Transfer learning and curriculum learning in Sokoban

Yang, Z.; Preuss, M.; Plaat, A.; Leiva, L.A.; PUrski, C.; Markovich, R.; ... ; Schommer, C.

Transfer Learning and Curriculum Learning in Sokoban

Zhao Yang(✉), Mike Preuss, and Aske Plaat

LIACS, Leiden University, Leiden, The Netherlands
{z.yang,m.preuss}@liacs.leidenuniv.nl

Abstract. Transfer learning can speed up training in machine learning, and is regularly used in classification tasks. It reuses prior knowledge from other tasks to pre-train networks for new tasks. In reinforcement learning, learning actions for a behavior policy that can be applied to new environments is still a challenge, especially for tasks that involve much planning. Sokoban is a challenging puzzle game. It has been used widely as a benchmark in planning-based reinforcement learning. In this paper, we show how prior knowledge improves learning in Sokoban tasks. We find that reusing feature representations learned previously can accelerate learning new, more complex, instances. In effect, we show how curriculum learning, from simple to complex tasks, works in Sokoban. Furthermore, feature representations learned in simpler instances are more general, and thus lead to positive transfers towards more complex tasks, but not vice versa. We have also studied which part of the knowledge is most important for transfer to succeed, and identify which layers should be used for pre-training (Codes we used for this work can be found at https://github.com/yangzhao-666/TLCLS).

Keywords: Reinforcement learning · Transfer learning · Sokoban

1 Introduction

Humans are good at reusing prior knowledge when facing new problems. As a consequence, we learn new tasks quickly, a skill of great interest in machine learning. In the human brain, information received by our sensors is first transformed into different forms, and different types of transformed information are stored in different areas of our brain. When another problem arrives later on, we retrieve useful information and adjust it to better suit solving this new problem. The knowledge stored in artificial neural networks is also re-usable and transferable [31]. In supervised learning, pre-trained networks are commonly applied in computer vision [17,25] and natural language processing [3,9]. Feature representations learned from images or words overlap to some extent, which makes such feature representations reusable and transferable. In reinforcement learning (RL), transfer learning is relatively new, although with the spread of deep neural networks, reusing pre-trained models becomes possible in RL as well [1,7].

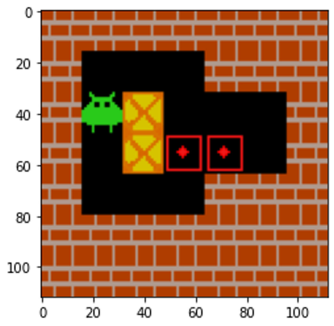


Fig. 1. An example instance of Sokoban.

Transfer learning works well in RL for recognition tasks, but tasks that rely heavily on planning are harder.

In this paper, we study transfer learning of behavior in Sokoban, a popular RL game in which planning is important [10,12]. It has already been proved that Sokoban is PSPACE-complete [8] and NP-hard problem [10]. An example instance from [22] is shown in Fig. 1. The goal of Sokoban is to control a warehouse worker that pushes all boxes onto targets. Sokoban is a challenging game where one wrong move can lead to a dead end (after a box has been pushed, it can not be pulled, and we cannot undo an inadvertent push). This non-reversibility is known to make games harder for AI agents [5]. Learning to solve Sokoban tasks is a challenge, especially in the multi-box scenario. For humans, if we have learned the basics of Sokoban (what is a box, what can an agent do), and if we are faced with a new, more complex instance, then we immediately focus on the new challenges in the instance, rather than re-learning the basics again. This building on prior knowledge saves time in the problem-solving process.

We investigate if we can achieve this kind of pretraining/fine-tuning learning in an RL agent. Our main hypothesis is that feature representations learned in Sokoban instances can be reused to improve solving other instances, and that features learned in simpler instances are more general and better transferable. We test this hypothesis by means of different experiments, in which parts of the neural network that has previously been trained on one type of instances (e.g. one box one target) are taken over (unchanged) to a new type of instances (e.g. two boxes two targets), whereas the remaining part of the network is trained on these new instances from scratch. The overall idea is that we see successful transfer if the preserved knowledge (in terms of network layers) leads to a faster learning process on the new problem type.

The main contributions of this paper are as follows: First, we show that feature representations learned in simple Sokoban instances can accelerate learning in more complex instances, indicating that curriculum learning can be used in Sokoban. Second, feature representations of simpler instances are more general and reusable than features learned in more complex instances. Third, our results confirm that in RL lower layers learn more general features. Interestingly, in some cases the best performance is achieved when more specific features are transfered, when source task and target task are similar enough to support these more specific features. Fourth, we found negative transfer from a simple supervised learning task, which tells us that choice and design of the source tasks are crucial. Fifth, we show that transferring top-fully-connected layers will not only be unhelpful but also harmful to the learning. We also used popular visualization techniques to explore potential reasons for successful transfers, which we explain in detail. Our code and test environments will be made available after blind review.

The paper is structured as follows: we first briefly review related work on transfer learning and Sokoban in the next section; then the environment and methods we are using are described in Sect. 3; Sect. 4 shows the experimental settings and results; in the last section, we conclude our work and discuss some potential future directions.

2 Related Work

De la Cruz et al. [6] studied the reuse of feature representations between two similar games: Breakout and Pong, using Deep Q Network (DQN). They used a 3-layer convolutional network. Weights learned in one game were transferred to improve learning the other game; results showed positive transfer of features between the different games. Pong and Breakout do not require planning; in our experiments, in Sokoban, we study how a curriculum of simpler instances can benefit the learning of complex instances. Spector et al. [26] used self-transfer in a DQN grid-world task to identify which parts should be transferred and which parts should be fixed, showing significant benefit of knowledge transfer.

Sokoban is a planning task that has been used as a benchmark for model-based reinforcement learning [16,22]. It has also been used in model-free RL [14,15], achieving performance competitive with model-based methods. The efficiency of AlphaZero-style curriculum learning has been shown by solving hard single Sokoban instances [11,12]. Previous works were aimed at solving single Sokoban instances; our paper focuses on the transferability of learned knowledge among *different* instances.

This transferability of learned feature representations was first studied in image classification problems [31]. It was shown that bottom layers in Convolutional Neural Networks (CNNs) extract more general features while ones extracted from back layers are more specific. In this paper, we verify this idea under RL settings.

Reinforcement learning [21,27] aims to reinforce behaviors of the learning agent by rewarding signals obtained from interactions with the environment. It has reached super-human performance in games such as Go [24], StarCraft [20,29], as well as Atari games [2] and robotic tasks. In this paper we follow the conventional MDP notation for RL [27].

Transfer learning reuses prior knowledge to improve the learning efficiency or performance in new tasks [28,30]. In reinforcement learning, higher-level knowledge such as macro actions, skills and lower-level knowledge such as reward functions, policies could be transferred. Transferring learned knowledge could take different approaches, such as reward shaping [4], learning from demonstration [19] and policy reuse [13].

3 Experimental Setup

The environment used in the paper is the Gym environment for Sokoban [23]; for the agent algorithms we follow Weber et al. [22]. Examples are shown in Fig. 2. The game is solved by controlling the agent (green sprite) to push all boxes (yellow squares) onto corresponding targets (red squares). There's no hint about which boxes should on which targets, and boxes can only be pushed; some actions are irreversible, and can leave the game in an unsolvable state. The difficulty of the game can be increased easily by putting more boxes as well as targets into generated rooms. The agent can go up, down, left, and right. The agent gets a final reward of 10 by pushing all boxes on targets. Pushing a box on a target will result a reward of 1 and a penalty of -1 for pushing a box off a target. We also give a small penalty of 0.1 for each step the agent takes.

We perform three types of experiments: (1) related tasks (source and target tasks are both RL tasks, while source tasks are to solve n-boxes Sokoban instances and target tasks are to solve m-boxes Sokoban instances, where $n \neq m$), (2) different tasks (source tasks are supervised learning (SL) tasks and target tasks are reinforcement learning (RL) tasks), and (3) different texture appearance (source and target tasks are both RL tasks, while source tasks are to solve original Sokoban instances and target tasks are to solve Sokoban instances with different texture appearance). The agent was first pre-trained on source tasks and then fine-tuned on target tasks. RL tasks are to solve 100 randomly generated n-boxes Sokoban instances. SL tasks are to recognize the location of the agent in Sokoban instances.

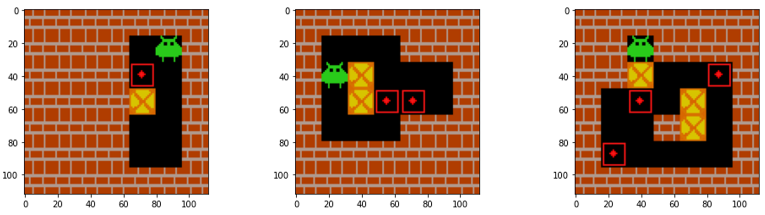


Fig. 2. Examples of Sokoban instances, increasing in difficulty from 1 box and 1 target to 3 boxes and 3 targets (Color figure online)

The overall statistics of the maps are shown in Fig. 3. As the number of objectives increases, the number of steps for the optimal solution also increases, and so does the difficulty of solving the game.

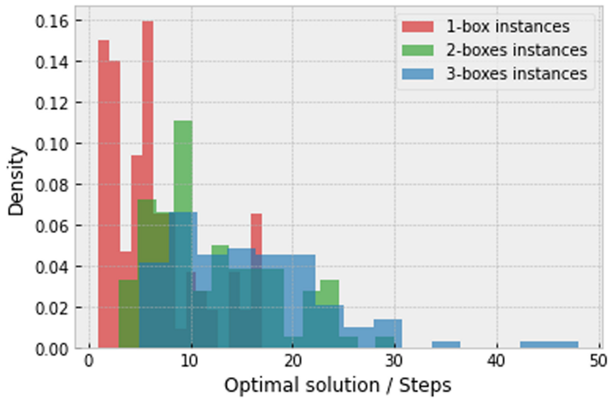


Fig. 3. Distribution of optimal solutions in different Sokoban instances.

3.1 Neural Network Architecture

The neural network we employ is taken from the DeepMind baseline [22] directly without hyper-parameter tuning. The model consists of 3 convolutional (Conv) layers with kernel size 8×8, 4×4, 3×3, strides of 4, 2, 1, and number of output channels 32, 64, 64. This is followed by a fully connected (FC) hidden layer with 512 units. The outputs of this FC layer will be fed into two heads: one for outputting the policy logits and one for outputting the state value. This is one of the most commonly-used architectures in RL, we selected it also in order to show what can be achieved with popular architecture. Details of architecture and hyper parameters we employ are found in Table 1.

Table 1. Hyper-parameters of the neural network and training.

Learning rate	$7 \cdot 10^{-4}$
Discount factor	0.99
Entropy coefficient	0.1
Value loss coefficient	0.5
Eps in RMSprop	10^{-5}
Alpha in RMSprop	0.99
Rollout storage size	5
No. of environments for collecting trajectories	30

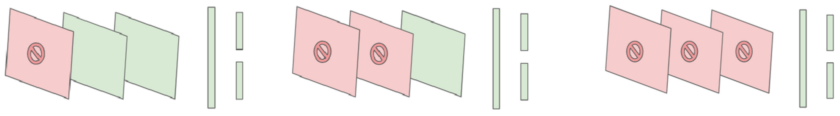


Fig. 4. Three different transfer approaches, red layers are fixed while green layers are trainable. They correspond $k = 1, 2, 3$ from left to right respectively. (Color figure online)

3.2 Transfer Approach

The main idea of our transfer approach is to reuse feature representations from source tasks learned by the Conv layers in new unseen target tasks. As detailed in the last sub-section, our model consists of 3 Conv layers and 2 FC layers. The feature representations were transferred to new tasks by copying the weights of the first k Conv layers trained in source tasks (where there are n_s boxes/targets) to initialize the new learning model in target tasks (where there are n_t boxes/targets). Then we froze these weights (they were no longer trainable) and retrained the remaining part of the model. In our experiments, $k \in \{1, 2, 3\}$, $n_s \in \{1, 2, 3\}$, $n_t \in \{1, 2, 3\}$. Please refer to Fig. 4 for an explanation of this approach. Different squares represent different layers of our neural network. The first 3 layers are Conv layers and the last two are FC layers. Reds are weights taken from pre-trained model and fixed, greens are weights reinitialized and trainable.

Solved ratios were used for evaluating agents, and evaluation executes every 1,000 environment steps. 20 randomly selected test instances were performed by the current learning agent. We say the transfer is *positive* when the performance with the transfer is better than without (training from scratch), and *negative* when the performance with the transfer is worse than without.

4 Experiments

We designed experiments with different source, target tasks and k, in order to verify the hypotheses we proposed. We experimented with Sokoban instances with 1, 2, and 3 boxes. All experiments were run for 1 million environment steps. We use abbreviations for each experiment. For instance, **s1t1k1** means **s**ource tasks are **1**-box instances, **t**arget tasks are **1**-box instances and we transfer and fix the **1** (first) layer. Exceptions are **sPt1k1** and **s1t1fc_game2**. **sPt1k1** stands for the source task is a supervised learning prediction task, and target task is the RL task over 1-box instances while we only keep the first layer. **s1t1fc_game2** is that the source and target tasks are both RL tasks over 1-box instances, but we transfer fully connected layers to instances with different appearance. The neural networks were trained using Advantage Actor Critic (A2C), a single threaded variant of A3C [18]. All experiments were performed 5 times with different random seeds, and figures were drawn using averaged results with 0.95

confidence interval. Heavy fluctuations were caused by irreversible actions, one irreversible action during the game could make the whole game unsolvable.

4.1 Transfer Among Related Tasks

Related tasks are tasks where the only difference between source and task is the difficulties of instances, i.e. the number of boxes and targets. (Recall that both source and task are trained on 100 different map-layouts, in all experiments.)

Figure 5 and Fig. 6 show results for training on 1-box, 2-boxes, 3-boxes instances with reusing features learned in different tasks, and we fix $k = 3$. All results showed that transferring feature representations learned in single-box instances is positive. Performance of agents (s1t1k3, s1t2k3, s1t3k3) who are using features learned from single-box instances always outperform other agents, including agents training from scratch and using features learned from other instances. The transfer, however, is not 'bi-directional', feature representations learned in multiple-box instances could not be successfully transferred to the learning in single-box instances. Their performance (s2t1k3, s3t1k3) converged to a relatively low solved ratio, which indicates that transferred features are not suitable for single-box instances. Just as humans learn more general knowledge in simpler cases, our agents also showed that the knowledge learned from single-box instances is more general and transferable than ones learned in multiple-box instances.

To further enhance performances of transferring features learned in single-box instances, we tried different k. We expected that the performance will be the best when $k = 1$ since the first layer learn the most general features. However, the results in Fig. 7 instead show that not $k = 1$ but $k = 2$ (s1t2k2, s1t3k2)

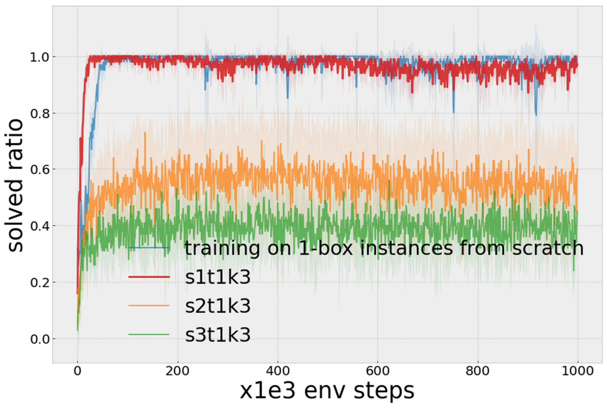
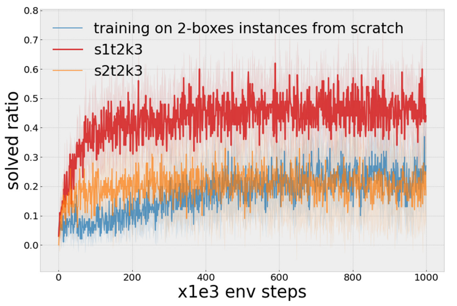


Fig. 5. Performance of transferring feature representations learned in 1-box, 2-boxes, 3-boxes instances to learning in 1-box with $k = 3$. $n_s = 1, 2, 3$, $n_t = 1$, $k = 3$. Pre-training on 1-box instances is much better than pre-training on 2 or 3 box instances when training new 1-box instances.



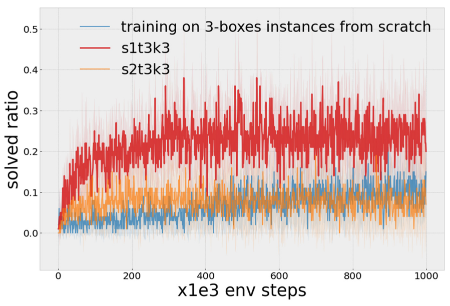


Fig. 6. Performance of transferring feature representations learned in 1-box, 2-boxes, 3-boxes instances to learning in 2-boxes (left) and 3-boxes (right) with $k = 3$. $n_s = 1, 2, 3$, $n_t = 2, 3$, $k = 3$.

perform the best. Similar to [6], features learned in the first 2 layers are still general enough for transfer; in addition, the difference between source tasks and target tasks is not as large as expected, and features learned between different instances are more overlapping than expected.

It is also interesting to see the influence of how many layers are fixed on the success of the transfer. In particular, we want to know whether a smaller k could change the negative transfer from multiple-box instances to single-box instances into positive. (We believe features from multiple and single-box instances are overlapping to some extent.) Results are shown in Fig. 8. We see that indeed the first layer (s2t1k1, s3t1k1) did learn enough general features from multiple-boxes instances to solve the single-box instances. Although agents with features only learned by the first layer could converge to decent performance in the end, the transfer is still negative. An interesting point is that $k = 3$ (s2t1k3) performs better than $k = 2$ (s2t1k2) when source tasks are 2-boxes instances. Note that $k = 2$ (s3t1k2) performs better than $k = 3$ (s3t1k3) when source tasks are 2-boxes instances. There are more overlapping features between the 2-boxes instances and single instances.

4.2 Transfer Among Different Tasks (SL/RL)

Feature representations learned from previous tasks can either be helpful or harmful. In the previous subsection we saw some positive transfer to related Sokoban tasks, in this subsection we study if transfer between supervised and reinforcement learning tasks works. We follow prior work, Anderson et al. [1] showed that features can be transfered from hand-crafted supervised learning (SL) tasks to reinforcement learning (RL). Their model was first trained to predict state dynamics of the environment, and then pre-trained hidden layers were helpful to accelerate solving RL tasks.

For transfer to different (randomly chosen) instances in Sokoban, we also formed a supervised task, which was to train a prediction model to recognize the location of the agent, shown in Fig. 9a. When humans are solving Sokoban,

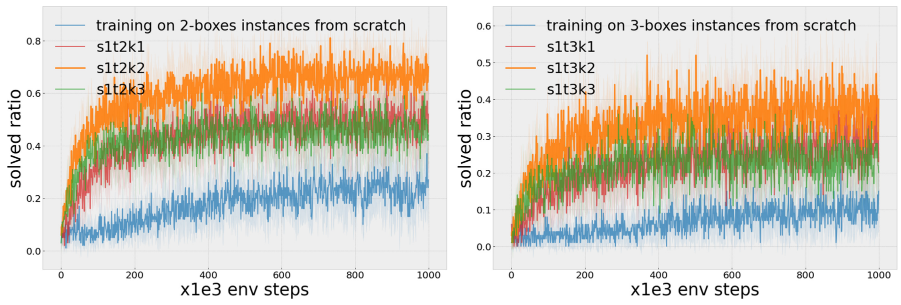


Fig. 7. Performance of transferring feature representations learned in 1-box instances to learning in 2-boxes (left) and 3-boxes (right) with different k. $n_s = 1$, $n_t = 2, 3$, $k = 1, 2, 3$.

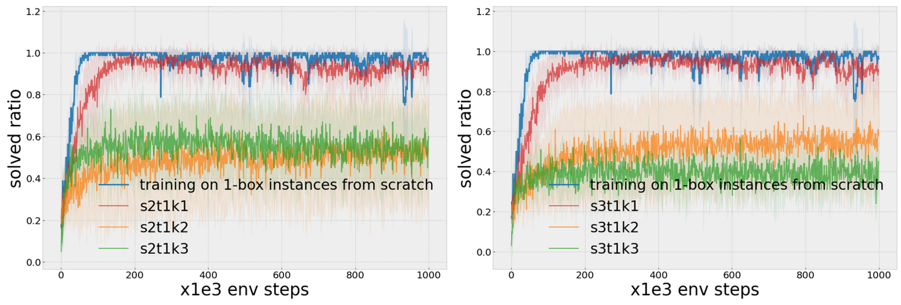
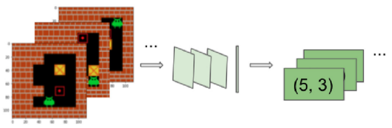


Fig. 8. Performance of transferring feature representations learned in 2-boxes (left) and 3-boxes (right) instances to learning in 1-box instances with different k. $n_s = 2, 3$, $n_t = 1$, $k = 1, 2, 3$.

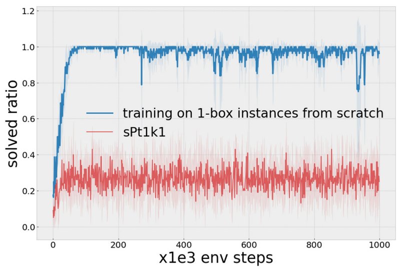
we first need to know where the agent is before we draw up a plan. If we already know the location of objectives, the solving process could be faster. After the prediction model could correctly recognize where the agent is, we took feature representations of the trained model and plug them into a new agent. The first layer of learned features is fixed, and we only train the remaining part. Figure 9b shows the performance of transferring and training from scratch. We find negative transfer for (sPt1k1): the performance is much worse compare with training from scratch.

4.3 Transfer to Different Appearance

Experiments we described in previous subsections were all trying to transfer Conv layers which learned feature representations. In the next experiment, we try to make the agent utilize another part of the learned model, which are back FC layers of the whole model. The source and target tasks were both single-box instances, but the target tasks were instances with different appearances. Figure 10b is an example. The maps used for two groups of tasks were the same,



(a)



(b)

Fig. 9. (a): How SL tasks work. Input states and neural network will learn to predict locations of the agent. (b): Performance of training from scratch and training with transferred feature representations from SL tasks.

the only difference was how they look like, the appearance was changed, with different textures, and we call it Game2. Figure 10a shows the transfer approach. We took FC layers trained in source tasks and fixed them, and retrained the remaining Conv layers. Since maps were the same, solutions of the instances were the same. When Conv layers learn new feature representations successfully, instances are solved then.

Figure 11a shows the performance. One would expect that transferred FC layers (s1t1fc_game2) are faster because the agent only needs to learn new feature representations. However, the experiments did not show this result. Apparently, when the whole model is trained jointly, it has more flexibility to be trained into the final shape; when the last part of the model is fixed, the learning of the first part will be trying to cater for the last part in order to solve the problem, which made the learning slower.

4.4 Visualizing Agent Detection

In order to better understand what the network learned, we provide a visualization. We follow Yosinski et al. who showed that convolutional neural networks can detect latent objectives without explicit labels [31]. We visualized a feature map of a trained neural network on 1-box RL tasks. Figure 11b shows the latent 'agent detector' for Sokoban. The neural network automatically learned to detect the agent without giving any labels or information. Left rows are pixel inputs, right rows are outputs of one specific feature map. Yellow-green units are detected agents. We note that although the network was trained in single-box instances, it still performed quite well in multiple-box instances, which is a potential reason for the successful transfer. The agent's abilities that were learned in source tasks are useful in target tasks.

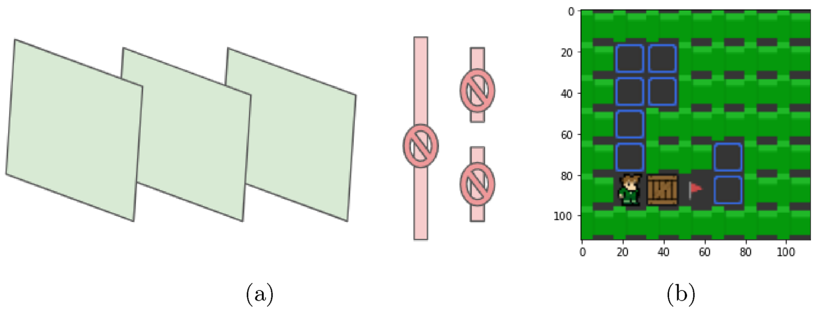


Fig. 10. (a): Transfer approach for transfer to Game2. FC layers are taken from previously training and fixed, only conv layers will be retrained. (b): An example instance in Game2. We changed appearances in Game2 with different textures of objectives.

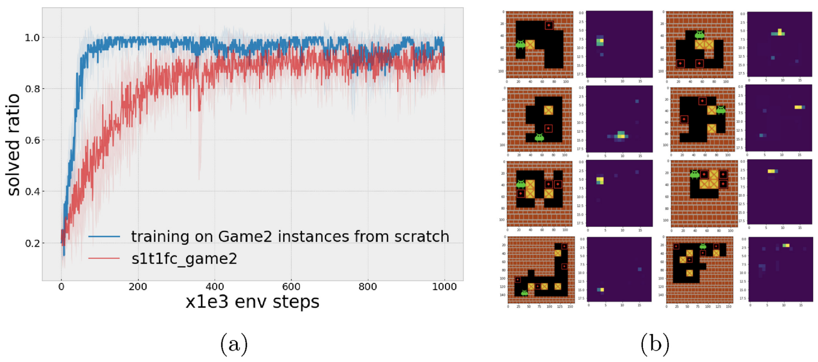


Fig. 11. (a): Training on Game2 using transferred FC layers. Its performance is worse than training from scratch. (b): The agent detector. Outputs of the twenty third feature map of the first convolutional layer, which is an agent detector learned from 1-box instances, and it's still usable in multiple-boxes scenarios. (Color figure online)

5 Conclusion and Future Work

Our experiments showed that in a reinforcement learning setting the agent in Sokoban can learn four characteristics that are similar to humans. (1) Feature representations learned previously can accelerate the new learning in other Sokoban instances. Knowledge learned in previous related tasks could be reused to accelerate new learning, transfer learning is occurring, creating an implicit learning curriculum. (2) Feature representations learned in single-box instances are more general, and are more effective for learning in multiple-boxes instances, but not vice versa. Knowledge learned in simpler tasks is more general and more effective, even in more complex tasks. Further experiments showed negative learning, that confirms these results. (3) Feature representations learned in unrelated supervised learning tasks can hurt fine-tuning performance. If the learned knowledge is required to be helpful in new coming tasks, it's better to learn from similar tasks, otherwise the choice of tasks needs to be careful. (4) Fixing the top-fully-connected layers and retraining the bottom convolutional layers slows down learning and hurts performance. We conclude that learning should have explicit order, less flexibility will not only be unhelpful but also hurt the learning process and the performance.

Our experiments showed that with a simple 5-layer convolutions/fully connected network (based on DeepMind's baseline [22]), transfer learning and curriculum learning of behavior to occur in Sokoban. This is surprising, since Sokoban is a planning-heavy problem, for which one would expect more elaborate network architectures to be necessary. Reusing pre-trained feature representations in RL fields is not well studied, and to the best of our knowledge, these are the first results show transfer learning and curriculum learning with such a simple network in such a planning-heavy behavioral task. In the future, we would like to see more utilization of pre-trained feature representations and of the entire pre-trained model in RL. We believe that reusing pre-trained model can significantly improve data-efficient reinforcement learning.

Acknowledgement. The financial support to Zhao Yang is from the China Scholarship Council (CSC). Computation support is from ALICE and DSLab. The authors thank Hui Wang, Matthias Müller-Brockhausen, Michiel van der Meer, Thomas Moerland and all members from the Leiden Reinforcement Learning Group for helpful discussions.

References

1. Anderson, C.W., Lee, M., Elliott, D.L.: Faster reinforcement learning after pre-training deep networks to predict state dynamics. In: 2015 International Joint Conference on Neural Networks (IJCNN), pp. 1–7. IEEE (2015)
2. Badia, A.P., et al.: Agent57: Outperforming the Atari human benchmark. In: International Conference on Machine Learning, pp. 507–517. PMLR (2020)
3. Brown, T.B., et al.: Language models are few-shot learners. arXiv preprint arXiv:2005.14165 (2020)
4. Brys, T., Harutyunyan, A., Taylor, M.E., Nowé, A.: Policy transfer using reward shaping. In: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, pp. 181–188 (2015)
5. Cook, M., Raad, A.: Hyperstate space graphs for automated game analysis. In: IEEE Conference on Games, CoG 2019, London, United Kingdom, 20–23 August 2019, pp. 1–8. IEEE (2019). https://doi.org/10.1109/CIG.2019.8848026
6. De la Cruz, G., Du, Y., Irwin, J., Taylor, M.: Initial progress in transfer for deep reinforcement learning algorithms. In: 25th International Joint Conference on Artificial Intelligence (IJCAI), vol. 7 (2016)
7. Cruz, G.V., Jr., Du, Y., Taylor, M.E.: Pre-training neural networks with human demonstrations for deep reinforcement learning. arXiv preprint arXiv:1709.04083 (2017)
8. Culberson, J.: Sokoban is PSPACE-complete (1997)

9. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
10. Dor, D., Zwick, U.: Sokoban and other motion planning problems. Comput. Geom. **13**(4), 215–228 (1999)
11. Feng, D., Gomes, C.P., Selman, B.: A novel automated curriculum strategy to solve hard Sokoban planning instances. Adv. Neural. Inf. Process. Syst. **33**, 3141–3152 (2020)
12. Feng, D., Gomes, C.P., Selman, B.: Solving hard AI planning instances using curriculum-driven deep reinforcement learning. CoRR abs/2006.02689 (2020). https://arxiv.org/abs/2006.02689
13. Fernández, F., García, J., Veloso, M.: Probabilistic policy reuse for inter-task transfer learning. Robot. Auton. Syst. **58**(7), 866–871 (2010)
14. Guez, A., et al.: An investigation of model-free planning. In: International Conference on Machine Learning, pp. 2464–2473. PMLR (2019)
15. Guez, A., et al.: Learning to search with MCTSnets. In: International Conference on Machine Learning, pp. 1822–1831. PMLR (2018)
16. Hamrick, J.B., et al.: On the role of planning in model-based deep reinforcement learning. In: 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, 3–7 May 2021 (2021)
17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
18. Mnih, V., et al.: Asynchronous methods for deep reinforcement learning. In: International Conference on Machine Learning, pp. 1928–1937. PMLR (2016)
19. Nair, A., McGrew, B., Andrychowicz, M., Zaremba, W., Abbeel, P.: Overcoming exploration in reinforcement learning with demonstrations. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 6292–6299. IEEE (2018)
20. Ontanón, S., Synnaeve, G., Uriarte, A., Richoux, F., Churchill, D., Preuss, M.: A survey of real-time strategy game AI research and competition in StarCraft. IEEE Trans. Comput. Intell. AI Games **5**(4), 293–311 (2013)
21. Plaat, A.: Learning to Play: Reinforcement Learning and Games. Springer, Heidelberg (2020). https://learningtoplay.net
22. Racanière, S., et al.: Imagination-augmented agents for deep reinforcement learning. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 5694–5705 (2017)
23. Schrader, M.P.B.: Gym-Sokoban (2018). https://github.com/mpSchrader/gym-sokoban
24. Silver, D., et al.: Mastering the game of go without human knowledge. Nature **550**(7676), 354–359 (2017)
25. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
26. Spector, B., Belongie, S.: Sample-efficient reinforcement learning through transfer and architectural priors. arXiv preprint arXiv:1801.02268 (2018)
27. Sutton, R.S., Barto, A.G.: Reinforcement Learning, An Introduction, 2nd edn. MIT Press, Cambridge (2018)
28. Taylor, M.E., Stone, P.: Transfer learning for reinforcement learning domains: a survey. J. Mach. Learn. Res. **10**, 1633–1685 (2009)
29. Vinyals, O., et al.: Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature **575**(7782), 350–354 (2019)

30. Xu, W., He, J., Shu, Y.: Transfer learning and deep domain adaptation. In: Advances in Deep Learning. IntechOpen (2020)
31. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? Advances in Neural Information Processing Systems, pp. 3320–3328 (2014)