Transferring Deep Reinforcement Learning with Adversarial Objective and Augmentation

I-Chao Shen , Shu-Hsuan Hsu , Bing-Yu Chen
National Taiwan University
{jdily, ssarcandy}@cmlab.csie.ntu.edu.tw, robin@ntu.edu.tw

Abstract

In the past few years, deep reinforcement learning has been proven to solve problems which have complex states like video games or board games. The next step of intelligent agents would be to generalize between tasks, and using prior experience to pick up new skills more quickly. However, most reinforcement learning algorithms, for now, are often suffering from catastrophic forgetting even when facing a very similar target task. Our approach enables the agents to generalize knowledge from a single source task, and boost the learning progress with a semi-supervised learning method when facing a new task. We evaluate this approach on Atari games, which is a popular reinforcement learning benchmark, and show that it outperforms common baselines based on pre-training and fine-tuning.

1 Introduction

Deep Reinforcement Learning (DRL), the combination of reinforcement learning methods and deep neural network function approximators, has recently shown considerable success in challenging tasks that have very complex states and many available actions, such as arcade video games [Mnih *et al.*, 2015], robotic manipulation [Levine *et al.*, 2016], and even the challenging classic games - Go [Silver *et al.*, 2016]. These methods can learn features that are often better than handcraft ones, which require more domain knowledge. For example, Deep Q-Network (DQN) [Mnih *et al.*, 2015] is one of the most famous DRL methods, and has achieved super human performance on the Arcade Learning Environment (ALE) [Bellemare *et al.*, 2013], which is a benchmark of Atari 2600 arcade games.

Although the DRL algorithms can usually learn how to take the best action based on the state of the environment, but it can only learn a single environment at a time, despite the similarities between those environments. For example, the tennis-like game of pong and the squash-like game of breakout are similar in that each game consists of trying to hit a moving ball with a rectangular paddle, but an agent that is good at pong cannot handle breakout well, and vice versa. Another issue of DRL is that training DRL agents can be very time-consuming, so many researchers are studying on the methods that can speed up the training time [Mnih *et al.*, 2016; van Hasselt *et al.*, 2016].

Some methods speed up the learning on new tasks by performing cross environment transfer [Rusu *et al.*, 2016; Parisotto *et al.*, 2015], but they all need to pre-train an agent on multiple source environments to generalize the knowledge, which is very time-consuming. In this work, we are trying to leverage the prior knowledge learned by an agent in a single-source environment to speed up the agent to handle another new environment. Using the prior knowledge in only one source environment can minimize the training time in another new environment, and can also solve some issues of reinforcement learning, including unable to handle similar tasks and long training time problems.

Inspired by previous proposed domaion adaptation works [Tzeng *et al.*, 2017; Sun and Saenko, 2016; Ganin and Lempitsky, 2015; Ganin *et al.*, 2016; Sun *et al.*, 2016; Xiao *et al.*, 2016; Ben-David *et al.*, 2010], we propose a semi-supervised transfer learning method that uses the concept of the adversarial objective. More specifically, a mapping function is learned from the target observations to the source feature space by fooling a domain discriminator that tries to distinguish the encoded target observation from the source examples. We also found it is helpful for transferring knowledge by altering the visual content with the proper setting when training on the source task, that is, adding proper augmentation when training will be helpful on transferring knowledge to another task. Our approach can be integrated into any DRL algorithm, and in this paper, we show our results by combining it with DQN [Mnih *et al.*, 2015] algorithm and performing on Atari 2600 benchmark.

Our contributions are two-folds: First, we propose a method that can leverage the knowledge learned from a single source task to speed up the training on another new target environment. Second, we found that performing proper augmentation on the environment to train a source agent and use it as a target task initialization often can help. With these proposed methods, the overall learning on the target task can be accelerated comparing with the baselines that we have considered. .

2 Approach

In this section, we present a transfer learning method that can speed up the learning progress when facing new target task.

Furthermore, we found a novel data augmentation method that can help single task agents to avoid over-fitting and thus learn more generalized policy.

2.1 Transfer with Adversarial Objective (AdvTransfer)

We present a framework for unsupervised domain adaption between deep reinforcement learning tasks. Assume a source environment Env_s and a target environment Env_t, there are domain shifts between Env_s and Env_t. We have access to the reward and next state after acted on Env_s.

The overview is shown in Figure 1, we first pre-trained the source task agent on Env_S, training a feature encoder F_S and Q-value predictor V_S that can return a vector of Q values for all possible actions by given features of observation, thus can select the best action a. And our goal is to learn a target feature encoder F_T and Q-value predictor V_T that can handle Env_T as fast as possible.

We integrate generative adversarial network [Goodfellow *et al.*, 2014] concept into transfer progress, as shown in Figure 1 (b). The target feature encoder F_T plays the role of the generator, and a domain classifier D that can predict the domain label (source or target domain) by seeing the encoded features output by F_S and F_T. We then perform adversarial adaption by training the F_T in order to to prevent the domain classifier from predicting the correct domain label from the encoder features. The target feature encoder (Generator) and domain classifier (Discriminator) are playing counterparts. At the end of the training, the target feature encoder F_T will learn the feature representations that can be generalized across both source and target domain.

In the adversarial objective approach, the main goal is to regularize the learning of the source and target mappings, we can minimize the distance between $F_S(S_s)$ and $F_T(S_t)$ distributions, where S_s and S_t is the states of Env_S and Env_T. If the distribution between $F_S(S_s)$ and $F_T(S_t)$ are similar, then we can directly apply source task Q-value predictor V_s to the F_t, skipping the need to learn a V_t.

We first describe the domain classifier, D, which classifies whether encoded features are drawn from the source or the target domain. Thus D is optimized according to a standard cross entropy loss, $L_D(S_s, S_t, F_s, F_t)$ where the labels indicate the origin domain, defined below:

$$L_D(S_s, S_t, F_s, F_t) = -\log(D(F_s(S_s))) - \log(1 - D(F_t(S_t))) \tag{1}$$

And for the generator, we train with the standard loss function with inverted labels [Goodfellow *et al.*, 2014]; then the loss can be described as:

$$L_G(S_t, D) = -\log(D(F_t(S_t))) \tag{2}$$

Then, the source and target mappings are optimized according to an adversarial objective; they are optimized to confuse D to unable to predict reliable domain label.

2.2 Augmentation

Most deep reinforcement learning algorithms can achieve great performance in a single environment, but agents often cannot handle a new slightly altered environment. A DQN agent that pre-trained on original Pong performs badly on two of its variations (-15.9 on Pong with Gaussian noise and -20.9 on Pong with inverted color).

Rusu *et al.* [Rusu *et al.*, 2016] analyzed the Pong to noisy Pong case, and found that the high-level filter on the clean task is not sufficiently tolerant to the added noise. Thus some new low-level vision has to be relearned to adapt to the new task environment. We found that by adding some augmentations to the source task during training on the source task would help avoid over-fitting and thus learn more critical features of the task. We added a data augmentation layer before feed the input data into replay memory; the data augmentation layer will randomly transform the input. For example, Eq. (3) demonstrates a data augmentation layer that will invert the color of the environment state (screen) with the probability of 30%, and remain unchanged otherwise.

$$S = \begin{cases} 1 - S & 30\% \\ S & \text{otherwise} \end{cases} \tag{3}$$

Detail Evaluation

We now further discuss the detailed evaluation of the augmentation setting, including why and how to find the good augmentation method for training.

Inspired by standard computer vision training pipeline, we added several varieties to the training set. By integrating the augmentation method into the reinforcement learning scenario, some constraints must be added:

1. The augmentation must not break the consistency of the visual content.
2. The augmentation should not be too hard for the agent to learn.
3. The augmentation should as different from original visual content as possible to avoid over-fitting.

The first constraint came from reinforcement learning characteristic that the position on visual content is crucial, using augmentation method like rotate or flip would break the consistency of visual content. Second, using the augmentation that altered the visual content heavily would cause the agent to fail to learn the task. And for the third point, using visual altered augmentation prevent the agent from over-fitting and increase the ability of the agent to tolerate noise. It is essential to find a balance between the second and third points, i.e., the augmentation should be as different from the original visual content as possible but not be too hard for the agent to learn.

We conduct an experiment on a series of different augmentation setting, evaluate both the distance between "with augmentation" and "without augmentation" and the final performance on the task. We perform three sets of different augmentation method, including *Gauss*, *Grid* and *Inverted*. Each method further test three different variation levels, the sample frames of different levels of method *Gauss* and *Grid* are shown in Figure 1 in supplemental material , and the *Invert* method levels are defined as the invert frame frequencies.

We use PSNR to evaluate the appearance distance between game playing frames, the smaller PSNR means longer dis-

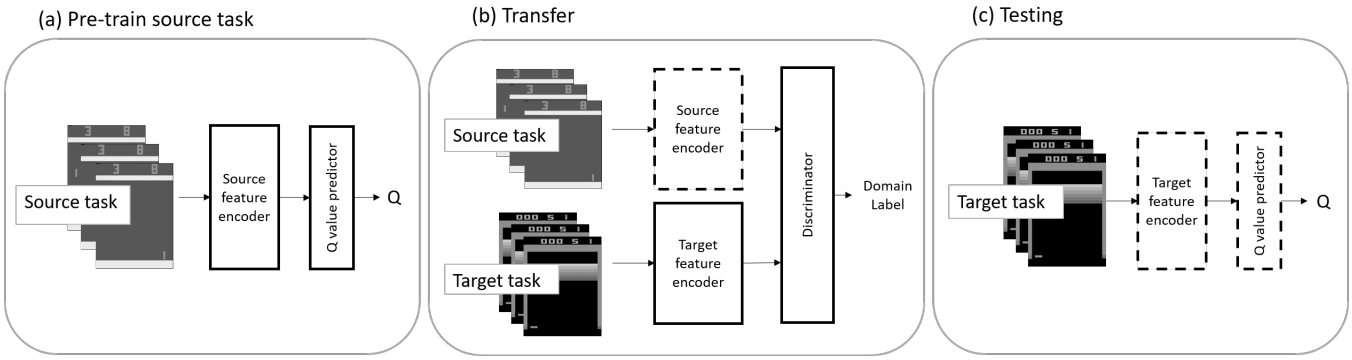

Figure 1: An overview of our transfer process. (a) We first well-trained an agent on source task environments, then (b) we train a domain classifier and target feature encoder that have an adversarial objective to learn a map that maps target feature encoder to source feature encoder. (c) At the testing time, we can use source task Q value predictor directly because the target feature encoder's output features are similar to source feature encoder. The dashed line means fixed parameters.

tance. We made some observations from this plot. First, the stronger level of augmentation would cause the agent gradually unable to handle the task. We need to choose the one that meets the constraints described above that the augmentation should not be too hard for the agent to learn but as different from normal as possible. Although all three augmentations (*Gauss*, *Grid* and *Inverted*) can have good performance in lower level setting, i.e., **Gauss** $\sigma = 20$, **Grid 30x30** and **Inverted 30%**, but the **Inverted 30%** is the only one that has smallest PSNR value and remains good performance so that it would be the better augmentation setting.

3 Experiments

In the following experiments, we evaluate the transfer effectiveness of our method using the Arcade Learning Environment (ALE) [Bellemare *et al.*, 2013]. First, we conducted an experiment to evaluate how our method improves the transfer effectiveness across synthetic variations of Pong. Second, we experiment on a more challenging setting, i.e. transfer between different Atari games. In each experiment, we compare the agent performances of following four methods:

- *random baseline*: train DQN directly target task with random initialized weights.
- *naïve baseline*: train DQN on target task with the pre-trained weights on source task. well-trained source task model parameters to target task network
- *our method w/o augmentation*: DQN with adversarial objective trnasfer (described in Section 2.1).
- *our full method*: DQN with adversarial objective transfer and frame augmentations.

3.1 Pong Variants

The first evaluation domain is a set of synthetic variants of the Atari game "Pong". We created synthetic variants by altering visual appearances of the original "Pong". The variants of Pong are **Noisy** (Gaussian noise is added to the inputs), **Grid** (fixed grid lines are added on input), **Invert** (input color is inverted), and **Scale** (input is scaled by 75% and with black padding). The purpose of this experiment is to test whether the agent can learn the core gameplay across its visual variants.

		Primes			
		gauss	**grid**	**scale**	**invert**
Naïve transfer	Pong	2	1.3	0.96	0.12
Our w/o Augmentation	Pong	2	1.79	1.75	1.47
Our	Pong	2	2	2	2

Table 1: Transfer score matrix. The higher score means the better-transfer performance. Colours indicate transfer scores (clipped at 2).

Figure 2 shows the transfer progress of each variant. Overall, our method (both with and without augmentation versions) achieve better learning progress on target task compare to baseline methods. Our method saves at least 1 millions frames of training time to reach the convergence state for **Pong-grid**, **Pong-scale** and **Pong-invert**.

Moreover, we measure the transfer performance by the transfer score [Rusu *et al.*, 2016]. The transfer score is defined as the relative performance of a method compared with the *random baseline* method. The Higher transfer score means more effective transfer. A transfer score greater than one means the performance is relatively better than the *random baseline* method, i.e. positive transfer. On the contrary, a transfer score smaller than one means the performance worse than the *random baseline* method, i.e. negative transfer.

As shown in Table 1, we can observe that our method get better transfer scores across all experiments. Interestingly, the *naive baseline* (initialized with pre-trained weights on "Pong") obtains worst transfer score. It indicates that there are negative transfer effects, especially for **Pong-invert** case.

Our method boosts the learning speed for both with and without augmentation versions of our method. As shown in Figure 2, we can observe that our method with augmentation learns significant faster compared to the version without augmentation. This shows that our method with augmentation can significantly help the agent learns the core gameplay across a single game's visual variants.

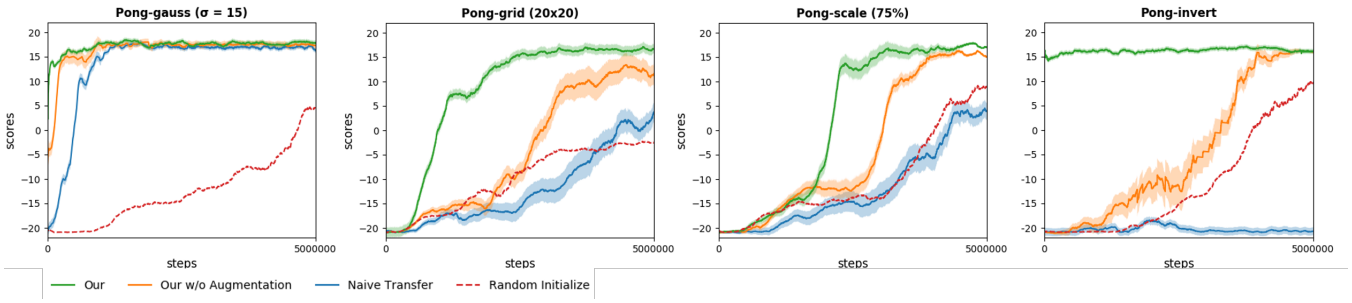

Figure 2: Transfer progress of Pong variants. Pong variants include noisy, inverted color and scaled transforms. The results are averaged over 3 runs and the shadow represent standard deviation.

We further look closely at specified transfer pair cases. For **Pong-gauss** case, the difference between source and target task is smallest. We can observe that even the *naïve baseline* method provides a very good transfer effect. Although the *naïve baseline* method reaches max transfer scores (i.e. 2.0), our method still outperform it on coverage time as shown in Figure 2. For **Pong-invert** case, it shows that *naïve baseline* method fails to learn on target task, which means that the knowledge learned from source task are hindering the agent to learn target task. On the other hands, our method without augmentation minimizes the negative effect because the generator will try to produce features that are similar with source task's features. In terms of our full method, it achieves great performance at starting time because the source task is trained with 30% inverted frame, in other words, the source agent has learned **Pong-invert** at training time, thus it can handle this specified case well. In general speaking, our method obtains the best transfer result, followed by our method without augmentation, and naïve transfer method obtains the worst result.

3.2 Multi-level Transfer

Next, we designed a multi-level game scenario to test the effetiveness of our transfer method. It is common that for each game, the designer designed multi-levels with increasing difficulties. Instead of training the agents separately on each level, we use the proposed transfer learning method to increase the learning speed when facing harder levels.

In our scenario, we build a multi-level version of Breakout[1] with different difficulties. We designed four different levels with increasing difficulties by shrinking the paddle width, the different paddle widths are: 30px width (level-1), 20px width (level-2), 10px width (level-3) and 5px width (level-4). Figure 3 in the supplemental material shows four different levels of sample frames.

And we perform three transfer cases, including level-1 to level-2, level-1 to level-3, and level-1 to level-4. The training progress shown as Figure 3.

We can find that the worst performance is "Random initialize" (red dashed line) because the agent was trained from scratch, without any prior knowledge. And all other transfer methods have very good transfer effectiveness and can be reached stable performance within 1 million steps. For width 30 to width 20 (level-1 to level-2) case, the performance of our methods (both w/ and w/o augmentation) are better than the Naive method, and augmentation (orange line) helps the agent to learn faster at the beginning. For width 30 to width 10 (level-1 to level-3) case, our methods still have a noticeable boost of learning effectiveness than the naïve method. Finally, for 30 to width 5 (level-1 to level-4) case, there is barely any difference between the naïve method and our methods. This suggested that the difficulties of the final level (i.e. level 4) has significant different from all the previous levels. The transfer method's performance can then be used to identify the difficulty differences during the game designing process.

[1]Environment source code can be found at https://github.com/SSARCandy/breakout-env

4 Conclusion

In this works, we investigate the knowledge transfer for deep reinforcement learning. Unlike previous works [Rusu *et al.*, 2016; Parisotto *et al.*, 2015], that requires training multiple agents on multiple source tasks for generalizing and transfering to target task, we proposed a method that can accelerate the training progress on a new task with a single prior task. Furthermore, we found that with a simple data augmentation method, the agent can learn the target task faster. And we demonstrated our method outperforms baselines in both easy and challenging cases using Atari 2600 benchmark.

References

[Bellemare *et al.*, 2013] 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, 2013.

[Ben-David *et al.*, 2010] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

[Ganin and Lempitsky, 2015] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *Proc. ICML'15*, volume 37, pages 1180–1189, 2015.

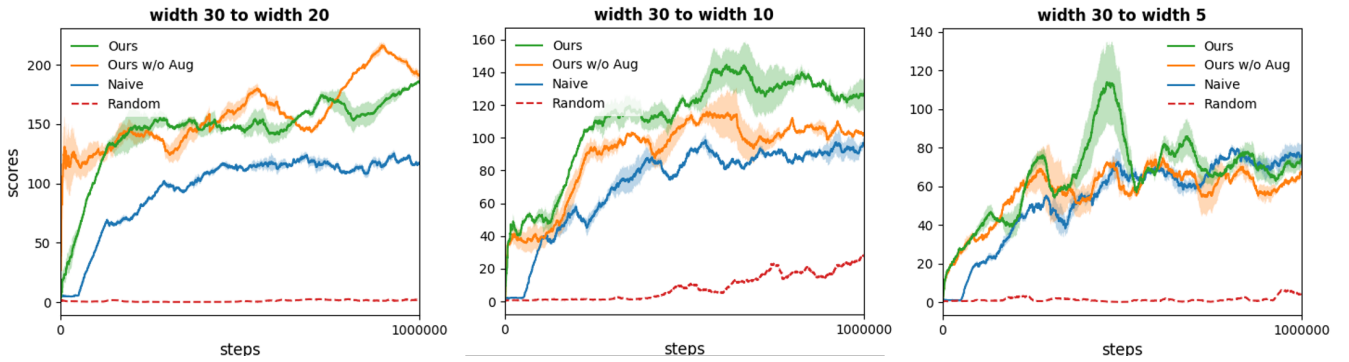

Figure 3: Transfer progress of cross-level scenario.

[Ganin *et al.*, 2016] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

[Goodfellow *et al.*, 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proc. NIPS'14*, volume 27, pages 2672–2680, 2014.

[Levine *et al.*, 2016] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[Mnih *et al.*, 2016] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proc. ICML'16*, volume 48, pages 1928–1937, 2016.

[Parisotto *et al.*, 2015] Emilio Parisotto, Lei Jimmy Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *CoRR*, abs/1511.06342, 2015.

[Rusu *et al.*, 2016] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *CoRR*, abs/1606.04671, 2016.

[Silver *et al.*, 2016] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[Sun and Saenko, 2016] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *ECCV'16 Workshops, Part III*, pages 443–450, 2016.

[Sun *et al.*, 2016] Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. In *Proc. AAAI'16*, pages 2058–2065, 2016.

[Tzeng *et al.*, 2017] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *IEEE CVPR'17*, pages 2962–2971, 2017.

[van Hasselt *et al.*, 2016] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proc. AAAI'16*, pages 2094–2100, 2016.

[Xiao *et al.*, 2016] Tong Xiao, Hongsheng Li, Wanli Ouyang, and Xiaogang Wang. Learning deep feature representations with domain guided dropout for person re-identification. In *Proc. CVPR'16*, pages 1249–1258, 2016.

Supplemental Material : Transferring Deep Reinforcement Learning with Adversarial Objective and Augmentation

I-Chao Shen , Shu-Hsuan Hsu , Bing-Yu Chen
National Taiwan University
{jdily, ssarcandy}@cmlab.csie.ntu.edu.tw, robin@ntu.edu.tw

1 Augmentation Game Content

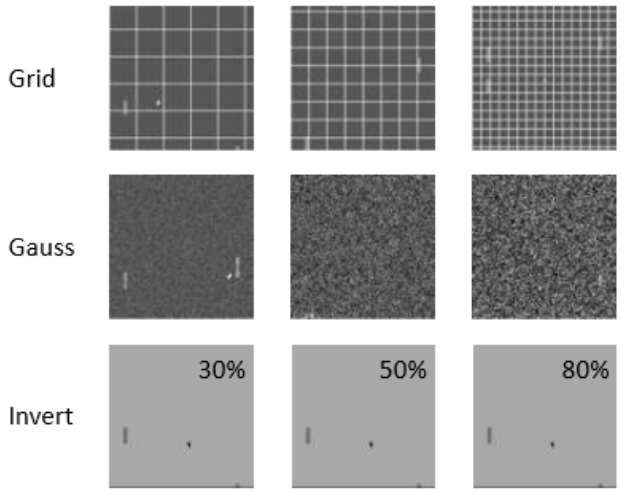

Figure 1: Different intensities of each augmentation on Pong. The first row shows the different size of grid noise, from left to right are 30, 20 and 10, the line color are all white; Second row is different level of Gaussian noise, from left to right are $\sigma = 20$, $\sigma = 50$ and $\sigma = 80$; Third row is different frequency of inverted frames, the percentage means how often will the frame be inverted color.

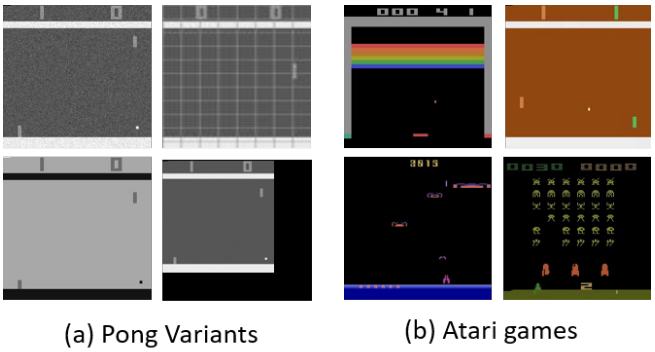

Figure 2: Example frames from different domains. (a) Pong variants: noisy, recoloured and scale transforms; (b) Atari games. From top right to bottom left: Breakout, Pong, Demon Attack, Space Invader. These games introduce a more challenging setting for transfer.

References

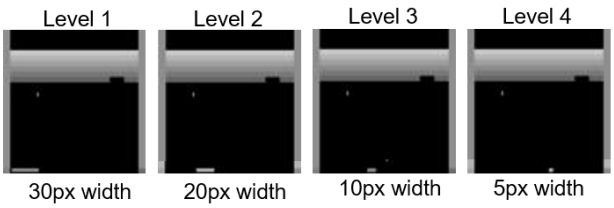

Figure 3: Sample frames of different level of Breakout.