total discounted reward for taking action a in state s over an infinite time horizon. *Deep Q-learning* is a deep RL method for approximating $Q(s, a)$ when the input space is too large to efficiently solve. $Q(s, a)$ is replaced by a neural network, and triples of state, action and discounted reward are gathered and sampled to train the network. Deep Q-learning is employed by most state-of-the-art GVGAI players [15].

Generalised AlphaZero. AlphaZero employed deep reinforcement learning in combination with UCT to learn to play three different board games from scratch to a superhuman level [17]. *Generalised* AlphaZero (GAZ) combines this method with a *knowledge representation and reasoning* technique to handle arbitrary games specified in the formal game description language GDL [7]. Specifically, GAZ uses *propositional networks* (*propnets*) [14,6] to process the GDL game rules. Moreover, various network components were adjusted to remove the assumptions of AlphaZero to make it suitable for use in the GGP environment: GAZ employs a single neural network to generate

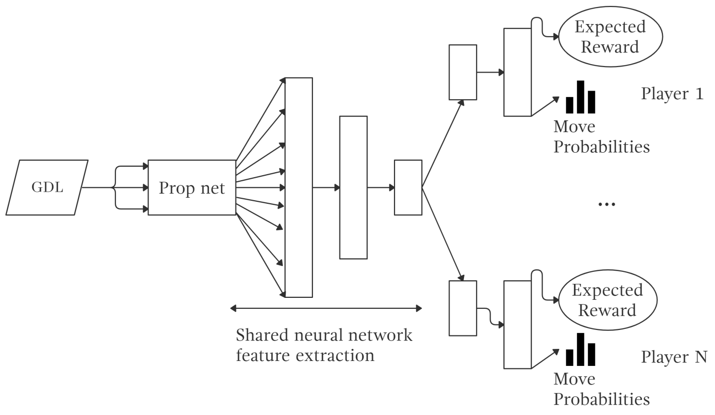

Fig. 1: Structure of a Generalised AlphaZero agent.

a better approximation for the UCT heuristic than MCTS alone. The network outputs an expected reward z_i and a probability distribution π_i over the action space. It then performs a number of self-play games running MCTS simulations to train the network. The MCTS simulations are conducted in the same way as standard UCT players with an adapted formula for the upper confidence bound. Upon reaching a leaf node, an option is expanded and the neural network is used to evaluate that node.

The structure of the GAZ network (shown in Figure 1) is influenced by three factors: the size of the propnet, the number of actions each player can take and the number of players. GAZ has undergone several experimental evaluations that demonstrate it is able to perform better than a UCT benchmark in many games, but requires significant training time [7,8]. When evaluated on Connect-4, Breakthrough, Babel and Pacman, the agent was successful in thwarting a UCT benchmark agent in all games but Babel. It took the GAZ agent between one and seven hours to beat UCT and 10 to 30 hours to complete a full training run on each game, significantly longer than the regular 10-minute *startclock* used in GGP competitions. This training time must therefore be significantly reduced for GAZ to succeed in GGP.

Transfer Learning. The training time of a neural network is influenced by two factors, the efficiency of a training round and the learning rate of the network. *Transfer Learning* (TL) is a technique that aims to increase the efficiency of learning systems by transferring the knowledge gained from learning one task to a similar new task [21]. An important aspect is the avoidance of so-called *negative transfer*, which occurs when transferring between inappropriate tasks leads to a slower, rather than faster, training time. The performance of TL agents is best evaluated by considering the time required to train on just the new task, or *target time* [20]. Other important metrics include: (1) the time to train the new and source agents, *total time*; (2) the improvement of initial performance, *jumpstart*; (3) the improvement of final performance, *asymptotic performance*; (4) the time it takes an agent to reach a predetermined threshold, *time to threshold*. *Deep TL* is the sub-field that concerns learning agents using deep RL methods similar to those used in GAZ and GVGAI. Deep TL works by copying the weights of a source network, or networks, directly to the target network. Several decisions are then made to help mitigate negative transfer. The source network is chosen and the layers of the network adjusted with the weights either fixed, reinitialised or allowed to be retrained [1]. When setting these transfer variables it is important to consider the difference between the source and target tasks. This is particularly challenging in game playing, where goals, actions, and domains can differ significantly between games. As a result, most research on TL in game play focuses on game variants and groups of similar games.

Previous Work on Transfer Learning for General Game Playing

In GGP, so far there has been limited success incorporating TL into agents. Past research proposed using a transfer hierarchy [19] and exploiting structural similarities in the game tree [3,11] to pair source and target tasks. The transfer hierarchy only succeeded transferring between game variants and is therefore of limited use in GGP [19]. Despite this, the tiered testing methodology, which proposed to first test on simple game variants before increasing the complexity of the variations, contributed a practical methodological framework for assessing transfer in games. The more successful method [3] achieved positive transfer between completely different games. This agent did not use deep RL, however, so its heuristic method was not considered in our agent design. Nevertheless, these results were important as they set the precedent that positive transfer in GGP is possible. A missing feature of all the TL approaches to GGP so far is a way to prevent or mitigate negative transfer [9].

General Video Game Playing agents that incorporate deep TL techniques into their architectures have had more success, achieving state-of-the-art results more efficiently when playing several games. Like GGP, the variety of games in GVGAI limits TL, and the most noteworthy research addresses this by generalising from multiple source tasks to mitigate or even eliminate negative transfer [12,22]. The use of single-network transfer and experiments with fixing, retraining, or re-initialising each network layer were successfully applied to PuckWorld and Snake [1]: By allowing all layers of an expert Puckworld network to retrain, the time to learn Snake could be significantly reduced. However, these results could not be replicated training Puckworld from Snake.

Multi-network transfer methods include the Actor-Mimic Network (AMN) [12] and Policy Transfer Framework (PTF) [22]. The seminal AMN established multi-network transfer learning in video game playing trading off *total time* for resilience to negative transfer. This powerful approach distils information from *multiple* expert networks into a generic network (the mimic) that is then used as the source for transfer. Transfer with the AMN was tested by training a mimic network on 13 source games and then using the result as the basis of transfer to learn 7 new games. Performance improved learning 6 of the 7 new games. AMN not fully addressing the problem of negative transfer, PTF further extends this approach by framing TL as an option learning problem in order to facilitate the elimination of negative transfer [22]. Still, the broad, and repeated, success of AMN makes it an ideal basis for adaptation to improve GGP agents.[1]

[1] Other methods in GVGAI focus on transferring learned skills about *visual* information [18,10]. Visual information is not relevant to GGP and therefore not considered in this paper.

3 Transfer Learning in Deep RL General Game-Playing Agents

Generalised AlphaZero training too slowly to compete in traditional GGP competition [7] motivates the use of transfer learning as a potential solution. Past work on GGP confirms that the barrier to successful knowledge transfer is prevention of negative transfer (cf. Section 2). In this section, we present a method based on the recent success of several multi-network deep Transfer Learning solutions to prevent negative transfer recent work in GVGAI [12]. We will describe two approaches and designs for deep reinforcement agents to make use Transfer Learning in General Game Playing: a *single* and a *multi-network* transfer learning method for GGP. We also distinguish between *simple* and *complex* transfer.

3.1 Simple Network Transfer in General Game Playing

Simple network transfer occurs when the source and target network have identical structures. Two Generalised AlphaZero networks (cf. Figure 1) have identical structures when the games they are learning have the same propnet size, number of actions and number of players. Because the structure of the two networks is identical, transfer happens by way of copying the weights layer-wise without any shape transformations. Simple Network Transfer has three variables: (1) the expertise of the source network, (2) weight transformations, and (3) the number of source networks.

Fully vs.˙ semi-trained source networks. We use the term *fully trained source networks* to describe expert networks that have been trained long enough to play at optimal level in a particular game. In contrast, *semi-trained networks* have only been trained for a fraction of the time, thus learning only some of the expert behaviour. Partial training could mitigate negative transfer, since the agent has less information to unlearn if all of the information in a fully trained source network is tailored to a specific game. Our agent design allows to use both fully and semi-trained agents interchangeably, with the hypothesis that fully trained agents will work best when transferring between the most similar game variants, and semi-trained agents will have better results in the least.

Weights. Weights can either be fixed; reinitialised, i.e. given random initial weights; or retrained, i.e. initialised from a pre-trained model [1]. Given the history of negative transfer in game playing and the difficultly of transferring between games, fixing weights is not an option for any layers in the agent designs. The intermediate layers are limited to retraining only, since reinitialising these layers would be equivalent to not performing transfer at all. The output layer for the move probabilities (cf. Figure 1) can be retrained or reinitialised. This is because this layer directly impacts the chance of an agent taking a particular action based on the probability distribution. If the action layer is allowed to retrain, the network will initially repeat the behaviour of the source network but is allowed to adjust behaviour over time. This should perform well when the old and the new game have similar goals, so that good moves in one game tend to be good moves in the other game too. On the other hand, reinitialising the action layer will cause the network to choose moves randomly in its first round of training, thus encouraging exploration of the game tree. It is predicted that this will improve performance when transferring to less similar games.

Number of source networks. In *single* network transfer, only one expert network is used. Past research shows this approach could be vulnerable to negative transfer. *Multi-network* transfer methods were developed to address this concern, creating a source network using the guidance of multiple expert networks. To train a source network, these multi-network transfer methods employ a multitask training method. The multi-task training method developed for our proposed agent design will be detailed in Section 3.3 below.[2]

3.2 Complex Network Transfer

If the source and target network have different structures, then additional variables for transfer must be introduced to map the structure of the source network to the target network. The additional variables required for what we consider *complex network transfer* depend on the factors that vary between the two games. The variables are: (1) input transformation, (2) action transformation, and (3) player transformation.

Input transformation is necessary when the size of the propnet changes. This has a cascading effect on the size of each of the layers in the shared feature extraction. As a result, the new network must be adapted to fit the source network within its feature extraction layers. To maintain a resemblance to the structure of Generalised AlphaZero, a gradual input transformation is proposed to slowly introduce the source network to the new network. In this transformation, layers of half or double the size are added between the input of the new network and the first layer of the source network to gradually converge their size. The aim of these layers is to help learn how to convert the input from the propnet to a set of features that approximate the input of the original game. The final transformation layer is then used as the input layer to a copy of the source network.

Action transformation is necessary when the set of actions is different between the two games. In our agent designs, if the actions are completely different then the action layers are reinitialised, whereas if the games share some actions then there are two options: reinitialisation or complete action mapping. Action mapping means to map the weights of the shared actions from the source network to the new network. This is expected to be successful when takin an action in the new game that exists in the previously learned game and tends to contribute to a winning strategy there, does so in the new game too.

Finally, if there is a change in the number of players then a *player transformation* will be necessary. If the number of players increases, the new player head could be initialised with one of the existing player heads. If the number of players decreases, then one of the player heads should not be copied.

3.3 Multitask Training for General Game Playing

Our multitask method for GGP is inspired by the success of AMN [12], which trains based on a policy distillation and a scaled feature regression component. For GGP, we

[2] It is worth noting that many of the multitask methods in GVGAI are not directly applicable to GGP and Generalised AlphaZero because they assume specific features of the network, such as identical input shapes and action spaces, regardless of the game being played.

use only the policy distillation component since this is considered the more successful part of the network, with improvements to the AMN removing the feature regression component [23,4]. The policy distillation component trains based on the KL divergence of the multitask network compared to the expert. In the method we propose, the mimic network retains the structure of Generalised AlphaZero, but instead of training based on self-play it uses the KL divergence from the experience of a series of experts. During training, the multitask network completes N rounds playing a particular game guided by the expert on that game, before moving on to the next game and expert. This process is repeated M times. Like in AMN, it is important to balance N as too many rounds of training on a particular game will over-fit the multitask network. Once trained, the multitask network is used as the source for transfer. It is important to note that this method is limited to train on games that have the same propnet size and number of actions.

4 Experimental Evaluation

4.1 Generic Agent Designs

We tested two generic agent designs based on the distinction between single network agents and those that use a network trained according to the multitask training method defined in Section 3.3. Each agent has a variety of settings that can be adjusted to set the variables for each type of transfer.

4.2 Game Variations

Our experimental evaluation of transfer learning applied to deep reinforcement GGP agents focuses on two types of game variation, namely, goal variation and environment variation. Goal variation concern the winning conditions of a game. Specifically, we investigate five such goal variations: (1) **original**; (2) **subset**, where solutions to the original game will still win the new game but there are other paths to victory; (3) **minor goal changes**, where solutions to the original game form a partial solution of the new game; (4) **superset**, where some solutions to the original game will lose the new game; and (5) **inverse goal**, where all winning states become losing states and vice versa.

Environment variations change the environment while retaining the “nature” of the original game. These could be adjustments to the board, the number of players, or new features such as obstacles on a board etc. We categorise environment variations by whether they affect the structure of a Generalised AlphaZero network. If there is no change to the input or action space, the change is considered minor, otherwise, i.e. when at least one of the two is altered, it is considered major. An example of a major change is adjusting the board size. Figure 2 ranks these variations based on similarity and maps them to the transfer variables hypothesised to perform best.

4.3 Evaluation Methodology

We implemented the generic agent designs on Google Cloud’s Deep Learning VM Image running Tensorflow 1, Python 3.7 and Cython 3.7.[3] The agents were evaluated by

[3] `https://cloud.google.com/deep-learning-vm`, accessed 01/19/21.

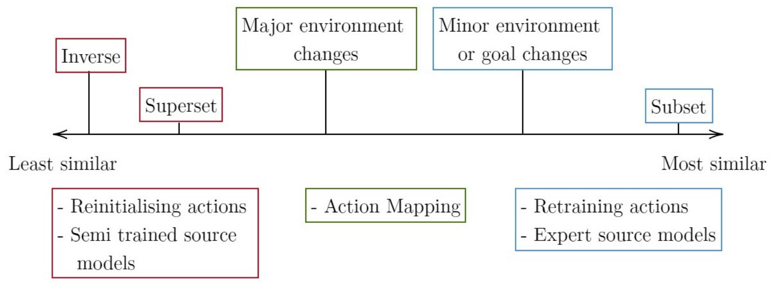

Fig. 2: Hierarchy of game variants and strategies.

playing sets of 50 simulated games against a regular UCT agent every 50 rounds of training. Each player had a time limit of two seconds to decide their next move, and the first two moves of each game were randomised. This randomisation was necessary to ensure a spread of independent tests as all of the agents used are mostly deterministic. The randomised moves were seeded so that in each trial game all agents began with the same two moves. This allows direct comparison between agents while retaining an independent set of tests. UCT was chosen as a benchmark because it represents state-of-the-art in GGP without transfer or deep reinforcement learning [5]. The parameters of the underlying Generalised AlphaZero agent were kept constant.

Even though there are only two principled agent designs (single and multi-network), the number of combinations of variables required to run over 60 experiments on three different base games. Single network agents were tested with both semi- and fully trained source networks and with weight, action, and input transformations. Multi-network agents were tested with gradual input transformation and reinitialised move layers.

We tested variants of Connect-4 and Breakthrough along with the simple game Nim; these games were chosen due to their prominence in past GGP competitions [5]. We apply a common evaluation metric known from the literature [20], with a particular focus on *jumpstart* and *time to threshold* as the main goal of GGP is to get the best agent in limited time. A threshold was chosen for each game based on the winning score an agent without transfer converged to. Thresholds varied due to the effect of the random initial moves. Connect-4 variant thresholds ranged from 25 to 40 games, and from 40 to 45 for Breakthrough and Nim variants. As TL has no effect on the time required for each round of training, *target time* is measured in rounds rather than seconds or hours.

5 Results and Discussion

Transfer Learning proved to increase the efficiency of Generalised AlphaZero (GAZ) on most game variants. Both single and multi-network agents outperformed agents without transfer. The multi-network design had the greatest success, outperforming agents with single network transfer. Negative transfer was only experienced when complex transfer was necessary and when learning superset goal variants. Successful agents had a strong *jumpstart* and significant decrease in *time to threshold*. Consequently, the training time for these agents went down. On some occasions, the *jumpstart* of the multi-network agent outperformed a network without transfer that had trained for thousands of rounds.

5.1 Single Network Agents

Although not as successful as their multi-network counterparts, single network agents still achieved positive transfer on a variety of the game variants. Experiments with subset goal variation, minor goal changes, and minor environment changes all showed positive transfer: Single network agents with tuned variables could improve *time to threshold* significantly when learning *Connect-4 Zig Zag*[4] (150 rounds to reaching the threshold with transfer, vs. 400 rounds with no transfer learning), *Connect-5* (500 vs. 2700), *Breakthrough Suicide*[5] (150 vs. 1500), and *Nim Variant-2*[6] (300 vs. 500). Even when there was no improvement to *time to threshold*, single network agents often had a strong *jumpstart*. Figure 3 shows the *jumpstart* of the single network retrain agent learning Connect-4 on a 10x8-board and Breakthrough Suicide, where the agent outperformed GAZ for more than 250 and 600 rounds, respectively. This suggests that expert networks contain good general information about the original game they train on, resulting in an initial advantage playing the new game. In GGP competitions, *jumpstart* is an advantage. If the *startclock* ends within the period of *jumpstart*, then strong initial performance would win.

Tuning the SNT variables had the expected results. Semi-trained agents gave networks a boost in *time to threshold* in the least similar games, like Connect-4 Inverse Goal, and had longer *time to threshold* in the most similar game variants, like Connect-5. Weight transformations to the action layer also performed as expected. Retraining gave agents an advantage when learning games where behaving in the same was as you would in the source game was likely to win in the target game as well. Examples included Nim Variant-2, Connect-4 and -5 on a 10x8 board. Reinitialising yielded better results in the least similar variants, including Connect-4 Inverse Goal, Nim Inverse Goal and Connect-4 Miss-1[7].

5.2 Complex Network Transfer

Both single and multi-network agents struggled to outperform those without transfer learning when complex transfer was necessary. Gradual input transformation failed to reduce the input to a state representation similar to the input of the source network. As a result, agents encountered negative transfer at the beginning, even with small changes to the input size. The larger the change, the more negative transfer was observed. The relatively small changes in Nim were best handled with only short periods of negative transfer for the first 50-100 rounds of training. Learning Breakthrough on 5x6 and 4x7 boards, where the input changed in size by a few hundred nodes, agents experienced several hundreds of rounds of negative transfer as they had been trained to expect input in the format of the original game.

[4] with a “zig zag” pattern of four pieces as an additional winning condition
[5] with inverted goal
[6] with stacks 2,2,10,10 instead of the original 1,5,4,2
[7] with the goal to connect 2 pieces, *miss* the third slot, and put a piece in the fourth slot

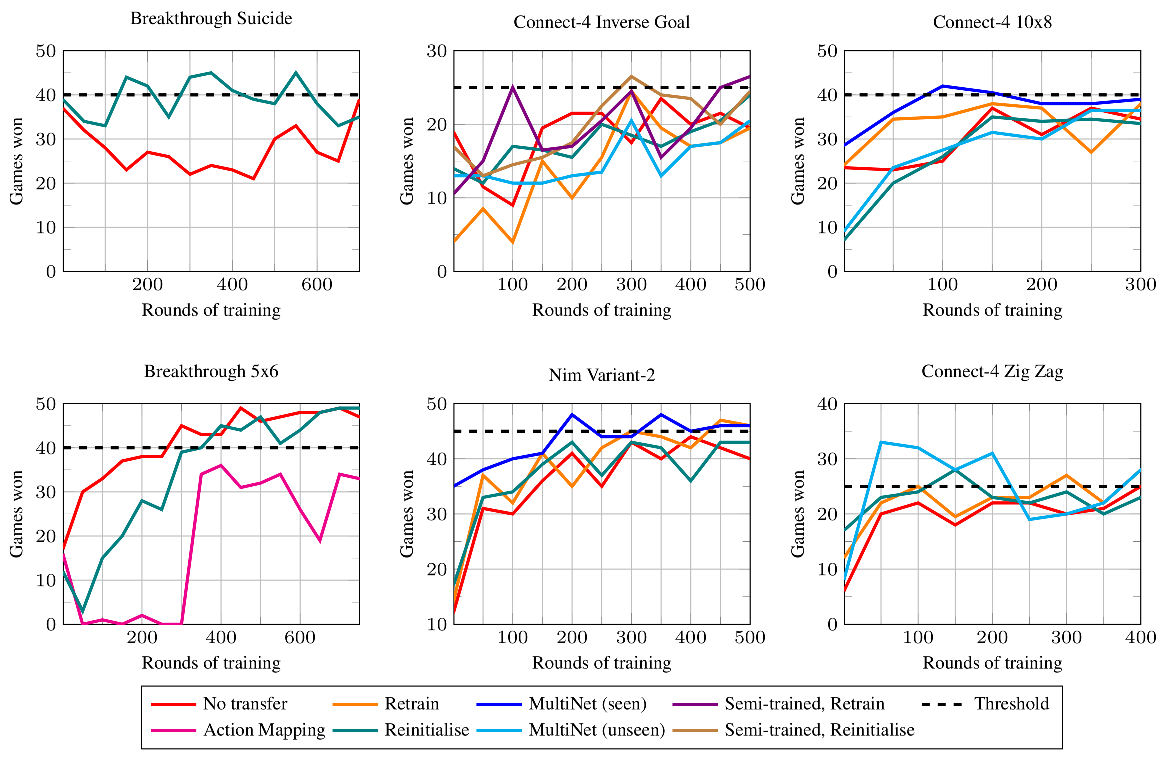

Fig. 3: Results testing transfer learning techniques and variables across Connect-4, Breakthrough and Nim variants.

5.3 Multi-Network Agents

Multi-network agents mitigated negative transfer in many more game variants than single networks. They had much stronger jump starts. In Connect-4 Zig Zag the *jumpstart* (cf. Figure 3) performed even better than the GAZ once it was fully trained. The agent typically met thresholds in 50 rounds of training when learning games they had expert knowledge in, and in 150 rounds of training learning new games. Of the 16 game variants multi-network agents were tested on, they outperformed all other transfer agents in 14 variants. Negative transfer was experienced in the superset variants Connect-4 Miss-1 and Connect-4 Inverse Goal when not been previously seen by the network. Analysis of the different variables for transfer drew the same conclusions as for the single networks.

Choosing a large number and wide range of expert networks further mitigated negative transfer. Multi-network agents trained without any inverse variants struggled to learn unseen inverse goals. Adding an inverse variant to the set of experts improved this even when there were many experts; the multi-network agent trained in variants of Nim showed no discernible negative transfer when learning variants with inverse goal. Performance also improved with more expert networks. Increasing the number of Nim variants used in multitask training worked best with the highest number of expert networks, even when the number of training rounds was kept constant. This suggests that increasing the number of varied experts is more effective than hand picking them.

The efficiency of the multitask training and its superior performance suggests that it is always better to train a multi-network agent rather than a single network one. The primary concern with multi-network agents is their increased training time. The multitask training method defined in this research is much faster than GAZ training due to its lack of self-play. As a result, the difference in training time between single and multi-network agents is negligible. If a library of buffers for each of the experts is built when they are trained, it takes just a few minutes to complete multitask training. The self-play component of GAZ takes anywhere between 20 and 80 seconds per round of training, so even when assuming optimal performance, adding a multitask network to the architecture takes roughly the same time as six rounds of training. Given that networks must train for hundreds or thousands of rounds, multitask training adds a negligible amount of time to the process but with a significant jump in performance.

6 Conclusion and Future Work

Introducing TL to the architecture of GAZ reduced the training time on game variants. The multi-network approach was the most successful method with strong results in seen and unseen games. Multi-network agents were much more resilient to negative transfer across the spectrum of variants, provided a diverse set of source games was chosen. When using single network agents, it proved prudent to consider the disparity between game variants when setting the variables for transfer. Retraining action weights assisted transfer to the most similar variants, whereas reinitialising action weights and using semi-trained source networks worked best in the least. Single and multi-network struggled to complete positive transfer between networks with different structures. Addressing this in future work is key to stable performance across a wider variety of variants.

References

1. Asawa, C., Elamri, C., Pan, D.: Using transfer learning between games to improve deep reinforcement learning performance and stability. URL http://web.stanford.edu/class/cs234/past_projects/2017/2017_Asawa_Elamri_Pan_Transfer_ Learning_Paper.pdf (2017)
2. Auer, P.: Using confidence bounds for exploitation-exploration trade-offs. The Journal of Machine Learning Research **3**, 397–422 (2003)
3. Banerjee, B., Stone, P.: General game learning using knowledge transfer. In: Proceedings of IJCAI. pp. 672–677 (2007)
4. Czarnecki, W.M., Pascanu, R., Osindero, S., Jayakumar, S.M., Swirszcz, G., Jaderberg, M.: Distilling policy distillation. CoRR **abs/1902.02186** (2019)
5. Genesereth, M., Björnsson, Y.: The international general game playing competition. AI Magazine **34**(2), 107 (2013)
6. Genesereth, M., Thielscher, M.: General Game Playing. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool (2014)
7. Goldwaser, A., Thielscher, M.: Deep reinforcement learning for general game playing. In: Proceedings of AAAI. pp. 1701–1708. AAAI Press (2020)
8. Gunawan, A., Ruan, J., Thielscher, M., Narayanan, A.: Exploring a learning architecture for general game playing. In: Proceedings of AI. pp. 294–306. Springer, Canberra (2020)
9. Hinrichs, T., Forbus, K.D.: Transfer learning through analogy in games. AI Magazine **32**(1), 70–83 (2011)
10. Hsu, S., Shen, I., Chen, B.: Transferring deep reinforcement learning with adversarial objective and augmentation. CoRR **abs/1809.00770** (2018)
11. Kuhlmann, G., Stone, P.: Graph-based domain mapping for transfer learning in general games. In: Proceedings of ECML. pp. 188–200 (2007)
12. Parisotto, E., Ba, J., Salakhutdinov, R.: Actor-mimic: Deep multitask and transfer reinforcement learning. CoRR **abs/1511.06342** (2015)
13. Pérez-Liébana, D., Liu, J., Khalifa, A., Gaina, R.D., Togelius, J., Lucas, S.M.: General video game AI: a multi-track framework for evaluating agents, games and content generation algorithms. CoRR **abs/1802.10363** (2018)
14. Schkufza, E., Love, N., Genesereth, M.: Propositional automata and cell automata: Representational frameworks for discrete dynamic systems. In: Proc. of AI. pp. 56–66 (2008)
15. Shao, K., Tang, Z., Zhu, Y., Li, N., Zhao, D.: A survey of deep reinforcement learning in video games. arXiv preprint arXiv:1912.10944 (2019)
16. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., et al.: Mastering the game of go with deep neural networks and tree search. Nature **529**, 484–503 (2016)
17. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., et al.: A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. Science **362**(6419), 1140–1144 (2018)
18. Sobol, D., Wolf, L., Taigman, Y.: Visual analogies between Atari games for studying transfer learning in RL. arXiv preprint arXiv:1807.11074 (2018)
19. Taylor, M.E., Kuhlmann, G., Stone, P.: Accelerating search with transferred heuristics. In: ICAPS-07 workshop on AI Planning and Learning (2007)
20. Taylor, M.E., Stone, P.: Cross-domain transfer for reinforcement learning. In: Proceedings of the ICML. p. 879–886. New York (2007)
21. Taylor, M.E., Stone, P.: Transfer learning for reinforcement learning domains: A survey. Journal of Machine Learning Research **10**, 1633–1685 (2009)
22. Yang, T., Hao, J., Meng, Z., Zhang, Z., Wang, W., et al.: Efficient deep reinforcement learning through policy transfer. arXiv preprint arXiv:2002.08037 (2020)
23. Yin, H., Pan, S.J.: Knowledge transfer for deep reinforcement learning with hierarchical experience replay. In: Proceedings of AAAI. p. 1640–1646. AAAI Press (2017)