# Selective Kernels Transfer in Deep Reinforcement Learning

**Jesús García Ramírez, Eduardo F. Morales and Hugo Jair Escalante**
**Instituto Nacional de Astrofísica Óptica y Electrónica**
Sta. María Tonantzintla, Puebla, México
{gr_jesus, emorales, hugojair}@inaoep.mx

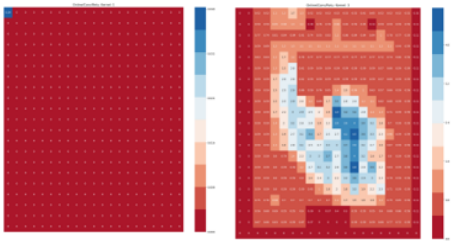
**Abstract**

Deep Reinforcement Learning (DRL) combines the benefits of Deep Learning and Reinforcement Learning. However, it still requires long training times and a large number of instances to reach an acceptable performances. Transfer Learning (TL) offers an alternative to reduce the training time of DRL agents, using less instances and possibly improving performance. In this work, we propose a transfer learning formulation for DRL across tasks. Relevant source tasks are selected considering the action spaces and the Wasserstein distances of an output in a hidden layer of a convolutional neural network. Rather than transferring the whole source model, we propose a method for selecting only relevant kernels based on their entropy values, which results in smaller models that can produce better performances. In our experiments we use Deep Q-Networks (DQN) with Atari games We evaluated the proposed method with dierent percentages of selected kernels and show that we can obtain similar performances than DQN in less interactions and with smaller models.

## 1 Introduction

Deep Reinforcement Learning (DRL) takes advantage of both, Deep Learning (DL) and Reinforcement Learning (RL), to learn from experience and raw to solve complex learning tasks. Impressive results have been obtained in several problems of outstanding complexity, as obtaining human-level control in Atari games (Mnih et al. 2015; Bellemare et al. 2013; Hessel et al. 2018), beating human experts in challenging games like Go (Silver et al. 2016; 2018), and even Starcraft, a very complex strategical game (Zambaldi et al. 2018). Nevertheless, DRL methods still require of long training times even with specialized hardware, as well as large numbers of instances in order to reach satisfactory performances.

Transfer Learning (TL) offers an alternative to reduce training times and the number of necessary instances to achieve acceptable performance. TL reuses previously obtained knowledge from one or more *source tasks* to learn a new *target task*. Several authors have used TL in DRL. Some of them use a distillation approach based on a teacher-student scheme to obtain a model with less parameters than the source one, but they are trying to learn the same task, while others try to obtain a generic model for different tasks (a.k.a. policy distillation) (Rusu et al. 2015; Schmitt et al. 2018; Parisotto, Ba, and Salakhutdinov 2016; Yin and Pan 2017; Rusu et al. 2016). Other approaches use fine-tuning, in one (de la Cruz et al. 2016) (transfer knowledge directly to the target task) or in two steps (find a intermediate representation with a deep model, then train the agent with DRL algorithm) (Mittel, Munukutla, and Yadav 2018; Carr, Chli, and Vogiatzis 2018).



(a) Uniform output (b) Output with diverse values

Figure 1: Examples of convolution outputs of two different kernels

We propose a new distance measure to identify, among a set of candidate source tasks, one that could be more useful to transfer for a new target task, while trying to avoid negative transfer. The proposed measure is based on a comparison between action spaces of source and target tasks, together with the Wasserstein distance between distributions of attributes extracted from the output of convolutional layers. The idea is to compare visual attributes extracted by a pre-trained model, that could be useful for the target task. Contrary to standard approaches that transfer the whole model (e.g., in a straightforward fine tuning setting), we perform an additional filtering process where we select only the most relevant kernels from the source DQN based on their entropy values. We hypothesize that kernels that produce outputs with the most diverse values are better to transfer than those that produce uniform/uninformative values, see

e.g., Figure 1. The proposed method was tested on several games of the Atari Learning Environment (ALE) (Bellemare et al. 2013) and we transfer different percentages of kernels in each convolutional layer.

The main advantages of the proposed approach are: the kernel selection avoids to transfer useless elements to the target task, this accelerate the training of the target model because the number of parameters is reduced. The prediction of one model that improves target model training is a relevant problem approached in this work. Our experimental results show that the proposed approach can significantly reduce the training time when compare with the same algorithm without transfer learning.

This paper is organized as follows. Section 2 presents some background information on DRL and TL. Related work is analysed in Section 3. The proposed measure to predict the most useful task and the kernel selection method are presented in Sections 4 and 5, respectively. Experimental results are described in Section 6 and conclusions and future research work are given in Section 7.

## 2 Background

In this section we review the relevant topics to this paper. First, we present a description of DRL and then we describe a TL framework.

### Deep Reinforcement Learning

A Reinforcement Learning problem can be modeled as a Markov Decision Process (MDP) described by a 5-tuple $(S, A, P, R, \gamma)$: where $S$ is a finite set of states; $A$, a finite set of actions; $P(s'|s, a)$ a probability state transition function; $R$, a reward function, and a discount factor $\gamma$ (Sutton, Barto, and others 2018).

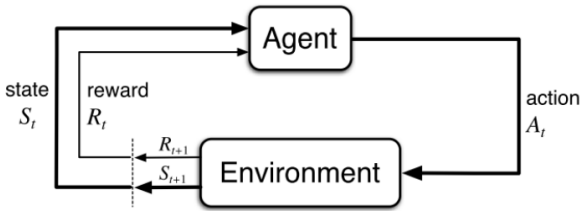



Figure 2: Interaction between the agent and the environment (Sutton, Barto, and others 2018)

In RL an agent interacts with the environment, as shown in Figure 2, to maximize an expected accumulated reward function. Deep Reinforcement Learning (DRL) algorithms are used when we want to approximate a policy and/or value functions using deep networks due to the size of action-state space (e.g., Atari games). After an interaction between the agent and the environment, the weights of an ANN are updated through backpropagation, taking into account the reward in a loss function (Mnih et al. 2015). For instance, some methods approximate the q-value (Mnih et al. 2015), others approximate both the state value and the q-value functions (Mnih et al. 2016) through gradient updates.

Most of the proposed DRL approaches are tested on the Atari Learning Environment (ALE) (Bellemare et al. 2013). This framework takes into account the states as the frames of a game, the actions are the combinations of the joystick movements and the button, and the obtained reward is simplified into three values [-1,0,1] (-1 when there is a negative change in the score, 0 for no change, and 1 for a positive change in the score).

The first approach that showed robust results in Atari games was Deep Q-Networks (Mnih et al. 2015) (DQN). Two novel strategies were used to obtain human-level performance in some games: experience replay and using an on-line and target networks. In experience replay a stack of experiences is used to randomly select instances to train the network. The other strategy used an on-line fixed network to update the target network during the training which were interchanged after several iterations. The target network was used to generate new instances.

In recent years several variations to this algorithm have been proposed: dueling networks architecture (Wang et al. 2015), prioritized experience replay (Schaul et al. 2015), approximating the output with a discrete distribution instead one value (Bellemare, Dabney, and Munos 2017); a combination of some techniques as rainbow (Hessel et al. 2018), among others. Nevertheless, these techniques are still computational expensive even with specialized hardware and a large number of instances are required to reach an acceptable performance.

### Transfer Learning

Transfer Learning (TL) offers an alternative to reduce training times, by reusing previously obtained knowledge in source tasks to quickly learn a new target task. Ideally a TL method could achieve a jump-start, converge faster, and possibly obtain a better asymptotic performance (e.g., see (Taylor and Stone 2009) and Figure 3).

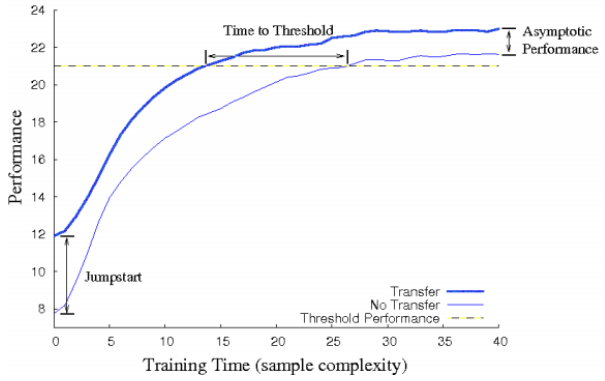



Figure 3: Performance of training with and without transfer (Taylor and Stone 2009).

TL uses a source task with sufficient label examples to obtain a conditional distribution $P(Y_s|X_s)$, using a marginal distribution $P(X_s)$, where $X_s = \{x_1, x_2, \ldots, x_n\}$ are the instances and $Y_s = y_1, \ldots, y_k$ are the labels for the source

task. The target task has few examples, insufficient to approximate the conditional distribution $P(X_t|Y_t)$. TL can approximate the conditional probability distribution of the target task using $P(X_s)$, $P(X_s|Y_s)$, and $P(X_t)$ (Pan and Yang 2009). One potential problem of TL is negative transfer, that appears when training with information from a source task has worse results than training the agent from scratch. To apply TL in the context of DRL, we need to define how to select potentially relevant source tasks and what information to transfer from the source to the target task.

## 3 RELATED WORK

Some research on TL for DRL has focused on using a pre-trained model to train a new one with less neurons for the same task aiming to obtain similar performance (policy distillation). The main advantage is that the target model has lower parameters and can be used in devices with few resources (i.e., a cellphone). However, in some cases the performance of the target model can be worst than the source model. For example, in (Rusu et al. 2015; Schmitt et al. 2018) two methods based on the teacher-student scheme are presented, where the teacher (source task model) is used to supervise the student training using regularizers. In (Schmitt et al. 2018), the authors improve the training of the agents 9.58 times (fewer steps), but they do not experiment with Atari games that is an important baseline for DRL agents.

In (Rusu et al. 2015) the authors proposed a multitask model, where they add a layer for each added task, so they obtain a model with more parameters, the main advantage is that a model for different task is obtained.

Parisotto et al. (Parisotto, Ba, and Salakhutdinov 2016) and Yin & Pan (Yin and Pan 2017) proposed a method that generalize a hidden layer for different games. Additionally, in (Parisotto, Ba, and Salakhutdinov 2016) a TL approach in two steps, one to obtain general features in a model and the other to fine-tune them to a specific task. Also, a progressive learning technique is used to obtain a model that works on several Atari games (Rusu et al. 2016), where for each new learned task a new set of layers are added to the CNN. A disadvantages of these works is that the obtained model has more parameters than the source one. They also select the source and target task in an intuitive way and do not take into account other tasks that could be more relevant according to the pre-trained model.

There are approaches where a model is trained for two tasks at the same time. In (Mittel, Munukutla, and Yadav 2018) a generative model is trained for two similar games (Breakout and Pong), then the generative model is used to train an actor-critic approach for each game. A similar approach is proposed by Carr et al. (Carr, Chli, and Vogiatzis 2018), where an autoencoder is trained for different tasks, then a DRL approach is applied to train each task using part of the autoencoder. In (de la Cruz et al. 2016) a work based on fine tuning is presented, where the authors transfer different number of layers and outperform the baseline model. The main limitation of these works is that they use a two-steps training, one to obtain a hidden representation for every task, and in a second step they train separately the task using a DRL algorithm. In our research, we use a one-step approach where we transfer relevant knowledge of a pre-trained model to the target task, avoiding the first step.

The kernel selection in this paper could be seen as a pruning method for selecting the most useful kernels according to the target task. Pruning methods select the best weights/kernels of a model and remove the less useful ones in order to reduce the number of parameters in a neural net. Consequently, the obtained model could be used in computers with limited resources. According to (He et al. 2018), pruning methods can be divided into: *data dependent filter pruning* (He et al. 2018; Luo and Wu 2017), that use the training or testing data to select the filters. While, *data independent filter pruning* (Li et al. 2016), that seearch for the most useful filters based on the magnitude of the obtained weights after training. Our approach is similar to the method proposed by Jian-Hao Luo (Luo and Wu 2017), where they propose a pruning method using the entropy values of the entire outputs of features maps produced in the convolutional layers using the test dataset. Instead, we use the maximum value in each position of the generated feature map, obtaining a summary of those outputs. Lou approach is used for pruning classification models (AlexNet (Krizhevsky, Sutskever, and Hinton 2012), VGG (Simonyan and Zisserman 2014), GoogleNet (Szegedy et al. 2015) and ResNet (He et al. 2016)), instead of that we transfer the kernels for a new model in a new task.

## 4 MEASURING TASKS DISTANCE

Having a set of source tasks it is important to select one that will obtain an effective training (avoid negative transfer and in some cases outperform the baseline performance) in the target task. In order to solve the last problem we propose a novel distance measure based in two important elements of a MDP: the state space (for our experiments, frames of a video game) and the action spaces (combination of joystick and a button).

To compare the state space we use the output of a hidden layer using the pre-trained model in a source task. Also, we propose a comparison between action spaces (source and target task). The proposed measure can be seen in Equation 1, where $cnn_s$ is a pre-trained CNN in the source task and $A_k$, $A_t$ are the action spaces of the source and target task respectively. Each part of the distance are described in the next subsection.

$$d(cnn_t, A_k, A_t) = \alpha \cdot dist_1(cnn_s) + (1-\alpha) \cdot dist_2(A_k, A_t) \quad (1)$$

Finding the optimal value for $\alpha$ could be a limitation of the proposed measure and it is important to select this value carefully. In preliminary tests we found that a value of $\alpha = 0.5$ provides a good equilibrium between the two parts of the measure, as shown in the experimental results. In order to get both parts of the measure in the same range $[0, 1]$ we normalize the outputs.

## Comparing State Spaces

For comparing the state space we use samples of instances of the source and target tasks and evaluate them into a pre-trained model of the source task. The proposed method for obtaining the first part of the Equation 1 is shown in Figure 4, where each instance of the source and target samples are evaluated in the pre-trained model of the source task. The rationale of the measure is to find if instances of the source and target tasks produce similar distributions in the output of a hidden layer. The main assumption is that similar distributions imply high relevance of the source task to the target task. We use the Wasserstein metric (Vallender 1974) to find the distance between these two distributions. We use this metric because of the advantages between other measures as the kullback-leibler divergence that not satisfy the triangle inequality.

To apply the Wasserstein metric in both tasks we find a discrete distribution of each attribute ($a \in At$, where $At$ is the attribute space) of the output in a hidden layer of a CNN. So we use a numerical vector after applying a flatten operation over the output of the selected convolutional layer, in this case we use the deepest convolutional layer. Then, we will have a distribution for each attribute for both samples source and target. Then, we find the Wasserstein distance between the corresponding distribution. Finally, we sum the results of the distances and the obtained value is the result of the comparison between state space.

We propose to find the distribution of each attribute in order to obtain specific information about how is the behaviour of both tasks in the model, while using the entire values extract general information about the outputs of the model.

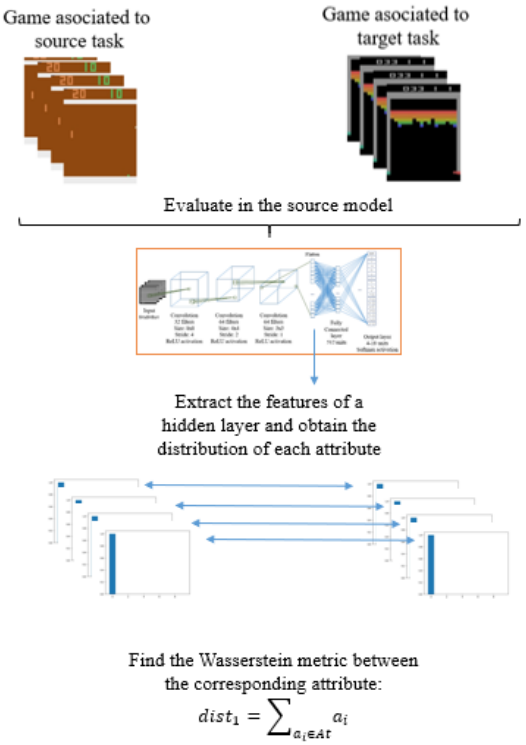



Figure 4: Method to obtain the first part of the measure.

## Comparing Action Spaces

In order to compare the action spaces ($A_k$ and $A_t$, action spaces of source and target action spaces, respectively) we propose the Equation 2. This equation returns the maximum distance value if both action spaces are disjoint, in other case, it compares the intersection between the two sets in the first part, and the cardinality of the difference in both directions, source and target action spaces, with respect to the universe of actions ($A_\mathcal{U}$). In the numerator of Equation 2 we add one to avoid divisions by zero and we use the cardinality of the universe of actions to compare all the source tasks.

$$d_2(A_k, A_o) = \begin{cases} 2 \textbf{ if } A_k \cap A_o = \emptyset \\ \frac{1}{|A_k \cap A_o|} + \frac{|A_o - A_k| + 1}{A_\mathcal{U} - |A_k - A_o|} \textbf{ else} \end{cases} \quad (2)$$

The behaviour of the Equation 2 can be seen in Figure 5, where we can observe that higher differences imply higher distance values, while smaller differences produce the smaller distance values. The equation, contrary to other measures like the Jaccard index (Real and Vargas 1996), considers the entire action space of all the tasks, and it need to be asymmetric in order to compare the difference between both task, if this value is higher the performance of the selected task could not have good performance in the target task.

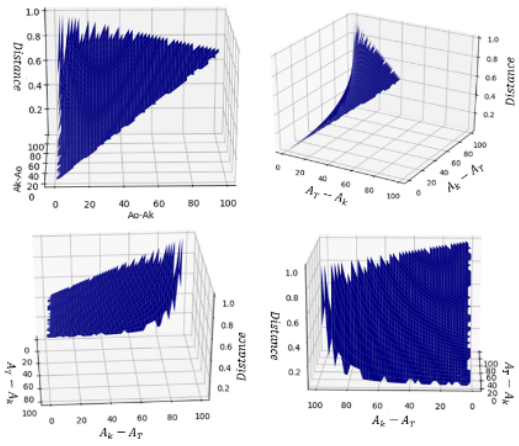



Figure 5: Behaviour of the Equation 2 using an action space of 100 for both source and target task. The $x$ and $y$ axes correspond to the difference between the sets of action spaces $|A_k - A_o|$ and $|A_o - A_k|$, $z$ represents the value of the second part in Equation 2.

## 5 SELECTING RELEVANT KERNELS

In this section we describe the method for selecting the most relevant kernels of the source model to the target task. The hypothesis is that those kernels that produce outputs with diverse values with different inputs, such as the one plotted in Figure 1b are more relevant than kernels that that produce a uniform output, like the plotted in Figure 1a. To measure diversity we use entropy.

The proposed method for selecting the kernels works as follows. First, we evaluate samples of the target task over a pre-trained model in the source task, to obtain a feature map of each kernel. Then, we seek for the maximum value in each position of the feature map, here we combine all the outputs in a new feature map with the maximum value in each position. In order to obtain the entropy value of each kernel we discretize the combined feature map using histograms (10 bins with size=0.1) after a normalization process with values between [0,1] (Figure 6).

Finally, in each layer we select those kernels with the largest entropy values to build a new model for the target task, we transfer a percentage of kernels in each convolutional layer. Then we fine-tune the model as the baseline training. In the next section we report results with different percentages of kernels.

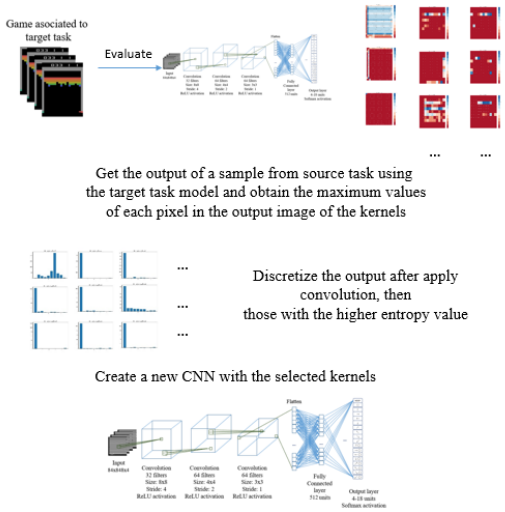



Figure 6: Kernel selection method.

## 6 RESULTS

We use the original DQN architecture to test our proposed method (Mnih et al. 2015), but other architecture could be used as well. Due to processing time restrictions we select nine games which obtained better human performance, then among them we seek for a uniform selection based on the normalized performance (three of the first-top games, three in of the middle-top and three of the worst): Atlantis (AT), Boxing (BX), Breakout (BR), Freeway (FW), Gopher (GP), Pong (PN), Space Invaders (SI), Up and Down (UD), and Video Pinball (VP).

The source architecture follows the original DQN architecture $\{32,64,64,512\}$, the first three number correspond to the number of filters in the convolutional layers and the last one to the number of neurons in the fully connected layer. For our experiments we transfer different percentages of kernels in each convolutional layers: 25% $\{8,16,16,128\}$ 50% $\{16,32,32,256\}$, and 75% $\{24,48,48,684\}$. The fully connected layer and the output one are initialized with random numbers using Glorot's initialization (Glorot and Bengio 2010).

We use dopamine-rl library (Castro et al. 2018) to test our approach, this library follows the recommendations of (Machado et al. 2018), so an iteration corresponds to train the model with 2,500,000 instances followed by 125,000 instances of testing with an $\epsilon - greedy$ policy with $\epsilon = 0.001$. We use as baseline the best accumulated reward of five independent runs that are provided by dopamine-rl experiments (also, this models are used to apply our TL approach). Two modifications are applied to the DQN algorithm in our experiments: we use prioritized experience replay and instead of RMSprop we use Adam optimizer.

In the experiments, we reported only the results of the best two tasks selected by our proposed measure with $\alpha = 0.5$ due to hardware limitations. In preliminary experiments we noticed that the best source to transfer was always among the top two tasks selected by our measure.

The top two selected tasks by our measure for 25%, 50%, and 75% of transferred kernels, are presented in Table 1, in the columns 1st and 2nd. The selected tasks are different because we apply first the kernel selection, then we evaluate the obtained models with our distance measure.

For our experiments we use a 8-GPU Nvidia GTX 1080 Ti computer with 128GB in RAM memory, an iteration of each experiment finish in approximately forty minutes. The number of iterations that takes each transferred model to reach the baseline score and the time gain for different percentages of kernel transferring are shown in Table 1, for 25%, 50% and 75% of transferring, respectively.

It can be seen that after training the target models with 25% of the kernels we obtain scores that are close to the baseline in 3/9 games and only one of them outperforms the baseline scores (see Figure 8a). We hypothesize that the size of the model is not big enough to capture the relevant patterns of the games and it is also very likely that relevant kernels are discriminated because of the target model size. When we transfer 50% of the kernels there is a clear improvement over the previous experiment. The results shows that in 7/9 games the simplify model outperforms the baseline scores, in one game it obtains a similar performance, and in one game it has a clearly worst performance (see Figure 8b).

The worst case is the selection of the Boxing model for the game Gopher, however, the second selected model outperforms the baseline score. After looking more closely to the Gopher instances in the Boxing pre-trained model we can appreciate that this produces uniform outputs (every position in the feature map has zero values), where the kernels produce negative values and ReLU activation deviate this values to zero as can be seen in Figure 7. As future work we

| | 25% | | | | | | 50% | | | | | | 75% | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Target | 1st | I | Gain | 2nd | I | Gain | 1st | I | Gain | 2nd | I | Gain | 1st | I | Gain | 2nd | I | Gain |
| BX | GP | +50 | 0 | VP | +50 | 0 | SI | 25 | 16.6 | GP | 35 | 10 | PN | 29 | 14 | GP | +50 | 0 |
| FW | UD | +50 | 0 | GP | +50 | 0 | UD | 22 | 7.3 | GP | 21 | 8 | UD | 17 | 10.66 | BX | +50 | 0 |
| UD | BX | +50 | 0 | VP | 48 | 0 | GP | 29 | 10 | VP | 31 | 8.6 | FW | 15 | 19.33 | BX | +50 | 0 |
| AT | VP | +50 | 0 | SI | +50 | 0 | VP | +50 | 0 | GP | +50 | 0 | PN | 34 | 10 | SI | 33 | 10.6 |
| PN | GP | +50 | 0 | SI | +50 | 0 | SI | 44 | 4 | GP | 28 | 14.6 | GP | 47 | 2 | SI | +50 | 0 |
| BR | VP | +50 | 0 | GP | +50 | 0 | PN | 46 | 2.6 | SI | +50 | 0 | PN | 45 | 3.33 | SI | +50 | 0 |
| GP | BX | +50 | 0 | VP | +50 | 0 | BX | +50 | 0 | SI | 41 | 5.3 | PN | 26 | 15.33 | SI | 34 | 10 |
| SI | VP | +50 | 0 | BX | +50 | 0 | BX | +50 | 0 | GP | +50 | 0 | PN | +50 | 0 | GP | +50 | 0 |
| VP | BX | +50 | 0 | AT | +50 | 0 | SI | 38 | 0 | PN | 45 | 0 | PN | 43 | 0 | UD | +50 | 0 |

Table 1: Iterations and gain time in hours to reach the baseline, the full training time is 34 hours, "+50" means negative transfer, I=iterations.

will consider this special case to avoid it.

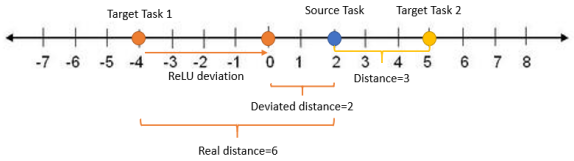



Figure 7: Deviation produced by ReLU activation function.

In the case where we transfer 75% of the kernels we can see a similar behavior than when we transfer only 50%. In 8/9 games the performance of out approach outperform the maximum baseline score, and only one game (Space Invaders) obtains negative transfer. We can see that our measure obtain better prediction when the percentage of transferring is higher. While, transferring 50% predicts a useful model in 4/9 games in the fist position, but in the same number of cases (4/9) the second nearest task obtain better performance than the nearest one according to our distance measure.

From the experiments, we can see that: (i) it is possible to use transfer learning to learn a new game using a different game model and (ii) we can even use a simplified model and learn is less iterations (iii) the proposed measure shows most robustness while the percentage of transfer increase, 75% predicts 7/9 useful models as the nearest one, while 50% predicts a useful model in 4/9 cases as the nearest and the same in the second position (iv) in the best cases our experimental results show that an agent could be trained in few hours transferring knowledge to a new case (i.e. UD to FW case arise baseline performance in 14.67 hours while the full experiment take 34 hours) and in other cases take more time but the baseline performance is outperformed (PN to VP took more time to reach the baseline performance but they reach a 1.59 better performance).

## 7 CONCLUSIONS AND FUTURE WORK

We present a new distance measure to select relevant source tasks for a target task. Although not shown for all the games, i.e., we cannot guarantee, without further experiments, that we are truly selecting the best source, the best selected task with our measure is able to outperform in most of the cases the baseline. We also propose a method to select the most relevant kernels of a pre-trained model. The experimental results show that we can transfer half of the kernels and still outperform, in most of the cases, the baseline model (the best accumulated reward of five independent runs of dopamine-rl library (Castro et al. 2018)).

As future work, we would like to establish a clear criterion to decide the right percentage of kernels to transfer depending on the source and target tasks. We would like to do experiments with other DRL methods, with more games, and in different domains, in particular, for solving classification problems.

## Aknowledgements

The authors thankfully acknowledge computer resources, technical advice and support provided by Laboratorio Nacional de Supercómputo del Sureste de México (LNS), a member of CONACYT national laboratories with the project No. 201901047C. Jesús García-Ramírez acknowledges CONACYT for the scholarship that supports his PhD studies associated to CVU number 701191.

## References

Bellemare, M. G.; Naddaf, Y.; Veness, J.; and Bowling, M. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research* 47:253–279.

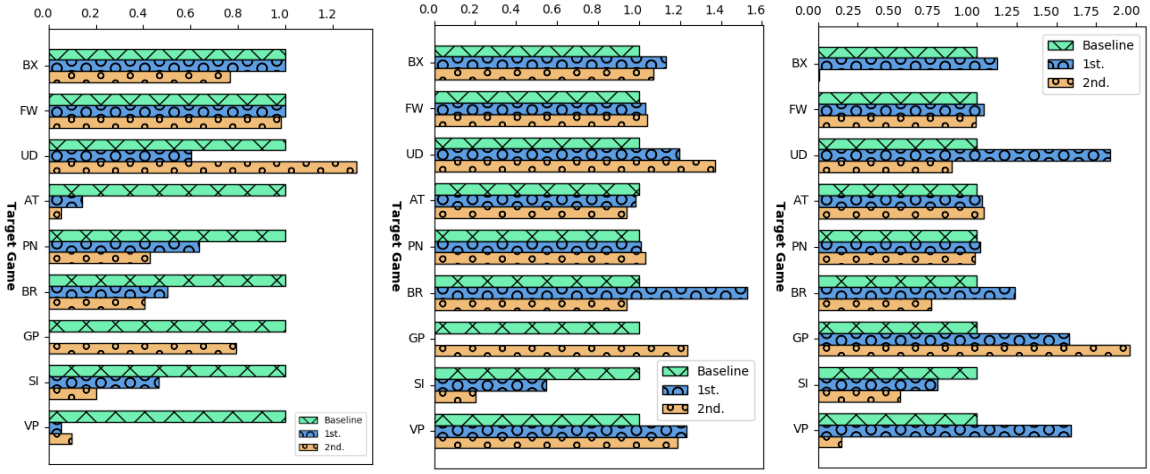
Bellemare, M. G.; Dabney, W.; and Munos, R. 2017. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 449–458. JMLR. org.

Carr, T.; Chli, M.; and Vogiatzis, G. 2018. Domain adaptation for reinforcement learning on the atari. *arXiv preprint arXiv:1812.07452*.

Castro, P. S.; Moitra, S.; Gelada, C.; Kumar, S.; and Bellemare, M. G. 2018. Dopamine: A Research Framework for Deep Reinforcement Learning.

de la Cruz, G.; Du, Y.; Irwin, J.; and Taylor, M. 2016. Initial progress in transfer for deep reinforcement learning algorithms.

Glorot, X., and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Pro-*




(a) 25% (b) 50% (c) 75%

Figure 8: Maximum score of training with transfer, normalized by the maximum of the baseline.

*ceedings of the thirteenth international conference on artificial intelligence and statistics*, 249–256.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

He, Y.; Liu, P.; Wang, Z.; Hu, Z.; and Yang, Y. 2018. Filter pruning via geometric median for deep convolutional neural networks acceleration.

Hessel, M.; Modayil, J.; Van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.; and Silver, D. 2018. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

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

Li, H.; Kadav, A.; Durdanovic, I.; Samet, H.; and Graf, H. P. 2016. Pruning filters for efficient convnets.

Luo, J.-H., and Wu, J. 2017. An entropy-based pruning method for cnn compression.

Machado, M. C.; Bellemare, M. G.; Talvitie, E.; Veness, J.; Hausknecht, M.; and Bowling, M. 2018. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research* 61:523–562.

Mittel, A.; Munukutla, S.; and Yadav, H. 2018. Visual transfer between atari games using competitive reinforcement learning. *arXiv preprint arXiv:1809.00397*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, 1928–1937.

Pan, S. J., and Yang, Q. 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22(10):1345–1359.

Parisotto, E.; Ba, J. L.; and Salakhutdinov, R. 2016. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*.

Real, R., and Vargas, J. M. 1996. The probabilistic basis of jaccard's index of similarity. *Systematic biology* 45(3):380–385.

Rusu, A. A.; Colmenarejo, S. G.; Gulcehre, C.; Desjardins, G.; Kirkpatrick, J.; Pascanu, R.; Mnih, V.; Kavukcuoglu, K.; and Hadsell, R. 2015. Policy distillation. *arXiv preprint arXiv:1511.06295*.

Rusu, A. A.; Rabinowitz, N. C.; Desjardins, G.; Soyer, H.; Kirkpatrick, J.; Kavukcuoglu, K.; Pascanu, R.; and Hadsell, R. 2016. Progressive neural networks. *arXiv preprint arXiv:1606.04671*.

Schaul, T.; Quan, J.; Antonoglou, I.; and Silver, D. 2015. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

Schmitt, S.; Hudson, J. J.; Zidek, A.; Osindero, S.; Doersch, C.; Czarnecki, W. M.; Leibo, J. Z.; Kuttler, H.; Zisserman, A.; Simonyan, K.; et al. 2018. Kickstarting deep reinforcement learning. *arXiv preprint arXiv:1803.03835*.

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

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel,

T.; et al. 2018. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science* 362(6419):1140–1144.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sutton, R. S.; Barto, A. G.; et al. 2018. *Introduction to reinforcement learning*. MIT press Cambridge.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1–9.

Taylor, M. E., and Stone, P. 2009. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research* 10(Jul):1633–1685.

Vallender, S. 1974. Calculation of the wasserstein distance between probability distributions on the line. *Theory of Probability & Its Applications* 18(4):784–786.

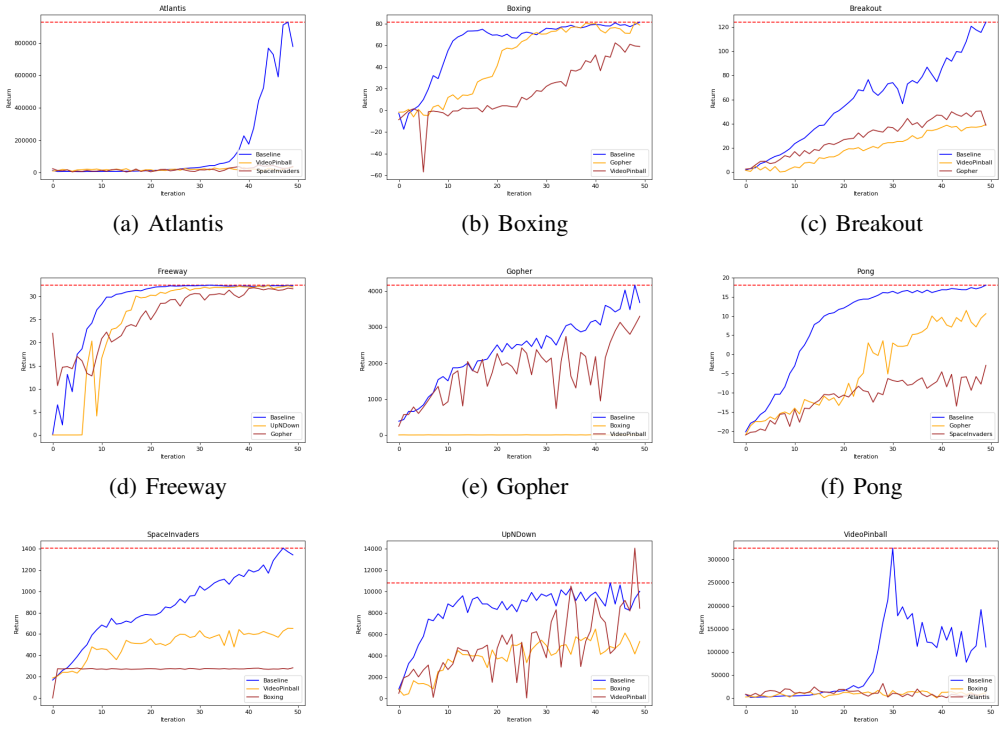
Wang, Z.; Schaul, T.; Hessel, M.; Van Hasselt, H.; Lanctot, M.; and De Freitas, N. 2015. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*.

Yin, H., and Pan, S. J. 2017. Knowledge transfer for deep reinforcement learning with hierarchical experience replay. In *AAAI*, 1640–1646.

Zambaldi, V.; Raposo, D.; Santoro, A.; Bapst, V.; Li, Y.; Babuschkin, I.; Tuyls, K.; Reichert, D.; Lillicrap, T.; Lockhart, E.; et al. 2018. Deep reinforcement learning with relational inductive biases.

# Appendix A. Learning Curves

In this section we present the full learning curves of our experiments, the baseline of our experiments is the best score obtained by five independent runs during the first 50 iterations. The learning curves are shown in Figures 9, 10 and 11.

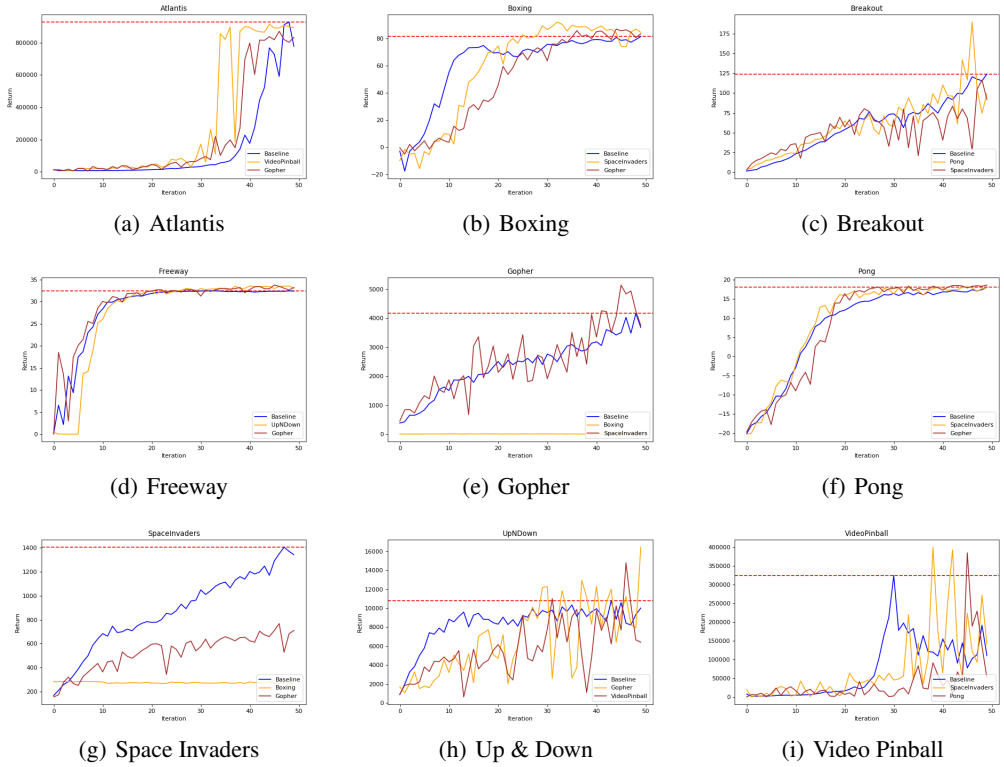



(a) Atlantis (b) Boxing (c) Breakout

(d) Freeway (e) Gopher (f) Pong

(g) Space Invaders (h) Up & Down (i) Video Pinball

Figure 9: Learning curves when we transfer 25% of the kernels.

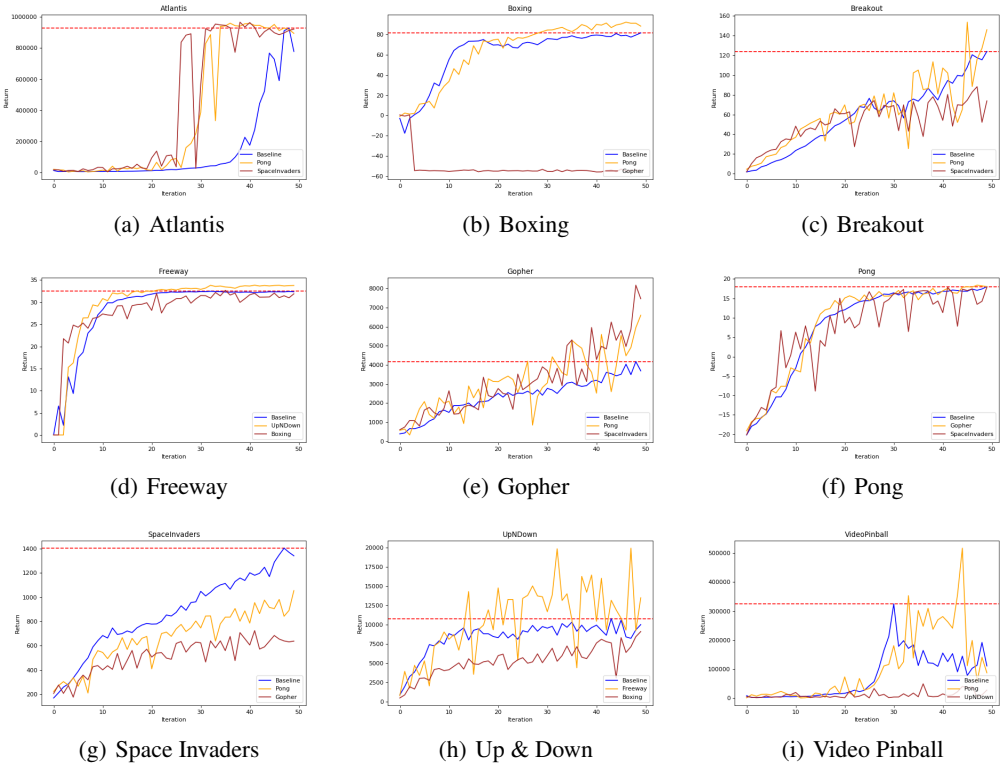



(a) Atlantis (b) Boxing (c) Breakout

(d) Freeway (e) Gopher (f) Pong

(g) Space Invaders (h) Up & Down (i) Video Pinball

Figure 10: Learning curves when we transfer 50% of the kernels.




(a) Atlantis (b) Boxing (c) Breakout

(d) Freeway (e) Gopher (f) Pong

(g) Space Invaders (h) Up & Down (i) Video Pinball

Figure 11: Learning curves when we transfer 75% of the kernels.