On Neural Consolidation for Transfer in Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Although transfer learning is considered to be a milestone in deep reinforcement learning, the mechanisms behind it are still poorly understood. In particular, predicting if knowledge can be transferred between two given tasks is still an unresolved problem. In this work, we explore the use of network distillation as a feature extraction method to better understand the context in which transfer can occur. Notably, we show that distillation does not prevent knowledge transfer, including when transferring from multiple tasks to a new one, and we compare these results with transfer without prior distillation. We focus our work on the Atari benchmark due to the variability between different games, but also to their similarities in terms of visual features.

1 Introduction

In spite of the rapid progress made in Deep Reinforcement Learning in the last decade, and although state-of-the-art algorithms are more and more efficient, many fundamental issues still have not been solved and remain major limitations in current approaches. In particular, existing algorithms train networks from scratch on each new task, which is very computationally costly. This issue motivated the development of transfer learning [Pratt et al., 1991, Taylor and Stone, 2009], the study of how to transfer and reuse knowledge from a neural network to another in order to accelerate learning and benefit from previously acquired abilities. Various methods for transfer have been proposed over the years, from simple ones such as fine-tuning, to more complex ones such as using distillation in a multi-task setting [Rusu et al., 2016a].

Although primarily developed for network compression [Bucilua et al., 2006], *distillation* is a technique that aims at copying the behavior of a *teacher* neural network into a *student* one by ensuring the two represent the same function. It has been successfully used to compress multiple teachers in a single student, thus achieving multi-task learning [Hinton et al., 2015]. We argue that distillation in a multi-task context, which we refer to as *network consolidation*, is useful for knowledge transfer and can help understand the underlying mechanisms behind transfer. More specifically, we compare different methods to achieve consolidation on multiple tasks in an efficient manner and discuss the importance of key details in the algorithmic design. We then study the effect of the final performance of the consolidated network and show that transfer can occur even when the consolidation process does not reach convergence. Finally, we argue that consolidation does not prevent transfer in general and therefore can be used to parallelize transfer mechanisms.

In order to study these claims, we propose a set of experiments based on the use of the AMN algorithm [Parisotto et al., 2016] to alternate between training and consolidation phases. Our approach is motivated by current neurobiological theories about how knowledge transfer and lifelong learning work in the mammalian brain, such as the Complementary Learning Systems theory [McClelland et al., 1995, CLS]. CLS states that memorization is based on two distinct parts of the brain: the

Submitted to the 5th Workshop on Meta-Learning at NeurIPS 2021, held virtually. Do not distribute.

hippocampus, responsible for short-term adaptation and rapid learning, and the neocortex, which assimilates this knowledge slowly and retains it on a long-term basis. We try to emulate this by dividing learning into two phases.

We present an overview of the state-of-the-art regarding transfer learning in Section 2 before describing our experimental setup in Section 3. Then Section 4 discusses the effect of using the AMN algorithm for consolidation on networks' performance while Section 5 focuses on knowledge transfer. Finally, we try to better understand the different mechanisms behind transfer in Section 6.

2 Background and Related Work

To achieve transfer learning, one of the most explored method has been the use of distillation, as proposed by Rusu et al. [2016a] and since extended multiple times. For instance, Parisotto et al. [2016] and Jung et al. [2016] add an incentive to also copy the features in order to guide the training process, while Teh et al. [2017] build a central network encoding common behaviors to share knowledge between tasks.

One major challenge in RL today is lifelong learning, i.e. how to solve different tasks sequentially while avoiding *catastrophic forgetting*. Different approaches exist to tackle this problem that Parisi et al. [2019] propose to divide in to three categories. One possibility is to periodically modify the network architecture when facing new tasks in order to enhance its representative power [Yoon et al., 2018, Rusu et al., 2016b, Fernando et al., 2017]. Another approach is to use regularization to preserve previously acquired knowledge [Li and Hoiem, 2018, Kirkpatrick et al., 2017, Zenke et al., 2017]. Finally, the lifelong problem can be reduced to a multi-task learning setup by using a rehearsal strategy, memorizing every task encountered [Lopez-Paz and Ranzato, 2017, Rebuffi et al., 2017, Kaiser et al., 2020, Ha and Schmidhuber, 2018, Shin et al., 2017]. These three main categories are not mutually exclusive, and many of these algorithms make use of different techniques that belong to two categories.

The idea of alternating between an active phase of learning and a passive phase of imitation as inspired by the CLS has also been explored before. In particular, Berseth et al. [2018] introduce the PLAID algorithm that progressively grows a central network using distillation on newly encountered tasks. Similarly, Schwarz et al. [2018] successively compress different expert networks in a *knowledge base* that is then reused by new experts via lateral layer-wise connections as introduced by Rusu et al. [2016b].

Instead of learning to solve multiple tasks, another possibility is to learn how to be efficient at learning: this is the meta-learning approach. One intuitive way to achieve that is by using a meta-algorithm to output a set of neural network weights which are then used as initialization for solving new tasks [Finn et al., 2017, Nichol et al., 2018]. On the other hand, Beaulieu et al. [2020] propose the use of a second network whose role is to deactivate part of a classical neural network. By analogy with the human brain, this network is called the neuromodulatory network as it is responsible for activating or deactivating part of the main network depending on the current task to solve. Finally, He et al. [2020] propose a framework for meta-algorithms which divides them into a "What" part whose objective is to identify the current running task from context data, and a "How" part responsible for producing a set of parameters for a neural network that will be able to solve this task.

3 Actor-Mimic Networks for consolidation in Lifelong Learning

In order to study the consolidation process and its interaction with knowledge transfer, we explore the use of the Actor-Mimic (Network) algorithm [Parisotto et al., 2016, AMN] that acts as a policy distillation algorithm with an additional incentive to imitate the teacher's features. In classical policy distillation, as proposed by Rusu et al. [2016a], the distilled network — also called student network — learns to reproduce the output of multiple expert networks (policy regression objective) using supervised learning. In addition, the AMN algorithm adds another feature regression objective that regularizes the features of the student network (defined as the outputs of the second-to-last layer) towards the features of the experts. Intuitively, the policy regression objective teaches the student *how* it should act while this feature regression objective teaches the result of the expert's "thinking process" that indicates *why* it should act that way.

The AMN algorithm makes it possible to consolidate several expert networks at the same time while extracting features containing the same information as the experts'. As the target tasks can be quite different especially on a low-level point of view (e.g. color palette), these extracted features should be quite high-level and thus hopefully generalizable. We use this property to propose a new training protocol composed of two main phases that emulate the day-night cycles: an active learning phase in which neural networks — that we call "expert networks" — are trained individually on a set of tasks, and a passive learning phase in which the knowledge acquired by all these experts is consolidated into a central common Actor-Mimic Network responsible for maintaining knowledge in the long term.

During the active phase, each expert network is trained on its corresponding task using a classical Reinforcement Learning algorithm; in our case we used Rainbow [Hessel et al., 2018]. These phases are interrupted early on before performance reaches state-of-the-art levels, as the objective is to extract general features that will encourage the next experts to avoid focusing on task-specific pixel-level characteristics. The passive phase consists in consolidating an AMN from these experts. Training using distillation is more sample efficient than usual RL methods [Rusu et al., 2016a], therefore the AMN only needs a fraction of the number of time steps of the active phase to exhibit the same performance as the experts. The next active phase is then started by initializing the new expert with the AMN weights, a process we describe further and discuss in Section 4.2, before repeating this procedure several times.

We evaluate this protocol on the Atari benchmark [Bellemare et al., 2013, Machado et al., 2018], and more precisely on the games Breakout, Carnival, Pong, SpaceInvaders and VideoPinball, selected for their diversity and their balanced difficulty. Although the choice of these specific games may limit our analysis, we find this benchmark interesting in that some of these games can appear to human players as similar (e.g. hit a ball moving in straight lines with a paddle) but are different from a visual perspective. We follow the choices of Castro et al. [2018] to report the performance of the experts during training by averaging the return on every completed episode during iterations of 50000 time steps. We describe the AMN performance in terms of percentage of the teachers' final performance. For visibility reasons, on each experiment we only report the results on three games as it usually encompass every interesting behavior we discuss, but all the remaining graphs can be found in Appendix B. All experimental details can be found in Appendix A.

4 Improving performance via consolidation

Even with the AMN algorithm to perform the consolidation phase, many questions still remain about how to perform the whole training process. In particular, we discuss here different possible methods to carry out the passive phase (Section 4.1) and knowledge transfer between a passive and an active phase (Section 4.2).

4.1 On the passive consolidation phase

Although the AMN algorithm makes it possible to consolidate several expert networks during the same training phase, different methods exist to train a single network to imitate the output of several different expert networks *simultaneously*. An intuitive approach to that problem is simply to minimize a single loss that is the sum of the AMN losses for each expert network. In that setting, one gradient descent step will try to minimize each individual loss at the same time, ensuring the simultaneity.

However, this approach has a drawback: two tasks could theoretically result in opposite gradient directions that would cancel one another, preventing the consolidated network to improve on any of these tasks. This issue is studied in more details by Yu et al. [2020] who show that this situation can occur frequently under the right circumstances. In our case though, this issue did not occur and the use of a single composite loss gave satisfying results. Figure 1 compares the performance when optimizing a single composite loss and when optimizing the separate losses when switching tasks at the end of each episode.

Instead of training the student network on each task simultaneously, another possibility is to alternate between them and optimize the different losses sequentially. This approach solves the issue of losses of different orders of magnitude, as we use the Adam optimizer which modifies the gradients to be on the same scale as the learning rate; consequently no task can be favoured by the optimization process. However, this method also introduces a new critical hyperparameter: the frequency with which to

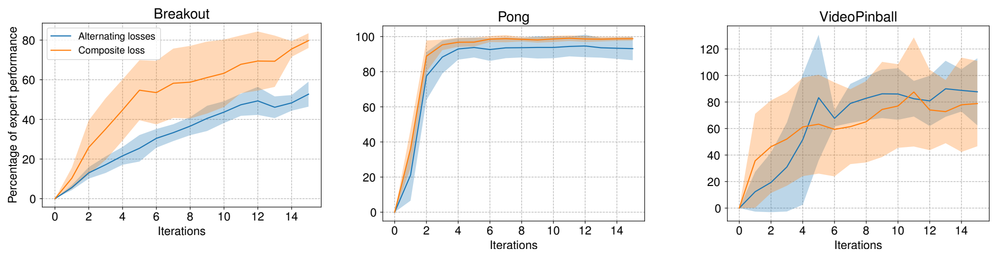


Figure 1: Performance of an AMN network consolidated on 5 experts when minimizing the composite loss vs each individual loss sequentially and measured in percentage of the experts final score. Each experiment is repeated 3 times; the shaded area corresponds to one standard deviation around the mean.

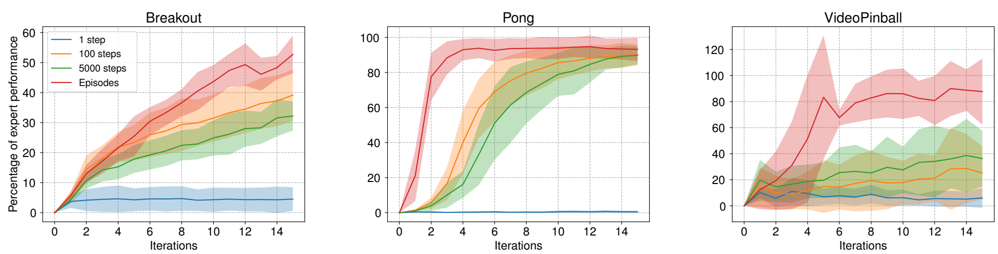


Figure 2: Performance of the AMN network measured in percentage of the experts' final score. The gradient descent process switches between tasks every 1, 100, 5000 steps or every episode.

switch from one task to the next. One possibility is to switch after each episode on any task, although one main drawback in that case is that episodes can greatly vary in length between tasks and even within tasks (between early phase and late phase of training), thus the learning process can become very imbalanced. Figure 2 compares four different values for this hyperparameter: a high frequency switch (every time step), a medium frequency (every 100 time steps), a low frequency (every 5000 steps) and switching every episode.

The experiments show that switching every time step prevented any learning of the AMN, and the resulting policy did not appear to be a good initialization point for the next active phases, suggesting that no interesting features were extracted during training. For both medium and low frequency switching, the results were quite dependent on the given tasks: for certain games, the network could not replicate the experts performance after the passive phase, whereas for others it was quickly able to reach the same score. Finally, despite the imbalance of number of steps per episode, it appears that switching only at the end of full episodes results in better performance on every task. These results show that switching too often between tasks prevents the optimization process convergence, suggesting that contrary to when using the composite loss, here the different gradients might cancel each other out.

4.2 Transfer from passive phase to active phase

Once the passive phase is finished, the AMN has the same performance as the experts of the previous active phase. Our objective is now to transfer knowledge from this AMN to new experts on the next active phase, and one obvious technique to achieve that is just by using the weights of the AMN as an initialization for the experts. In the case where the AMN has a larger number of outputs than an expert, we simply mask the supplementary weights and copy only the fitting subpart of the network. To study the contribution of the passive phase, we evaluate transfer from an AMN consolidated on 5 games to an expert that resumes training on one of these 5 games using the AMN as initialization.

Figure 3 compares the average returns per episode between the expert network trained on the first active phase (so initialized randomly) and the one trained on the second active phase (initialized with

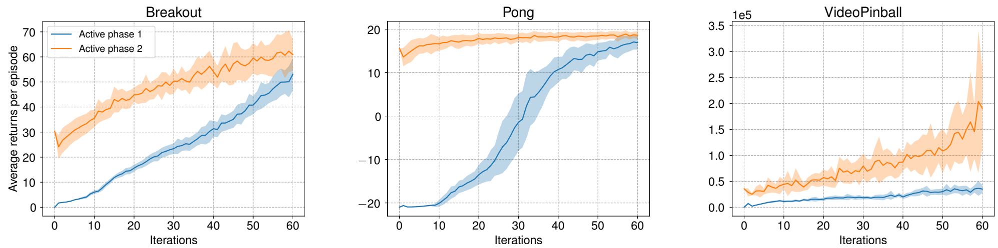
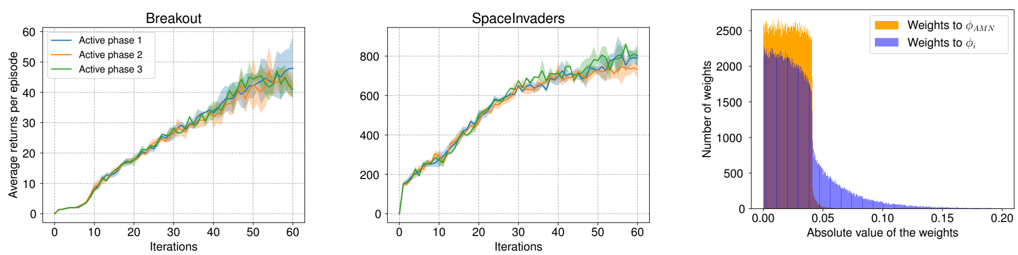


Figure 3: Average return per episode for an expert initialized randomly (active phase 1) or from an AMN after a passive phase (active phase 2)



(a) (b) (c)

Figure 4: Graphs 4a and 4b show the average return per episode for experts trained during different active phases. Graph 4c highlights the weight distribution between an expert output layer and its feature layer (ϕ_i) or the AMN feature layer (ϕ_{AMN}) in the case of an expert trained on Pong.

the AMN weights). This experiment shows that the transfer has a significant jumpstart effect as the second expert networks all start above the initial value of a random policy. This is not surprising as the AMN is close to the performance of the initial expert at the end of the passive phase, and the weight copy preserves the policy so the new expert starts with at least that knowledge. However, the long term effect of this initial boost in performance is largely dependent on the task tackled, and sometimes it disappears quickly (Breakout) or keeps a certain advantage during the whole training (VideoPinball).

One issue with copying the AMN weights is that if two tasks are very different from one another, the features extracted on one task can be detrimental to the other, leading to negative transfer. Instead of forcing the feature initialization, a more complex approach is to make the features accessible by the expert via a lateral connection from the AMN feature layer to the expert output layer. That way, if the previously learned features are useful for tackling new tasks, they are easily accessible for the network and can accelerate the training process, but the expert can also develop entirely new features specifically crafted for the new task.

Although this approach seems more flexible than the simple duplicate of the AMN weights, in our case it didn't yield positive results. Figure 4 shows the evolution of the average return per episode for experts trained during successive active phases: surprisingly the training is not faster in the expert with the lateral connections. This can mean that the features are not interesting — but the AMN is also able to play the different games so this hypothesis doesn't hold — or that the expert network is just not using these features. We verify this hypothesis by analysing the value of the weights between the output layer and the randomly initialized ϕ_i or the AMN features ϕ_{AMN}, and we plot the histogram of their absolute values in figure 4c. It shows that every weight linking to the AMN features has a very low magnitude, confirming that the expert doesn't use previous features at all compared to the new features developed during the active phase. These results suggest that freezing the AMN layers is too constraining for the network to actually reuse this knowledge, and in the end adding a lateral connection was not effective at transferring information.

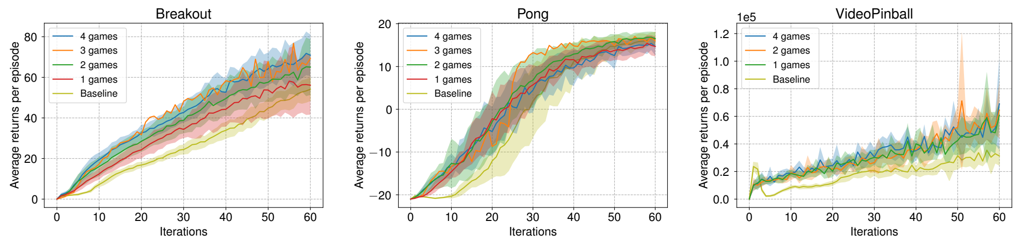


Figure 5: Performance of experts initialized from an AMN trained on varying numbers of games during the passive phase. The baseline curve corresponds to an expert initialized randomly.

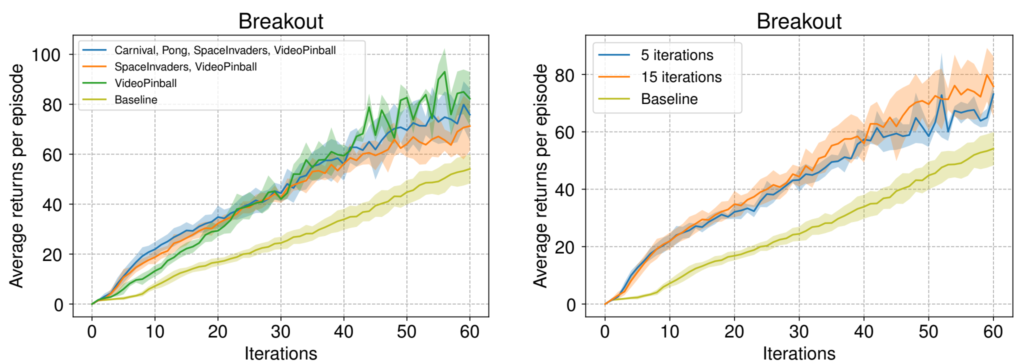


Figure 6: Performance of an expert trained on Breakout and initialized from an AMN trained on different subsets of games (left), or during a different number of iterations (right)

5 Consolidation for Transfer Learning and Domain Generalization

In the previous section, we analyzed the effects of consolidation when each active phase trains on the same set of games in order to compare training with and without extracted features. Our objective was twofold: first, show that the AMN algorithm is able to extract general features useful for several tasks at the same time, and secondly prove that starting training with access to these features can help improve the optimization process in any way. In this section, we now train each active phase on a completely new set of tasks to measure the generalizability and transferability of these features.

5.1 Consolidation towards new unseen tasks

The main purpose of extracting general features in the lifelong learning context is to be able to reuse them in diverse contexts in order to avoid learning from scratch each time. To evaluate the transferability of the knowledge acquired by the AMN during the passive phases, we modified our experiments so that the first and second active phases don't share any game. First of all, we measured the effect of varying the number of games to consolidate on during the first passive phase to verify if consolidating on more games could lead to more general features. Figure 5 compares the performance of the second active phase on different games when initialized by an AMN consolidated on subsets of 1, 2, 3 or 4 different Atari games. Most games exhibit a very small jumpstart effect in the beginning of the training, compared to an agent trained without consolidation, but the number of games tackled during the first active phase doesn't seem to have any impact on the performance of the new experts. The only exception is Breakout on which the effect of consolidation scales almost linearly depending on the number of games, which would tend to show that first, the expert can benefit from the learned features and that second, these features are more useful when they are extracted from multiple games.

However, the transfer effect appears to be more dependent on the games chosen for the consolidation step than on the number of games. In the case of Breakout, only the presence of VideoPinball in the

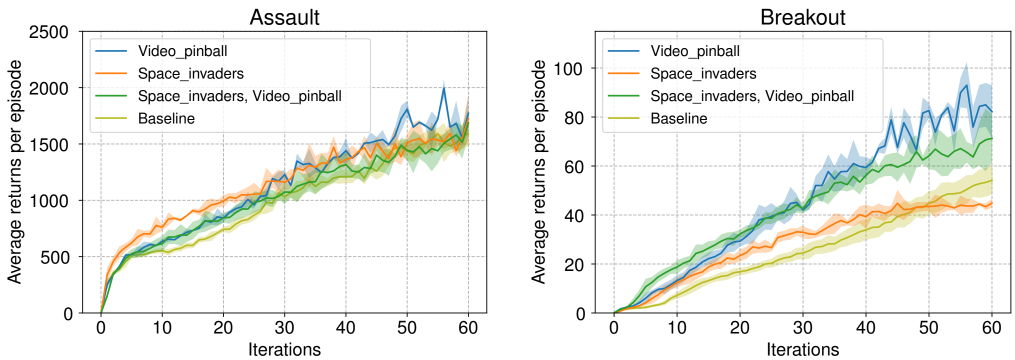


Figure 7: Performance of experts on Assault (left) and Breakout (right) initialized from an AMN trained on different sets of games during the passive phase. The baseline corresponds to an expert initialized randomly.

set of basis games have a significant effect on the newly trained experts. Figure 6 (left) compares three agents trained on VideoPinball alone, with one or with three additional games, and in each case the Breakout expert reaches the same score with very little variation, except a slight jumpstart that quickly disappears.

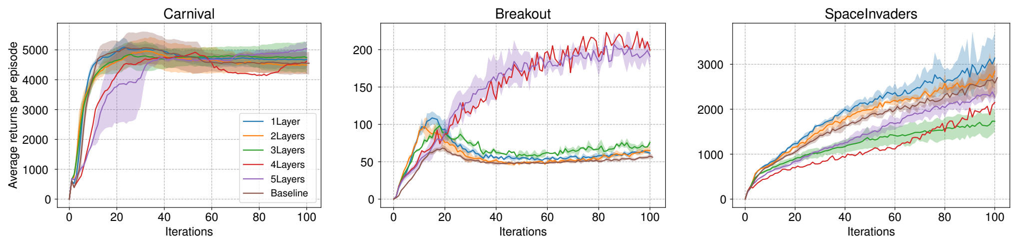
Once again, this shows that transfer is independent of how similar the games seem to be from a human point of view: a common feature between the two games is that in each case, the gameplay revolves around hitting a ball at the bottom of the screen, but the means of hitting the ball are very different as the player manipulates a pad in Breakout and flippers in VideoPinball. The game physics and the color palette (except the black background) are also quite distinct. In all cases, Figure 6 proves that something is definitely transferred between these two specific games despite the dissimilarities.

In all these experiments, the feature extraction process does not seem to require the convergence of the AMN. In the previous section, the final performance at the end of the passive phase was directly reusable by the new expert so the observed jumpstart effect scaled monotonously in relation to it. However, when transferring to new tasks, the policy has very low chances of being transferable as is, therefore the AMN final score is not as important as the features it managed to extract. To measure the importance of the AMN results, we drastically shorten the passive phases to only 5 iterations instead of 15 (Figure 6, right). In that limited amount of time, the AMN is only able to reach 75% of the VideoPinball expert performance, while it otherwise reaches around 105%. Still, in this configuration this drop in performance was not transmitted to the expert training on Breakout, as can be seen on Figure 6 (right). We reproduced this experiment on different pairs of games, and in each case the final score of the AMN plays a limited role in the success of knowledge transfer, reinforcing the idea that the improvement comes from a feature transfer rather than from a policy transfer.

5.2 Consolidation does not prevent transfer

During our research, we noticed a property of the consolidation step that appeared to be verified on each experiment: although increasing the number of games used during the passive phase doesn't necessarily lead to better performance, it also doesn't degrade it. For instance, when transferring from VideoPinball to Breakout, training the AMN on several additional games doesn't have any impact on the learning curve. We made the hypothesis that the consolidation process induces the same improvement as the game which would have had the strongest impact if transferred alone (i.e. with a passive phase trained only on it). This hypothesis could provide an easy method to quickly find which games can benefit from transfer from other games by using a binary search.

Finally, we explored whether there can be interference between games during the consolidation of an AMN. Figure 7 shows that SpaceInvaders has a positive impact on Assault and VideoPinball on Breakout, however consolidating an AMN on both of the source games does not make it a good initialization point for either of the target games. These experiments show that although consolidation



(a) (b) (c)

Figure 8: Performance of experts initialized by copying the weights of the first n layers of another expert trained on VideoPinball. The baseline corresponds to an expert initialized randomly.

does not prevent transfer in the majority of the cases, there still exists some situations in which it can have a direct negative impact on the experts' performance.

6 Transfer without consolidation

In order to measure the real impact of the consolidation phase, we also evaluated direct transfer without consolidation. In this setting, an expert network is trained on a source task and then used as an initialization point for a new expert trained on a different target task. To better grasp the underlying mechanisms, we studied direct transfer from VideoPinball to other Atari games. As in the previous experiments, we dealt with different action sizes by masking or extending the output layer.

In order to better understand the importance of each individual layer in the transfer process, we compared the performance obtained when only certain layers were transferred (Figure 8). All our networks use the same architecture of three convolutional layers followed by two fully connected layers. Interestingly, the results are quite varied depending on the target task: first of all, on Carnival, transferring one, two or three layers is equivalent to not transferring anything, whereas transferring the first four or five layers surprisingly deteriorates the very early training. This result indicates that the agent is actually hindered by the previous policy, which could hint that transferring knowledge between these two games is difficult. On the contrary, transferring any number of convolutional layers to Breakout doesn't have any effect on the performance, while transferring the features (first four layers) or the policy (all layers) prevents the agent from plateauing and greatly improves the asymptotic performance. One interpretation is that in this situation, the network needs the complete extracted features to avoid local minima, but the visual features are not enough alone and the agent is not able to retrieve enough information from them. Finally, for SpaceInvaders, only the two first convolutional layers impact positively the agent's performance while the other layers have negative effects. This leads us to conjecture that only low-level visual features like edge detection are reusable in this case, and knowledge that is too specific only reduces the network plasticity. These experiments show that the concept of knowledge transfer can greatly differ depending on the tasks to solve, and these differences can explain the variability we observed in our results.

7 Conclusion

In this work, instead of an involved theoretical analysis, we propose an empirical phenomenological discussion of the practical aspects of neural consolidation for knowledge transfer in neural networks, which we believe brings a new light on the matter for the community, as well as open questions and perspectives. We found that it is difficult to set up the right conditions to observe a consistently positive impact on the performance, especially since the mechanisms behind transfer are still not clear. In the end, it would seem that meeting the necessary conditions to transfer and improve training is very dependent on the environment chosen and on several hyperparameters. Still, a potential method to deepen our understanding of the core mechanisms behind transfer would be to further analyse the few cases where the effect is really significant, and consolidation can be an interesting tool in this regard as it allows for comparison of different ways of transferring knowledge.

References

Shawn Beaulieu, Lapo Frati, Thomas Miconi, Joel Lehman, Kenneth O Stanley, Jeff Clune, and Nick Cheney. Learning to Continually Learn. In *24th ECAI European Conference on Artificial Intelligence*, page 10, 2020. URL `https://ecai2020.eu/papers/939_paper.pdf`.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research*, 47:253–279, June 2013. ISSN 1076-9757. doi: 10.1613/jair.3912. URL `https://www.jair.org/index.php/jair/article/view/10819`.

Glen Berseth, Cheng Xie, Paul Cernek, and Michiel Van de Panne. Progressive reinforcement learning with distillation for multi-skilled motion control. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=B13njo1R-`.

Cristian Bucilua, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, page 535–541, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933395. doi: 10.1145/1150402.1150464. URL `https://doi.org/10.1145/1150402.1150464`.

Pablo Samuel Castro, Subhodeep Moitra, Carles Gelada, Saurabh Kumar, and Marc G. Bellemare. Dopamine: A research framework for deep reinforcement learning, 2018.

Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A. Rusu, Alexander Pritzel, and Daan Wierstra. PathNet: Evolution Channels Gradient Descent in Super Neural Networks. *arXiv:1701.08734 [cs]*, January 2017. URL `http://arxiv.org/abs/1701.08734`. arXiv: 1701.08734.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, July 2017. URL `http://proceedings.mlr.press/v70/finn17a.html`. ISSN: 2640-3498.

David Ha and Jürgen Schmidhuber. World Models. *arXiv:1803.10122 [cs, stat]*, March 2018. doi: 10.5281/zenodo.1207631. URL `http://arxiv.org/abs/1803.10122`. arXiv: 1803.10122.

Xu He, Jakub Sygnowski, Alexandre Galashov, Andrei A. Rusu, Yee Whye Teh, and Razvan Pascanu. Task Agnostic Continual Learning via Meta Learning. In *ICML 2020 Workshop LifelongML*, June 2020. URL `https://openreview.net/forum?id=AeIzVxdJgeb`.

Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI*, pages 3215–3222, 2018. URL `https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17204`.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. *arXiv:1503.02531 [cs, stat]*, March 2015. URL `http://arxiv.org/abs/1503.02531`. arXiv: 1503.02531.

Heechul Jung, Jeongwoo Ju, Minju Jung, and Junmo Kim. Less-forgetting Learning in Deep Neural Networks. *arXiv:1607.00122 [cs]*, July 2016. URL `http://arxiv.org/abs/1607.00122`. arXiv: 1607.00122.

Łukasz Kaiser, Mohammad Babaeizadeh, Piotr Miłos, Błażej Osiński, Roy H. Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model Based Reinforcement Learning for Atari. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=S1xCPJHtDB`.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, March 2017. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1611835114. URL `https://www.pnas.org/content/114/13/3521`. Publisher: National Academy of Sciences Section: Biological Sciences.

Zhizhong Li and Derek Hoiem. Learning without Forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, December 2018. ISSN 1939-3539. doi: 10.1109/TPAMI.2017.2773081. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 6470–6479, Red Hook, NY, USA, December 2017. Curran Associates Inc. ISBN 978-1-5108-6096-4.

Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents. *Journal of Artificial Intelligence Research*, 61:523–562, March 2018. ISSN 1076-9757. doi: 10.1613/jair.5699. URL `https://www.jair.org/index.php/jair/article/view/11182`.

James L. McClelland, Bruce L. McNaughton, and Randall C. O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*, 102(3):419–457, 1995. ISSN 1939-1471(Electronic),0033-295X(Print). doi: 10.1037/0033-295X.102.3.419. Place: US Publisher: American Psychological Association.

Alex Nichol, Joshua Achiam, and John Schulman. On First-Order Meta-Learning Algorithms. *arXiv:1803.02999 [cs]*, October 2018. URL `http://arxiv.org/abs/1803.02999`. arXiv: 1803.02999.

German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, May 2019. ISSN 0893-6080. doi: 10.1016/j.neunet.2019.01.012. URL `https://www.sciencedirect.com/science/article/pii/S0893608019300231`.

Emilio Parisotto, Lei Jimmy Ba, and Ruslan Salakhutdinov. Actor-Mimic: Deep Multitask and Transfer Reinforcement Learning. In *ICLR*, January 2016. URL `https://openreview.net/forum?id=4zXscyOrI_m`.

Lorien Y. Pratt, Jack Mostow, and Candace A. Kamm. Direct transfer of learned information among neural networks. In *Proceedings of the ninth National conference on Artificial intelligence - Volume 2*, AAAI'91, pages 584–589, Anaheim, California, July 1991. AAAI Press. ISBN 978-0-262-51059-2.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental Classifier and Representation Learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5533–5542, July 2017. doi: 10.1109/CVPR.2017.587. ISSN: 1063-6919.

Andrei A. Rusu, Sergio Gomez Colmenarejo, Çaglar Gülçehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy Distillation. In *ICLR*, January 2016a. URL `https://openreview.net/forum?id=9nHQRKrWAMt`.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive Neural Networks. *arXiv:1606.04671 [cs]*, September 2016b. URL `https://openreview.net/forum?id=Hy1e7Z05`. arXiv: 1606.04671.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, pages 4528–4537, 2018.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 2994–3003, Red Hook, NY, USA, December 2017. Curran Associates Inc. ISBN 978-1-5108-6096-4.

Matthew E. Taylor and Peter Stone. Transfer Learning for Reinforcement Learning Domains: A Survey. *Journal of Machine Learning Research*, 10(56):1633–1685, 2009. URL `http://jmlr.org/papers/v10/taylor09a.html`.

Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 4496–4506, 2017.

Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong Learning with Dynamically Expandable Networks. In *ICLR*, February 2018. URL `https://openreview.net/forum?id=Sk7KsfW0-`.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 5824–5836. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/3fe78a8acf5fda99de95303940a2420c-Paper.pdf`.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual Learning Through Synaptic Intelligence. In *International Conference on Machine Learning*, pages 3987–3995. PMLR, July 2017. URL `http://proceedings.mlr.press/v70/zenke17a.html`. ISSN: 2640-3498.

Appendices

A Experimental Details

We reimplemented the AMN algorithm and the different modifications we made by building on the Dopamine framework [Castro et al., 2018]. Notably, we reuse the same network architecture comprised of three convolutional layers followed by two fully connected layers. More specifically, the architecture we use is 8x8x4x32-4 $\rightarrow$ 4x4x32x64-2 $\rightarrow$ 3x3x64x64-1 $\rightarrow$ 512 fully-connected units $\rightarrow$ outputs, where we note convolutional layers as WxHxCxN-S, with W and H the width and height of the convolution kernel, C the number of channels, N the number of filter maps and S the convolution stride. All layers except the action outputs are followed with a rectifier non-linearity.

We use the same hyperparameters as in the original paper [Parisotto et al., 2016], notably the scaling parameters in the feature loss and the masking process to adapt the size of the outputs to every Atari games. During the passive phase, we keep a Prioritized Replay Buffer [Schaul et al., 2015] per game to train the AMN, which are filled by interacting with the environments following the AMN actions. To ensure exploration, we use an ϵ-greedy policy both for the experts and the AMNs, with ϵ starting from 1 and annealing progressively to 0.1.

The code can be found on an anonymous github at `https://anonymous.4open.science/r/ConsolidationForTransferInRL`.

B Complete results and graphs

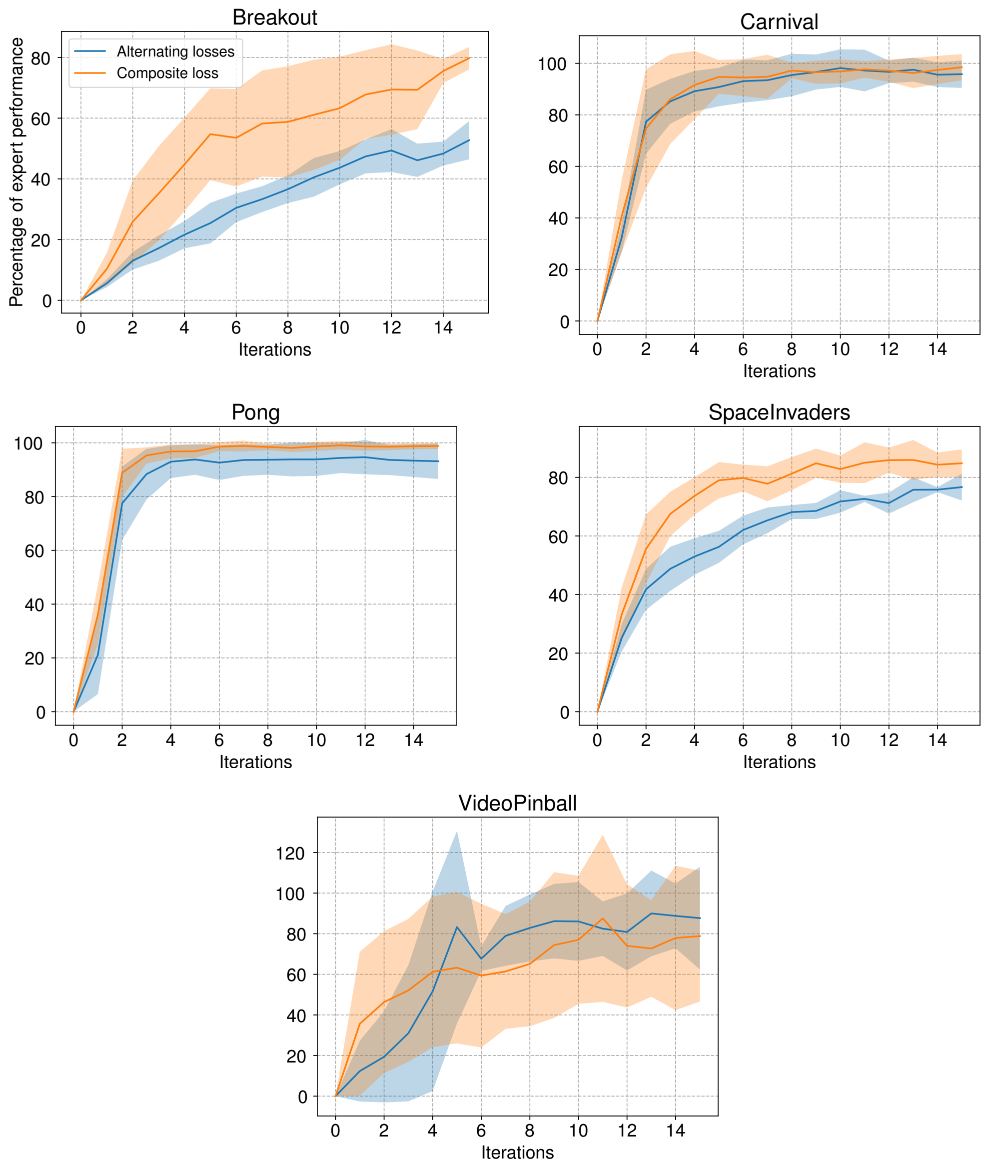


Figure 9: Performance of an AMN network consolidated on 5 experts and measured in percentage of the experts final score when minimizing the composite loss against when minimizing each individual loss sequentially. Each experiment is repeated 3 times, the shaded area corresponds to the standard deviation around the mean.

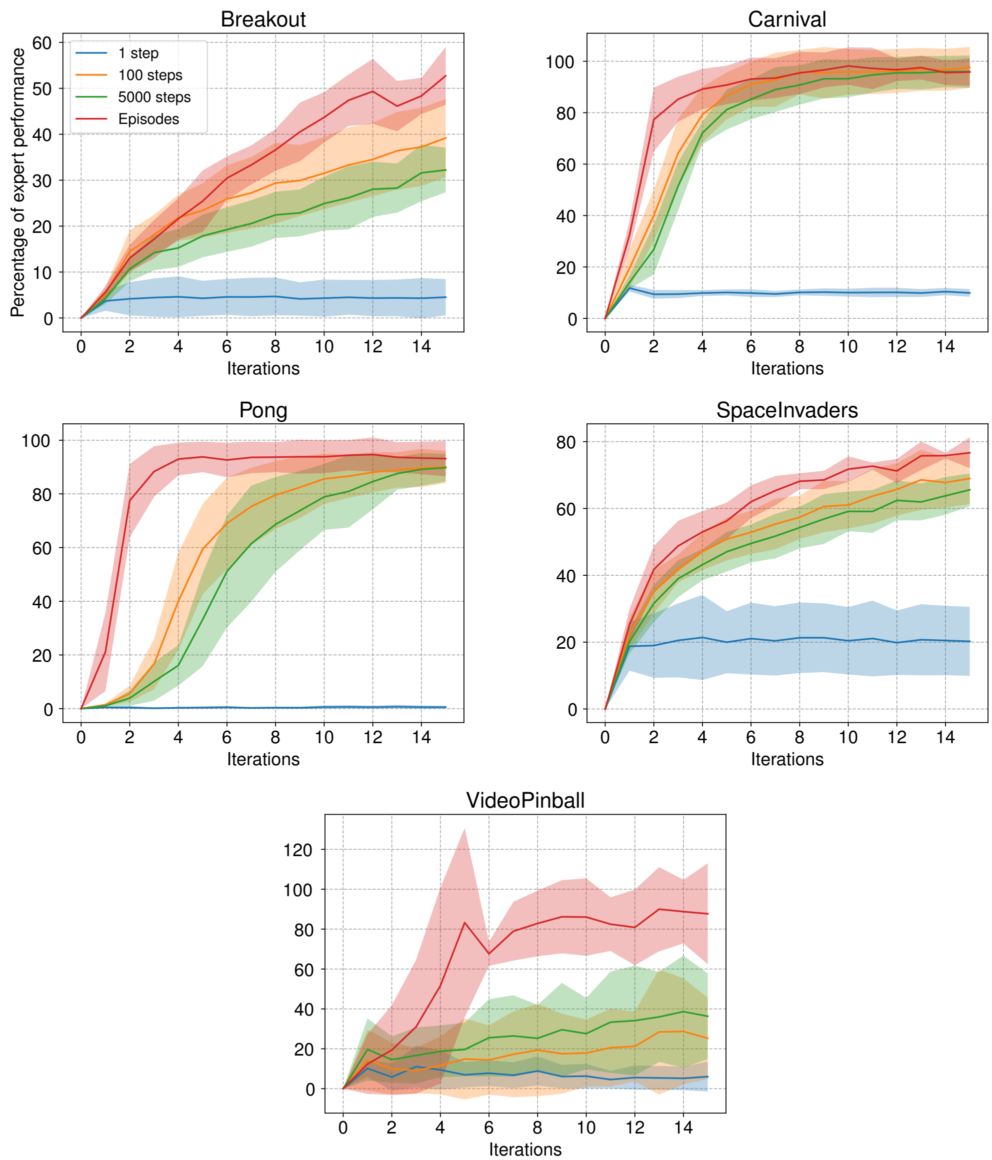


Figure 10: Performance of the AMN network measured in percentage of the experts' final score. The gradient descent process switches between tasks every 1, 100, 5000 steps or every episode.

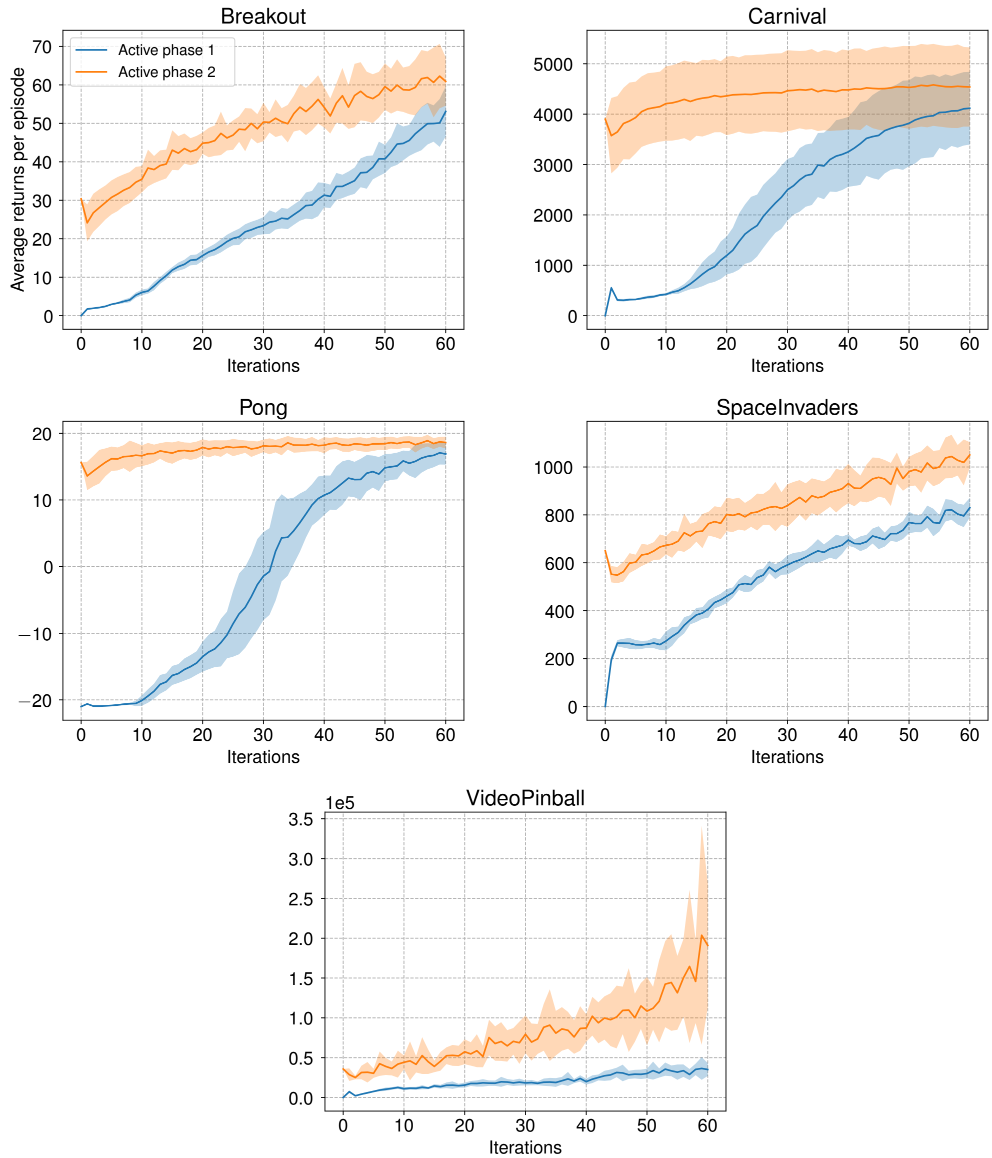


Figure 11: Average return per episode for an expert initialized randomly (active phase 1) or from an AMN after a passive phase (active phase 2)

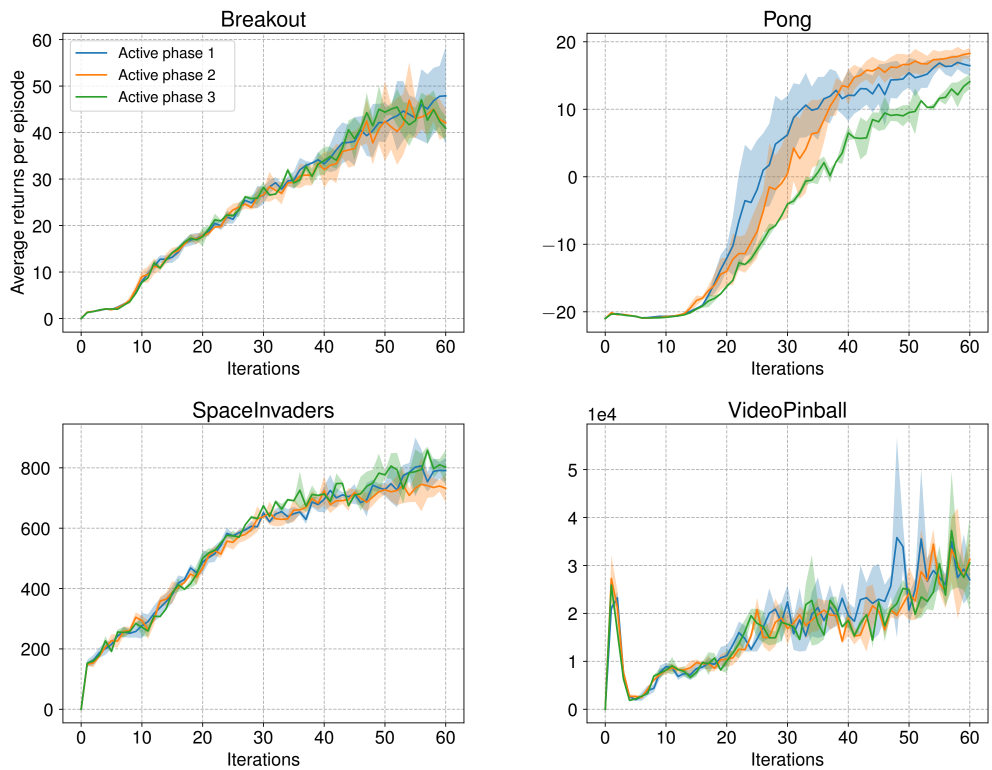


Figure 12: Average return per episode for experts trained during different active phases.

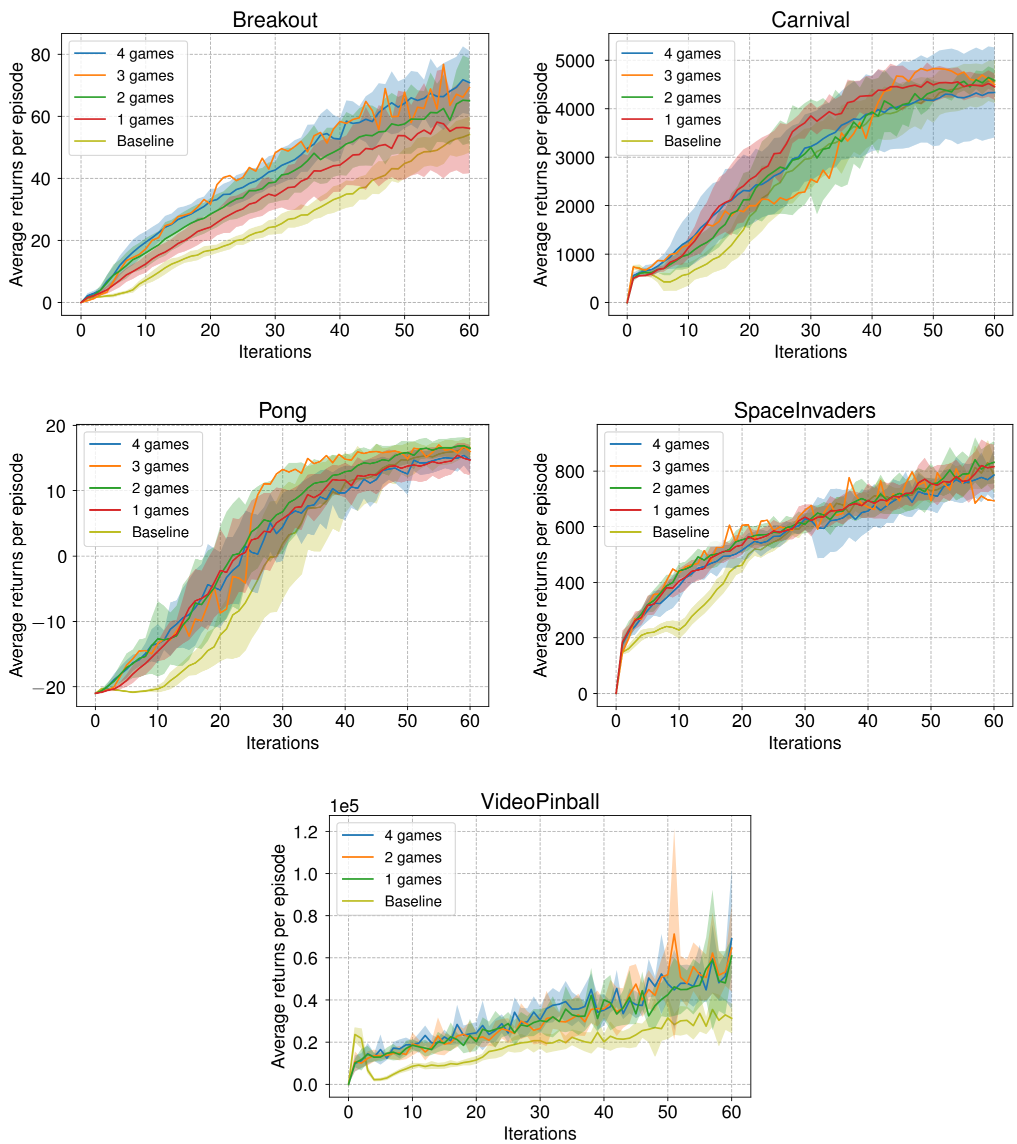


Figure 13: Performance of experts initialized from an AMN trained on varying numbers of games during the passive phase. The baseline curve corresponds to an expert initialized randomly.

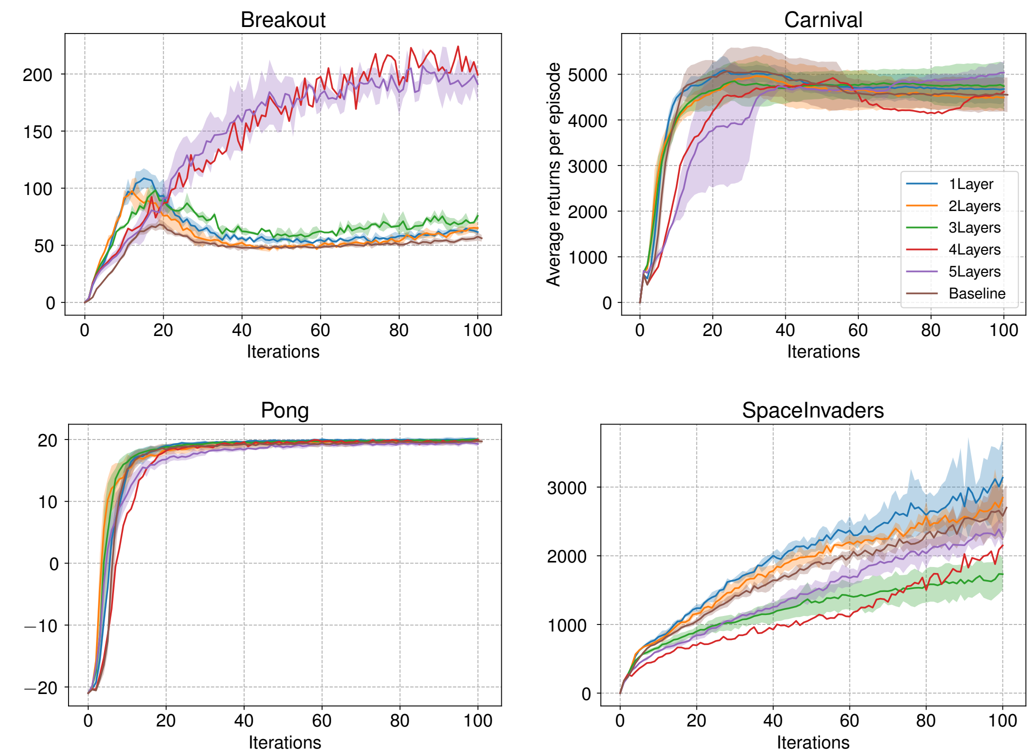


Figure 14: Performance of experts initialized by copying the weights of the first n layers of another expert trained on VideoPinball. The baseline corresponds to an expert initialized randomly.