A Transferable Neural Network for Hex

Chao Gao [a,*], Siqi Yan [b], Ryan Hayward [b] and Martin Müller [b]
[a] *Department of Computing Science, University of Alberta, Canada*
E-mail: cgao3@ualberta.ca
[b] *Department of Computing Science, University of Alberta, Canada*
E-mails: syan3@ualberta.ca, hayward@ualberta.ca, mmueller@ualberta.ca

Abstract. The game of Hex can be played on multiple boardsizes. Transferring neural net knowledge learned on one boardsize to other boardsizes is of interest, since deep neural nets usually require large size of high quality data to train, whereas expert games can be unavailable or difficult to generate. In this paper we investigate neural transfer learning in Hex. We show that when only boardsize independent neurons are used, the resulting neural net obtained from training on one base boardsize can effectively generalize — without fine-tuning — to multiple target boardsizes, larger or smaller. When transferring to larger boardsizes, fine-tuning provides faster learning and better performance. The strength of the transferable network can be amplified with search: with a single neural net model trained on games from a base boardsize, we obtain players stronger than MoHex 2.0 on multiple target boardsizes.

Keywords: Hex game, deep learning, transfer learning, neural network

1. INTRODUCTION

Invented by Piet Hein in 1942 (Hein, 1942), Hex is a two-player alternate-turn zero-sum perfect-information game played on an $n \times n$ hexagonal board. Players Black and White each own a distinct pair of opposing borders. On a turn, a player places a stone of their colour on an empty hexagonal cell. The winner is whoever forms a chain that connects their two borders. 11×11 is the original boardsize, with 13×13 and 19×19 also popular (Hayward and Weninger, 2017). Hex allows no draws: this follows by simple graph theoretic arguments (Pierce, 1961) equivalent to a two-dimensional version of Brouwer's fixed-point theorem (Gale, 1979). Nash (Nash, 1952) *ultra-weakly* solved the game with a *strategy stealing* argument. However, finding an explicit winning strategy for an arbitrary position is PSPACE-complete (Reisch, 1981; Édouard Bonnet et al., 2016). In practice, to mitigate the first player's paramount advantage, a *swap* rule is usually applied, by which White has the option to steal Black's opening move as their first move. Figure 1 shows Hex games on different boardsizes.

Hex games on different boardsizes are related: 1) by placing extra stones along the borders, the smaller size can be considered as a special case of the larger; 2) by placing extra empty rows along the borders, the larger size can be considered as a generalization of the smaller.

Deep neural networks have been applied to two-player board games, in particular Go (Tian and Zhu, 2015; Clark and Storkey, 2015; Maddison et al., 2015; Silver et al., 2016, 2017) and Hex (Gao et al., 2017a,b; Anthony et al., 2017), showing that the superior representation learning ability of deep nets can be harnessed for providing high quality knowledge. However, these studies are limited to a fixed boardsize, whereas the games allow multiple boardsizes. As seen in Figure 1, the similarities between Hex on different boardsizes raise the following questions:

[*] Corresponding author. E-mail: cgao3@ualberta.ca.

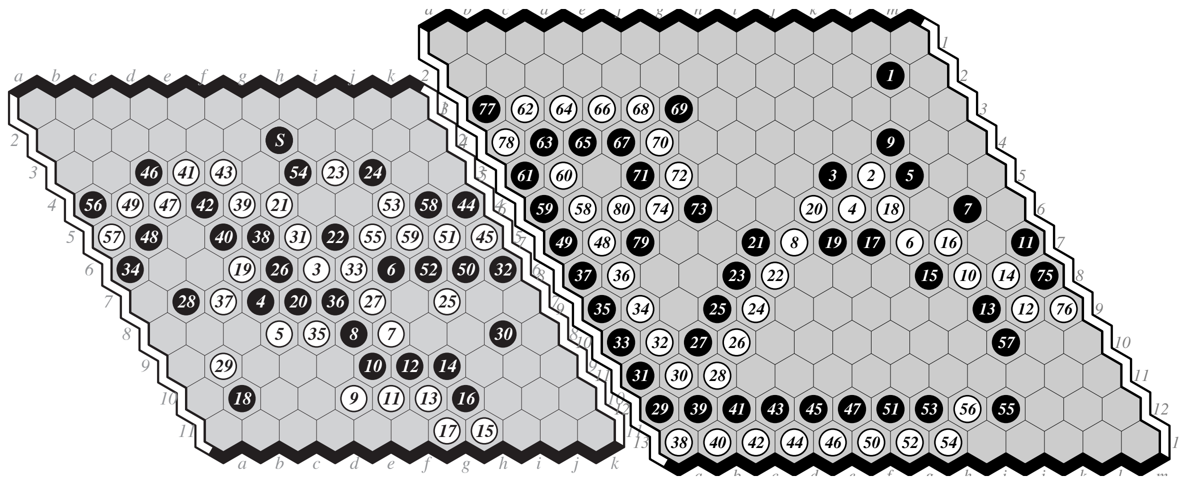


Fig. 1. Classic (left) and 13×13 (right) Hex games (Gao et al., 2017a).

- To what extent can learned neural net knowledge be reused on a smaller board?
- To what extent can such knowledge be generalized to a larger board?

In this paper, we investigate neural *transfer learning* (Pan and Yang, 2010; Yosinski et al., 2014) in Hex. We make these contributions:

- we introduce a transferable neural net architecture for multiple-boardsize Hex;
- we quantify knowledge transferability, both upward and downward;
- we improve MoHex 2.0 on multiple boardsizes.

2. GENERAL NEURAL NETWORKS IN HEX

2.1. General Architecture

Typical transfer learning with neural networks aims to reuse a *base* network f_{θ_A}, which has been trained on data set $\mathcal{D}_A$ for task $\mathcal{T}_A$, when learning a *target* neural net for task $\mathcal{T}_B$. Assume that the base network has n layers. These approaches could be used:

(1) Design a new network for $\mathcal{T}_B$, which borrows the first $m \in \{1, 2, \ldots, n\}$ layers from f_{θ_A}, then adds task specific layers onward. This approach requires a dataset $\mathcal{D}_B$ for the target task. When training on the new dataset, there are two obvious options: 1) keep the m borrowed layers *frozen*; 2) *fine-tuning*: use only the borrowed neurons' initial weights and optimize the whole neural net as usual.

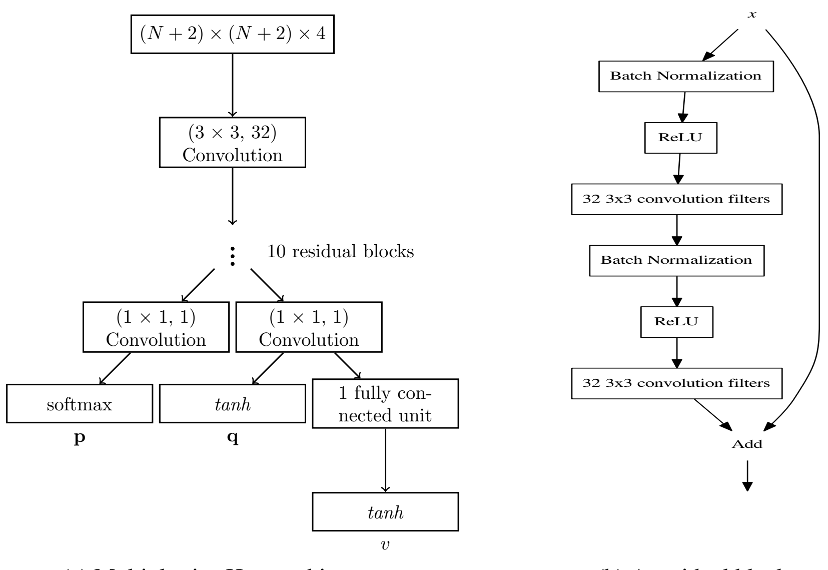
(2) Let the target network have the same architecture and the same set of neural parameters as in the base network, and reuse all learned weights from f_{θ_A} in the target net.

A limitation of 1. is that a dataset for target task must be available. A challenge of 2. is that the architecture used by f_{θ_A} must avoid task dependent neurons, e.g., fully-connected units. In Hex, 2. is more appealing, since for many boardsizes expert data are scarce and costly to generate. Also, neural nets are usually combined with search, which can amplify the net's strength via averaging lookahead evaluation. So, when transferring a neural net, it is desirable that the network can be strong enough to be useful in search, even without fine-tuning.

Convolution neural networks (CNNs) have been applied to various visual recognition tasks (LeCun et al., 1998). Unlike fully-connected neural nets, CNNs exploit spatial locality by using convolution *filters*, where each filter responds to patterns in a local receptive field. By a weight sharing scheme, a CNN filter is replicated across the entire visual field, so replicated units share the same parameterization. This replication allows convolution filters to detect specific features regardless of their location within the visual field.

In this paper, leveraging the translation invariance (LeCun et al., 1998) of CNNs, we make an effort towards 2.— we describe a Hex network that uses mainly boardsize independent neurons for move and value predictions, anticipating that the learned convolution filters can be instantly useful beyond the board size of training.

As with Go (Silver et al., 2017), we take a zero-knowledge approach when encoding input features. A Hex state is independent of move history, so we encode a position with four binary planes: 1) black plane (black stones are marked 1, others are 0); 2) white plane; 3) empty plane (empty board cells are marked 1, others are 0); 4) to-play plane, (1 if Black moves next, else 0).



(a) Multiple size Hex: architecture. (b) A residual block.

Fig. 2. A given Hex state of boardsize $8 \leq N \leq 19$ is padded with black or white stones along each border, then fed into a feedforward neural net with convolution filters. A fully-connected bottom layer compresses results to a single scalar v.

Figure 2 shows the architecture for multiple-boardsize Hex. The neural net output has three heads: **p**, **q**, v respectively estimate a move probability vector, an action-value vector, and a state-value scalar. The net has a total of $4 \cdot 3 \cdot 3 \cdot 32 + (32 \cdot 3 \cdot 3 \cdot 32 + 32 + 32) \cdot 2 \cdot 10 + 1 + 1 + n \cdot n = 185890 + n^2$ weights, where only the final fully-connected v-prediction layer depends on input boardsize n. The main difference between Figure 2 and the three-head architecture (Gao et al., 2018b) is that fully-connected layers are removed for the **q**-head: by doing so we expect that the action-value vector can generalize to boardsizes not used in training.

2.2. Optimization

Following (Gao et al., 2018b), to optimize the three-head neural net f_θ, we use a loss function that combines policy and value:

$$
\begin{aligned}
L(f_\theta; \mathcal{D}) = \sum_{(s,a,z_s)\in\mathcal{D}} \Bigg(& w\bigg(\frac{1}{2}(z_s - v(s))^2 + \frac{1}{2}(z_s + q(s,a))^2 \\
& + \frac{\max(-z_s, 0)}{|\mathcal{A}(s)|} \sum_{a'\in\mathcal{A}(s)} (z_s + q(s,a'))^2 + (\min_{a'} q(s,a') + v(s))^2 \bigg) - \log \pi(a|s) + c||\theta||^2 \Bigg)
\end{aligned}
\tag{1}
$$

where $0 < w \leq 1$ is a weighting factor used to control the relative weight of the value loss; c is a constant for the level of L_2 regularization; (s, a) is a state-action pair from dataset $\mathcal{D}$; z_s is the game result with respect to the player to-play at s, which can be extracted from dataset $\mathcal{D}$; $\mathcal{A}(s)$ is the set of actions at s. Hex has deterministic transitions, so $q(s, a)$ is equivalent to $v(s')$ given $s \xrightarrow{a} s'$. The neural net value outputs $v(s)$, $q(s, a)$ are with respect to the player-to-move at s or (s, a). The loss function has four parts: policy loss $-\log \pi(a|s)$; L2 regularization $c||\theta||^2$; w weighted value loss; and a quadratic inconsistency penalty between state-value $v(s)$ and minimum action-value $\min_{a'} q(s, a')$.

3. RELATED WORK

Transferable learning (Pan and Yang, 2010; Yosinski et al., 2014) is an everlasting quest of artificial intelligence. Starting with image recognition (Krizhevsky et al., 2012), learned deep neural networks have shown curious generalization ability on tasks that differ from their training. (Yosinski et al., 2014) documented transferability of neural features learned on ImageNet, concluding that transferability is negatively influenced by the distance between the base and target tasks. Even for distant target tasks, transferring neurons from an existing model improved performance after *fine-tuning*, perhaps because of the reduction of over-fitting due to neural *co-adaption*.

Training a net with separate policy and value estimates has been explored in computer Go (Silver et al., 2016, 2017). The trained nets provide high quality expert knowledge of the game, resembling human quick-decision intuition. This is amplified when combined with Monte Carlo Tree Search (MCTS) (Coulom, 2006; Silver et al., 2016, 2017). Deep neural nets have also been used in Hex (Gao et al., 2017a), yielding MoHex-CNN, which is stronger than MoHex 2.0 (Huang et al., 2013; Pawlewicz et al., 2015) on 13×13 Hex. MoHex-CNN uses policy net output for move prior probabilities during the *in-tree* phase. For leaf evaluation, MoHex-CNN still uses pattern-based playouts as in MoHex 2.0. After replacing the pattern-based playouts with neural value estimates, a three-head neural net (Gao et al., 2018b) — when trained on the same dataset — outperforms MoHex-CNN on 13×13 Hex.

4. EXPERIMENTS

4.1. Datasets

We use three publicly available datasets:

- about 6×10^4 8×8 games from interplay of MoHex 2011, MoHex 2.0 and Wolve (Gao et al., 2017b);
- 9×9 MoHex 2.0 self-play games yielding about 1 million state-action-value tuples (Gao et al., 2018a);
- about 1.5×10^4 13×13 MoHex 2.0 self-play games (Gao et al., 2017a).

4.2. Setup

Our neural net is implemented using Tensorflow (Abadi et al., 2016), trained by the Adam (Kingma and Ba, 2014) optimizer with learning rate 0.001 and mini-batch size 128 for 100 epochs. Model parameters are saved after each epoch. Experiments ran on an Intel i7-6700 CPU computer with one GTX 1080 GPU. We set the L_2 regularization constant c to 10^{-5}, with value loss weight $w = 0.01$. We ran two sets of experiments:

- Train the neural net on 13×13 games and investigate transferability to smaller boards. We split the 13×13 dataset into training and test sets as in (Gao et al., 2017a).
- Train the neural net on 9×9 games and investigate transferability to larger boards. We split the 9×9 dataset in the same way as (Gao et al., 2018a).

4.3. Prediction Accuracy across Boardsizes

We first investigate prediction accuracies of the saved neural net models. Figure 3 shows the test prediction accuracies of **p** and **q** heads of the neural net models obtained from learning on 13×13 games. As expected, neural net models optimized on 13×13 games achieved high accuracy on smaller boards. Prediction accuracy on 8×8 and 9×9 are better than on 13×13, reflecting that smaller boardsizes are easier to master. Compared to the neural net designed specifically for 13×13 Hex and trained on the same dataset (Gao et al., 2018b), this network's **q**- and **p**-heads generally achieved slightly larger errors. However, this sacrifice seems worthwhile in light of the ability of the new architecture to generalize to other boardsizes.

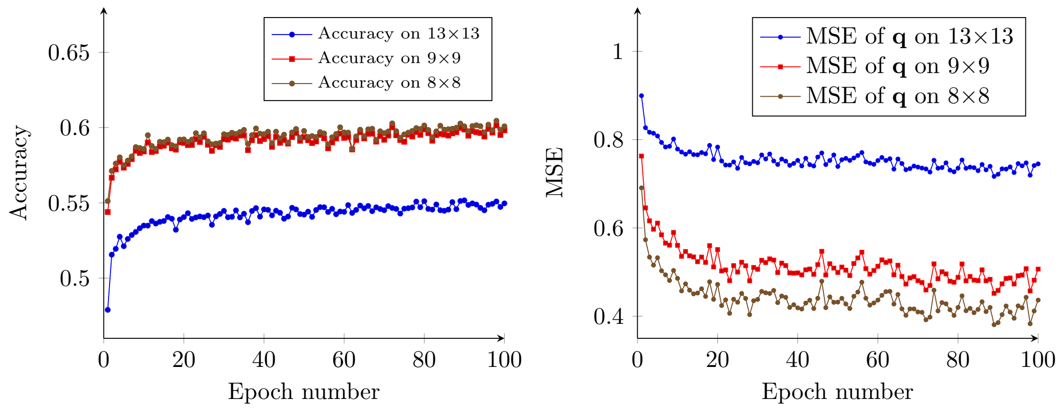


Fig. 3. 13×13 training: prediction accuracy across boardsizes.

It is more interesting to measure transferability from smaller boards to larger, as on smaller boards it is easier produce high quality games. Figure 4 shows prediction accuracy after training the net on 9×9 games. Surprisingly, both **q** and **p** yield reasonable prediction accuracy even on 13×13 Hex. However, on 13×13 Hex, by comparison, prediction accuracy of Figure 4 is much worse than in Figure 3.

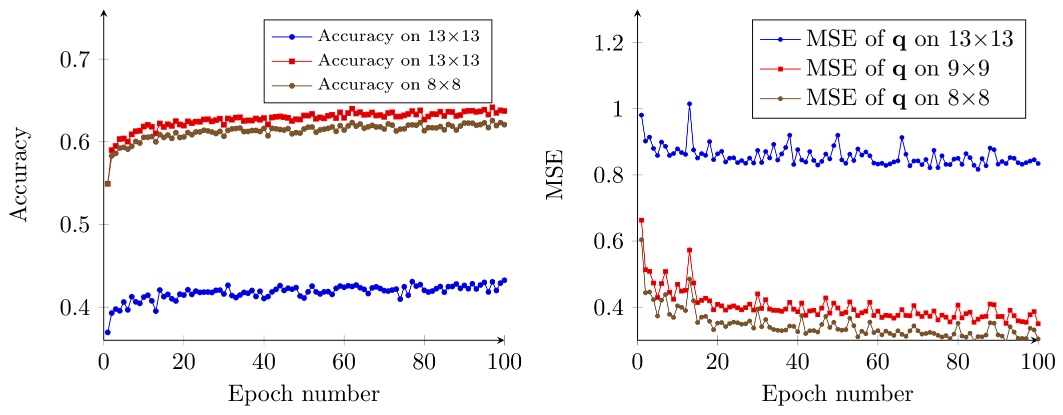


Fig. 4. 9×9 training: prediction accuracy across boardsizes.

4.4. Combined with Search

How useful are our neural nets when used in search? In this section we answer this question by combining our nets with the MCTS of MoHex 2.0. As in (Gao et al., 2017a, 2018b), we leave the search framework and hyper-parameters unchanged but 1) replace move prior probability with neural net move prediction (Gao et al., 2017a) and 2) use action-value in **q** to back up leaf estimates (Gao et al., 2018b). The *v*-head is boardsize dependent, so we use it only when the running boardsize is the same as that used in training; otherwise, only the **q**- and **p**-heads are used. In our implementation, move prior and action-value are both stored at each node creation. We call this new transferable neural net program MoHex3H.

We evaluate 10 neural net models at an epoch interval of 10. Each model is combined with MoHex3H and played against MoHex 2.0. Table 1 shows the results on boardsizes 9×9 to 13×13. We did not include 8×8 data, because positions on this board are easily solved (Henderson et al., 2009; Pawlewicz et al., 2015). For all programs, we allow 10^4 simulations per move, since on the GTX 1080 GPU, MoHex 2.0 and the new programs with neural nets take similar time per simulation (Gao et al., 2017a). Each tournament is played by iterating over all $N\times N$ opening moves. Each opening is used twice, with each program playing as first-player and second-player; and no swap rule.

Table 1
MoHex3H using 13×13-trained nets: win rate (%) versus MoHex 2.0 and MoHex-CNN. Columns 2-11 show strength by epoch.

Boardsize \ Epoch number	10	20	30	40	50	60	70	80	90	100
9×9	71.0	64.8	69.1	75.9	73.5	77.2	69.1	74.7	77.8	67.9
10×10	71.0	78.0	66.5	71.0	82.5	83.5	80.5	78.0	78.5	72.5
11×11	67.4	67.8	71.5	74.0	71.9	76.4	74.8	78.5	78.1	76.4
12×12	67.0	71.9	70.5	73.6	76.4	78.5	78.1	77.4	79.9	75.7
13×13	63.0	63.3	68.3	69.8	73.7	76.3	74.9	75.4	74.3	74.9
13×13	44.1	42.9	46.5	55.3	58.8	61.2	55.3	52.4	61.2	55.3

Table 1 shows results against MoHex 2.0 using net models learned from the 13×13 dataset. Notice that the high **p**- and **q**-head prediction accuracy of Figure 3 yields strong play on smaller boardsizes. MoHex-CNN ran only on 13×13 Hex, so we include the data against MoHex-CNN in the last row:

Table 2
MoHex3H using 9×9-trained nets: win rate (%) versus MoHex 2.0.

Boardsize \ Epoch number	10	20	30	40	50	60	70	80	90	100
9×9	63.6	69.1	63.0	65.4	65.4	72.2	74.7	72.8	71.6	72.8
10×10	63.5	70.0	71.5	68.0	67.0	75.5	76.0	76.0	76.5	77.5
11×11	62.8	58.3	64.0	66.1	69.0	66.1	67.8	64.0	66.1	65.7
12×12	51.0	45.8	56.2	53.5	45.8	60.1	54.9	49.3	58.7	58.0
13×13	35.3	39.4	47.1	47.6	46.5	42.9	42.4	42.9	52.9	54.1

Table 3
MoHex3H using 9×9-trained nets, with **p**-head only: win rate (%) versus MoHex 2.0.

Epoch number \ Boardsize	9×9	10×10	11×11	12×12	13×13
90	68.5	75.5	68.2	65.3	60.6
100	68.5	70.5	70.7	67.4	63.5

MoHex3H defeats MoHex-CNN with this transferable neural net, although the win rate is lower than that of a three-head net with fully connected bottom layer (Gao et al., 2018b).

Table 2 shows results against MoHex 2.0 using net models learned from the 9×9 dataset. The resulting program defeats MoHex 2.0 even on boardsize 13×13, although its margin of victory there is less than on other boardsizes. MoHex 2.0's playout policy was trained mostly on 13×13 games (Huang et al., 2013), so we conjecture that the low **q**-head prediction accuracy is insufficient to match the strength of MoHex 2.0 playouts. To verify this, we ran extra tournaments with the **q**-head in MoHex3H's search disabled, forcing MoHex3H to use the same pattern-based playouts as MoHex 2.0. By comparing the results in Tables 2 and 3, we can see that, at epochs 90 and 100, the winrate against MoHex 2.0 dropped on boardsizes 9×9 and 10×10 but rose on boardsizes 11×11 and larger, suggesting that the **p**-head consistently aided search, while the **q**-head was insufficiently accurate to replace pattern-based playouts on larger boardsizes.

In summary, we conclude that 1) without fine-tuning, transferring neural knowledge to larger boardsizes is harder than to smaller and 2) when transferring to larger boardsizes, the **p**-head seems more robust than the **q**-head.

4.5. Fine-tuning

We further investigate fine-tuning 9×9-trained neural models on 13×13 data. Specifically, we use the neural net model at epoch 100 in Figure 4 to initialize training on the 13×13 dataset.

Figures 5 and 6 compare the learning curves on the same 13×13 dataset by warm initialization and by starting from random weights. Notice that learning benefits from warm initialization: the neural net learns faster and achieves better accuracies. This strength gain is amplified with search: when used with the MCTS of the previous section, the resulting player quickly passed 70% wins against MoHex 2.0: at epoch 10, the rate is already 72.4%. By comparison, Table 1 shows that with random weights initialization this took more than 40 epochs.

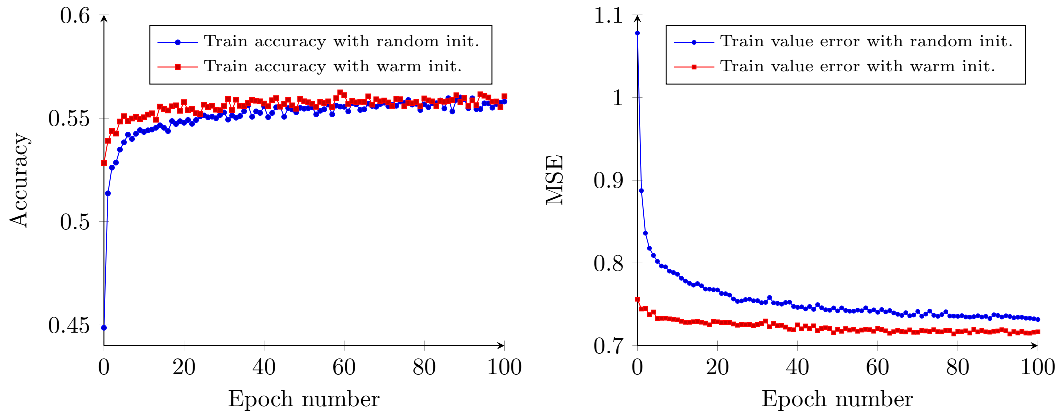


Fig. 5. 13×13 training errors, with and without warm initialization.

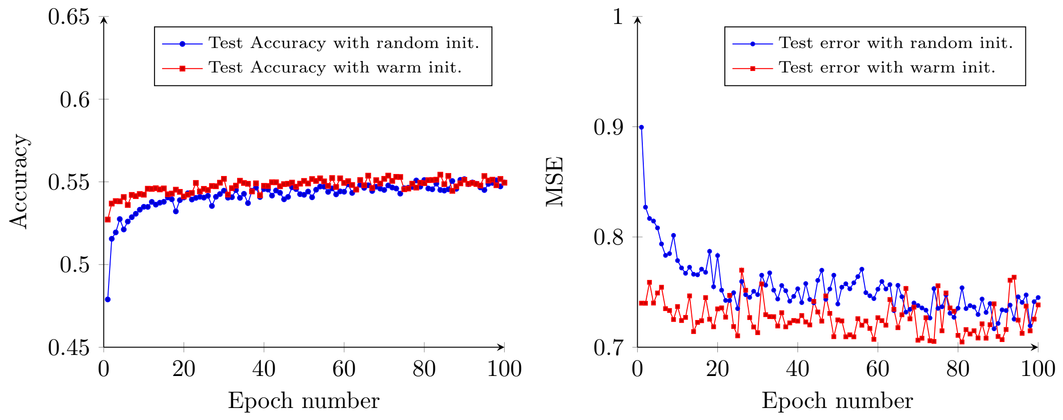


Fig. 6. 13×13 test errors, with and without warm initialization.

5. CONCLUSIONS AND FUTURE WORK

We have described a transferable neural network for the game of Hex. Our experiments show that both policy and value predictions can be used on boardsizes different than those used in training: as expected, knowledge transfer to smaller boards is more effective than to larger boards. Nonetheless, when combined with search, reusing neural network weights on larger boardsizes can result in strong play, regardless of whether there is fine-tuning.

To date, computer Hex players are weak on large boardsizes such as 19×19. A direction for future work is to use transferable neural nets to bootstrap the development of high performance playing programs on such boardsizes. Research on Go (Silver et al., 2017) has shown that iteratively optimizing neural nets on ever-stronger games generated by MCTS with previous neural net parameters produces a strong player. However, on regular computer hardware, an initially weak player might require long training times (). By using neural knowledge from smaller boardsizes, high performance playing on 19×19 Hex might be achieved with smaller training cost.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., et al. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Anthony, T., Tian, Z. & Barber, D. (2017). Thinking Fast and Slow with Deep Learning and Tree Search. *arXiv preprint arXiv:1705.08439*.

Clark, C. & Storkey, A. (2015). Training deep convolutional neural networks to play Go. In *International Conference on Machine Learning* (pp. 1766–1774).

Coulom, R. (2006). Efficient selectivity and backup operators in Monte-Carlo tree search. In *International Conference on Computers and Games* (pp. 72–83). Springer.

Édouard Bonnet, Jamain, F. & Saffidine, A. (2016). On the complexity of connection games. *Theoretical Computer Science*, *644*, 2 –28. Recent Advances in Computer Games.

Gale, D. (1979). The game of Hex and the Brouwer fixed-point theorem. *The American Mathematical Monthly*, *86*(10), 818–827.

Gao, C., Hayward, R.B. & Müller, M. (2017a). Move Prediction using Deep Convolutional Neural Networks in Hex. *IEEE Transactions on Games*.

Gao, C., Müller, M. & Hayward, R. (2017b). Focused Depth-first Proof Number Search using Convolutional Neural Networks for the Game of Hex. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17* (pp. 3668–3674).

Gao, C., Müller, M. & Hayward, R. (2018a). Adversarial Policy Gradient for Alternating Markov Games. In *ICLR 2018 workshop*.

Gao, C., Müller, M. & Hayward, R. (2018b). Three-Head Neural Network Architecture for Monte Carlo Tree Search. In *IJCAI*.

Hayward, R. & Weninger, N. (2017). Hex 2017: MoHex wins the 11x11 and 13x13 tournaments. *ICGA Journal*, 222–227.

Hein, P. (1942). Vil De laere Polygon. *Politiken newspaper*.

Henderson, P., Arneson, B. & Hayward, R.B. (2009). Solving 8×8 Hex. In *Proc. IJCAI* (Vol. 9, pp. 505–510). Citeseer.

Huang, S.-C., Arneson, B., Hayward, R.B., Müller, M. & Pawlewicz, J. (2013). MoHex 2.0: a pattern-based MCTS Hex player. In *International Conference on Computers and Games* (pp. 60–71). Springer.

Kingma, D. & Ba, J. (2014). Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Krizhevsky, A., Sutskever, I. & Hinton, G.E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Maddison, C.J., Huang, A., Sutskever, I. & Silver, D. (2015). Move evaluation in Go using deep convolutional neural networks. In *International Conference on Learning Representations*.

Nash, J. (1952). Some Games and Machines for Playing Them. Technical report D-1164, RAND Corporation.

Pan, S.J. & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, *22*(10), 1345–1359.

Pawlewicz, J., Hayward, R., Henderson, P. & Arneson, B. (2015). Stronger Virtual Connections in Hex. *IEEE Transactions on Computational Intelligence and AI in Games*, *7*(2), 156–166.

Pierce, J.R. (1961). Symbols, Signals and Noise (pp. 10–13). Harper and Brothers.

Reisch, S. (1981). Hex ist PSPACE-vollständig. *Acta Informatica*, *15*(2), 167–191.

Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*(7587), 484–489.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T. & Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature*, *550*(7676), 354–359.

Tian, Y. & Zhu, Y. (2015). Better computer Go player with neural network and long-term prediction. In *International Conference on Learning Representations*.

Yosinski, J., Clune, J., Bengio, Y. & Lipson, H. (2014). How transferable are features in deep neural networks? In *Advances in neural information processing systems* (pp. 3320–3328).

Leela-zero. https://github.com/gcp/leela-zero. Accessed: 2018-02-28.