VISUAL ANALOGIES BETWEEN ATARI GAMES FOR STUDYING TRANSFER LEARNING IN RL

Doron Sobol[1]**, Lior Wolf**[1,2] **& Yaniv Taigman**[2]
[1] School of Computer Science, Tel-Aviv University
[2] Facebook AI Research

ABSTRACT

In this work, we ask the following question: Can visual analogies, learned in an unsupervised way, be used in order to transfer knowledge between pairs of games and even play one game using an agent trained for another game? We attempt to answer this research question by creating visual analogies between a pair of games: a source game and a target game. For example, given a video frame in the target game, we map it to an analogous state in the source game and then attempt to play using a trained policy learned for the source game. We demonstrate convincing visual mapping between four pairs of games (eight mappings), which are used to evaluate three transfer learning approaches.

1 INTRODUCTION

One of the most fascinating capabilities of humans is the ability to generalize between related but vastly different tasks. A surfer will be able to ride a snowboard after much less training than a beginner in board sports; a gamer experienced with adventure games will solve escape rooms long before the one hour is up; and a veteran tennis player will often top the office's ping pong league. The goal of this work is to check if an agent trained via Reinforcement Learning (RL) can gain such an ability using visual analogies: an actor is trained and evaluated on a target task after learning a source task in the typical reinforcement learning setting and is also provided with mappers that given a frame in either game, are able to generate the analogous frame in the other game.

The bidirectional mappers between the video sequences are based on recent computer vision approaches for the task of finding visual analogies, in combination with an added regularization term. We evaluate our methods on two groups of games, and are able to successfully learn the mappers between all same-group pairs.

Building on the existence of these mappers, we propose several Transfer Learning (TL) techniques for utilizing information from the source game when playing the target game. These methods include techniques such as data-transfer and distillation. Unfortunately, none of these methods seem to be helpful, maybe with the one exception of a method, which trains on scenes that are visually adapted from the source game to the domain of the target game.

Despite the moderate success, we believe that our work presents value to the community in multiple ways. First, we are successful at the challenging video conversion task, which could benefit future efforts. Second, we devise a few possible TL methods that "almost work". Third, a critical view of the practical value of TL in the current RL landscape is seldom heard. Lastly, by sharing our results, code, and models, we hope to help others in minimizing wasted efforts.

1.1 RELATED WORK

While there has been a few contributions that employ reinforcement learning for computer vision tasks, e.g., Caicedo & Lazebnik (2015) and works that aim to solve perceptual problems in robotics using reinforcement learning, e.g., Kwok & Fox, we are not aware of a work that employs computer vision in order to promote transfer learning in sequential decision making.

In both computer vision and RL, Generative Adversarial Networks Goodfellow et al. (2014) (GANs) have created a sizable impact. In computer vision, GANs allow for the generation of realistic im-

ages using deep neural networks. In reinforcement learning, GANs have opened new avenues for imitation learning and similar forms of transfer by promoting domain confusion, e.g., making sure that the expert and the imitator are playing in indistinguishable ways.

Generating Visual Analogies The field of image to image translation has seen major advancement in the last few years with the presentation of new image to image GAN based translation methods such as DiscoGAN Kim et al. (2017), CycleGAN Lu et al. (2017), UNIT GAN Liu et al. (2017) and DistanceGAN Benaim & Wolf (2017). Those methods use some form of unsupervised heuristic alignment loss, to generate analog images in the target domain using an input image from the source domain.

Reinforcement Learning Methods Recent work in Reinforcement Learning such as the Asynchronous Advantage Actor-Critic (A3C) algorithm (Mnih et al., 2016) has made running reinforcement learning algorithms faster by allowing it to run on multiple GPUs in parallel. This is done by calculating the gradients of multiple different training episodes in parallel, and made running large amounts of experiments feasible, and therefore made this work possible. While other reinforcement learning algorithms such as rainbow DQN Hessel et al. (2017), Double DQN Van Hasselt et al. (2016) and other extensions of DQN Mnih et al. (2013) may be more efficient and achieve better results using less training episodes, they are less practical when running large amounts of experiments due to the fact that they can not be paralleled and therefore take more time in a real world scenario.

Transfer Learning in Reinforcement Learning Transfer learning in reinforcement learning has been studied in Rusu et al. (2016), where TL takes place between different Atari games by using the activations of every level of the network in the source domain as an additional input for the next layer in the target domain. This work does not consider the visual similarities between the environments. The results are evaluated using the area under the curve of the mean reward over the training session, which is high even in cases where the training is somewhat more successful at earlier stages but there is no improvement in overall performance.

Concurrently with our work, Gamrian & Goldberg (2018) have researched the use of unsupervised image to image translation to overcome visual differences between Atari games. In comparison to our work, they used the same base game as the target and the source game and only slightly changed the visual part of the target game by adding a constant noise to the game frames - for example green lines over the screen or a rectangle in a certain location. While in our work we transfer knowledge between completely different games.

2 SETTINGS

Our objective is to train an agent on the target game T in the most efficient way using knowledge from the source game S. In phase one the algorithm has unlimited number of diverse frames from both the target game and the source game, but does not have access to the interactive games environments or to the reward functions of the games environments. In addition the algorithm is not aware of the actions taken during the frame's production. In phase two, the algorithm is given the environment of both games and its goal is to train an agent on the target game with the least amount of training episodes from the target game, but with no limitations on the number of training episodes from the source game. In this work, the data of the first phase, namely frames from t and s is used to train (in an unsupervised way) the visual analogies mappers $G : s \rightarrow t$ and $G^{-1} : t \rightarrow s$. In phase two we are using those mappers and apply different TL methods from S to T in order to train an agent on the target game T in the most efficient way.

Learning Cross-Domain Video Mapping To create visual analogies between a pair of games, we need a large amount of relatively variant frames from both the source and the target games. We collect those frames offline by using an actor. This actor does not need to be an expert and we do not need to know the action it applies or to imitate it. However, it is required that the states that are generated by this agent are diverse enough and therefore the actor is required to remain in the game for a while on different game scenarios.

Despite recent advancement in unsupervised mapping between visual domains, the task of mapping between games on the raw image data is still too challenging. We therefore apply the following preprocessing steps, depicted in Fig 1:
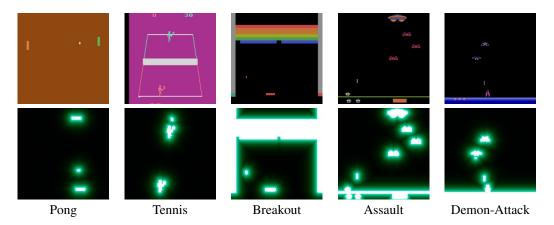
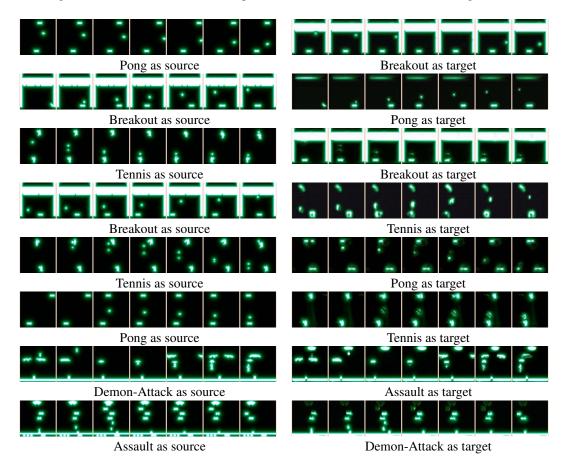Figure 1: The obtained attention maps for a frame from each of the five tested games.

Figure 2: Images of samples of consecutive frames from the source game (left) to the target game (right of the same line).

1. Rotating the frames, if needed, so that the main axis of motion is horizontal. For example in the game Pong the paddle movement is vertical and so we rotate the frames by 90 degrees in order for it to match the paddles movement of breakout and tennis.

2. Extracting the important features of the games by applying an attention operator to the frame, this is done by subtracting either the median pixel value at each location or the median pixel value of the entire image (depending on the game), and then applying a threshold to obtain a binary image.

3. Enlarging the relevant features in the frame by applying a dilation filter on the image. This is needed due to the fact that some small items, even 1-2 pixels in size, can be very important (e.g., bullets fired or the ball).

4. Creating three channels with varying visual scales by cloning the dilated image and applying two levels of blurring.

In order to perform the visual mapping, we modify the UNIT GAN method Liu et al. (2017). Namely, we train the mapper functions G and G^{-1}, using the network architecture of UNIT GAN. The UNIT GAN architecture uses two encoders E_s and E_t and two decoders Dec_s and Dec_t such that $Dec_s \circ E_s$ is a variational auto encoder (VAE) of domain s and $Dec_s \circ E_s$ is the mapper G. This yields the following losses for each domain , (Those losses are for the direction from s to t but the same losses apply in the opposite direction and trained simultaneously):

$$L_{VAE_t} = KL(p_t(z_t|x_t)||p_\eta) - \mathbb{E}_{z_t \sim p_t(z_t|x_t)} \left[\log p_{Dec_t}(x_t|z_t) \right]$$

Where $p_t(z_t|x_t)$ is the distribution of the latent vector, the prior distribution, p_η is a zero mean Gaussian distribution and p_{Dec_t} is the distribution that Dec_t generates.
The GAN loss on the images generated from the generators:

$$L_{GAN_t} = \mathbb{E}_{x_t \sim P_t} \left[log D_t(x_t) \right] + \mathbb{E}_{z_s \sim p_s(z_s|x_s)} \left[1 - \log D_t(Dec_t(z_s))) \right]$$

Where D_t is the discriminator of domain t.

The cycle consistency loss which means that the latent vector of a sample $x_t \in t$ is the same as that of $Dec_t(E_s(Dec_s(E_t(x_t))))$, is given by:

$$L_{cc_t} = KL(p_s(z_s|x_s)||p_\eta) + KL(p_t(z_t|Dec_t(E_s(x_s)||p_\eta))) -$$
$$\mathbb{E}_{z_t \sim p_t(z_t|Dec_t(E_s(x_s))} \left[\log p_{Dec_s}(x_s|z_t) \right]$$

Each encoder E_s/E_t contains three encoding layers followed by three residual blocks followed by one shared layer for both encoders. The decoders are a mirror of those layers in the reverse order and use de-convolutions instead of convolutions.

Those losses, by themself, fail to produce a desirable solution due to mode collapse. In order to fix this, we add the gradient-penalty regularization term of improved WGAN (Gulrajani et al., 2017), adapted to the problem of cross-domain mapping:

$$L_{GP} = \mathbb{E}[(||\nabla_{\hat{x}} D(\hat{x})||^2 - 1)^2]$$

where $\hat{x}$ is either $\hat{s} = \epsilon s + (1-\epsilon)G^{-1}(t)$ or $\hat{t} = \epsilon t + (1-\epsilon)G(s)$, D is the GAN's discriminator, s are samples of S, t are samples of T, and $\epsilon \sim U[0,1]$. Therefore we optimize on:

$$\min_{E_s, Dec_s, E_t, Dec_t} \max_{D_s, D_t} L_{VAE_s}(E_s, Dec_s) + L_{VAE_t}(E_t, Dec_t) + L_{GAN_s}(E_s, Dec_s, D_s)$$
$$+ L_{GAN_t}(E_t, Dec_t, D_t) + L_{cc_s}(E_s, Dec_s, E_t, Dec_t)$$
$$+ L_{cc_t}(E_t, Dec_t, E_s, Dec_s) + L_{GP_s}(D_s) + L_{GP_t}(D_t)$$

This regularization term on the discriminator makes the discriminator within the space of 1-Lipschitz functions, which makes the problem optimized by the GAN to be a Wasserstein distance function which has better theoretical properties, and makes the GAN converge to a better function and not to collapse. The weights we use to balance between the various loss terms are fixed throughout the experiments.

3 TRANSFER LEARNING METHODS

Training the strategy π_t for the target game and the strategy π_s of the source game (when used), is done with the asynchronous advantage actor-critic (A3C) algorithm (Mnih et al., 2016). The network architecture consists of four consecutive layers of convolutions, each layer i with kernels of size k_i (see below), and depth of size d_i; each of the layers is followed by max pooling of size two and the ReLU activation function. The convolutional stack is followed by a recurrent LSTM layer with λ hidden neurons. The network is topped with two fully connected layers, used to predict the action and its corresponding value.

We evaluated various methods of transferring knowledge between games, and those are the most promising three methods: (i) Data Transfer by Pretraining, which uses pretraining on the converted source game in order to bootstrap the training process, (ii) Continuous Data Transfer, which uses episodes from the converted source game as an additional training data, and (iii) Distilation, which uses the visual mapping obtained by G^{-1} between the source and the target, and π_s the policy of the source game, in order to initialize the network activations of the target agent.

Data transfer by pretraining In every pair of games we chose, there are major similarities between both the rewards and the game dynamics. Therefore, we can assume that after applying the visual transformation G on the source game S, the converted source game is a remote approximation of the target game. This transfer method finetunes, on the target game, an agent that was pretrained of the converted source frames: We first transform frames from the source game s using G to get the remote approximation of the target game. We then train a policy π_t using the A3C algorithm on these converted frames using the reward of the source game and using a static mapping of actions from the source game actions to the target game actions. We then fine tune the resulting policy on the target game, obtaining π_t. By a static mapping of the actions we mean, for example, that UP in the Pong game might correspond to RIGHT in Breakout and therefore the actions need to be adjusted. If the method works, one can try all possible permutations and select the best mapping. In our experiments, we provide this mapping, thus providing an upper bound on the actual performance.

Continuous data transfer In this method, we do not separate the training into multiple phases and instead provide a continuous training signal from the source game while training π_t. Namely, we train the target policy π_t on mixed samples from both $G(s)$ and t throughout the entire training process. This method could potentially help the agent overcome overfitting by creating a more general agent.

Distilation The methods above employed the mapping G and trained in the visual domain of the game T. This method employs G^{-1} and a pretrained source game policy π_s. A direct approach of fine-tunning $\pi_s \circ G^{-1}$, leads to a very large network, mainly since the image to image GAN G is relatively deep and consumes most of the training time, even though it is not being updated. We therefore train a network π_{mimic} of the same architecture as π_t in order to mimic $\pi_s \circ G^{-1}$, on the set unsupervised frames from the target game. We extract the convolution layers from the mimicking network and use them to initialize the convolution layers of a new network π_t. We then continue to train π_t using game data from the target domain. Therefore we distill the spatial information learned by π_s into a new network π_t, which is then fine-tuned on the target game (using all layers of π_{mimic} provides inferior performance).

4 RESULTS

The experiments are conducted on five Atari games, split into two groups. The first group contains the games Breakout, Tennis and Pong, in which the player has a paddle it controls and its goal is to achieve a certain objective by hitting the ball in a certain way. The second group of games are Demon-Attack and Assault, in which the player has a spaceship it controls and it needs to shoot the targets (similar to Space Invaders). Despite some effort, we were not able to identify other potential pairs among the Atari games for TL. The two groups give rise to four pairs of games, which yield eight transfer directions.

The Experiments are conducted on every pair of games in the following way:

1. A big number of samples are sampled from both games using the untrained agent described in Sec. 2.
2. We train the mappers G and G^{-1} using the method described in Sec. 2. We manually stop the training when we achieve visually adequate results.
3. We train π_t using one of the three transfer learning methods described in Sec. 3.

In Fig. 2 there are samples of the transferred frames between the target game and the source game and vice versa which were obtained by using the method described in Sec. 2, the images show the mapping between sequences in the source game (left) to the converted images of the target game. As can be seen from the correlation between the balls locations, and the paddle or the tennis player in the games Tennis, Pong and Breakout, and from the correlations between the spaceships and

the enemies locations in the games Assault and Demon-Attack, the mappings obtained using the improved UNIT GAN seem to convey the semantics of the games and to align frames from the source game to the target game in a reasonable fashion.

The architecture of the policy networks π_t and π_s is described in Sec. 3. We set $\lambda = 512$, $k_1 = k_2 = 5$, $k_3 = 4$ and $k_4 = 3$, $d_1 = d_2 = 32$, and $d_3 = d_4 = 64$, and max pooling with window size 2. In the RL phase, the training of the agent is done using the A3C algorithm with 16 processes for training and 6 processes for the test in order to achieve stable results. When using a transfer learning method that uses pretraining on the source domain, the source game is pre trained using the same parameters as the target game until it converges. As for the details of the games environment itself we use the deterministic version of the games, and in order to speed up training we skip some of the frames so that the agent is only looking at every fourth frame. When training with the continuous data transfer method, we train with 12 processes for the converted source and 4 processes for the target. We do that since training on the converted source game requires more processing power as it also needs to transfer the frames, and this division empirically yields the best results. Lastly, when using the distillation transfer method, we start by sampling frames from the target games. We do that using the same untrained agent that was used for sampling the unlabeled frames in Sec. 2 . Then, for every sample $t \in T$ we calculate $\pi_s(G^{-1}(t))$ as the labels of t, and train π_{mimic} using cross entropy loss.

The success of the TL methods is shown in Fig. 3, which depicts the training progress (local average reward as a function of the number of target domain sessions). Note that when trained using the second method (continues data transfer), we only count the training episodes from the target game. The results are not conclusive and we employ a subjective rating scale for describing the level of success of the TL methods on a given pair of games. A method is considered successful in transferring knowledge from the source game, if reaching a certain level of performance requires less supervised training samples in the target domain than the baseline method of vanilla training in that domain. We distinguish between three levels of success:

1. Upon convergence or reaching the maximum possible reward, the method that employs TL outperforms the baseline method - either in the actual reward or in the number of training episodes needed to reach that reward.

2. The TL method achieves almost all levels of performance between the random performance and the converged performance with less samples than the vanilla method.

3. The TL method achieves non-trivial levels of performance faster than the baseline method but then stops leading.

We also employ a star (*) to denote situations in which the TL method starts off, without any supervised samples from the target domain, in a level that is significantly better than random. This can happen with any level of success. In addition we employ a dash (-) to indicates a lack of success.

Tab. 1 shows the level of success reached by the various methods, in comparison to the baseline method. While the scoring is subjective, the table suggests that the first method of data transfer for the purpose of pretraining is the only method to consistently outperforms the baseline. Fig 3 contains the full training logs. Note that using the Mean Reward Evaluation score, as done in Rusu et al. (2016), would indicate that our methods mostly outperform the baseline. However, this can happen with each of the level of success above as can be seen in Fig. 4.

5   RESULTS ANALYSIS

It is easy to identify potential shortcoming of visual analogies. First, the analogies are partial, e.g., the opponent in Pong does not exist in Breakout. Second, the dynamics are of a different nature. For examples, in the game Tennis, the game is a projection of 3D world to 2D, which makes the ball move in a slight curve and in different speeds depending on the ball location, unlike Pong or Breakout. There are additional differences such as the speed of the ball relative to the paddle, speed of the bullet relative to the spaceship and the targets, etc.

We try to augment the game's speed in order to tackle some of these issues. The speed issue arises when transferring between the games Tennis and Pong where the speed of the tennis player with respect to the ball is slower relative to the speed of the Pong paddle with respect to the ball. This
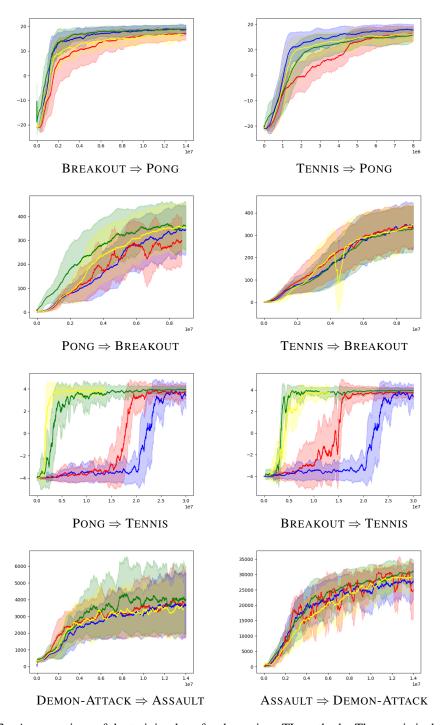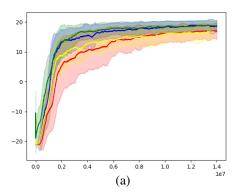
Figure 3: A comparison of the training logs for the various TL methods. The x-axis is the number of training steps, and the y-axis is reward. The plots are averaged over three independent runs where the transparent color is the variance. The blue line is the baseline, the red is distillation, the yellow is continuous data transfer and the green is data transfer for pretraining.
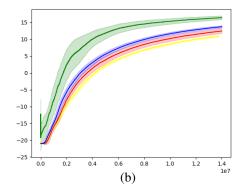
Figure 4: A side by side comparison between the mean reward over the entire training process and the current reward of the training for the various TL methods on the game Pong. The x-axis is the number of training steps, and the y-axis are the mean reward over the entire training until this point (a) or the current reward (b). The plots are averaged over three independent runs. The blue line is the baseline, the red is distillation, the yellow is continuous data transfer and the green is data transfer for pre training.

Table 1: The level of success (see text) reached by the various methods.

SOURCE ⇒ TARGET	DATA TRANSFER PRETRAINING	CONTINIOUS DATA TRANSFER	DISTILLATION
BREAKOUT ⇒ PONG	*,2	-	-
PONG ⇒ BREAKOUT	*,2	2	-
TENNIS ⇒ PONG	3	-	-
TENNIS ⇒ BREAKOUT	-	-	-
BREAKOUT ⇒ TENNIS	1	1	1
PONG ⇒ TENNIS	1	1	1
ASSAULT ⇒ DEMON-ATTACK	1 OR 2	-	-
DEMON-ATTACK ⇒ ASSAULT	2	-	2

leads to very different strategies and makes the Pong strategy much less relevant to the Tennis game. For example, a valid strategy in the Pong game is to wait until the last possible second and only than move to hit the ball with the side of the paddle. This strategy is impossible to apply in the Tennis game since the Pong agent will start moving too late in the Tennis game and the player will not be able to hit the ball on time.

To overcome this problem and check how much it affects the final results, we designed a set of experiments where we used different relative speed for the pretrained Pong game in order to over come the speed difference and make the pong paddle speed closer to the tennis player speed. We created those games by tranforming the agent π_s and inserting a NONE action after every few real actions played by the agent.

$$\pi_{tranform}(s) = \begin{cases} \pi_s(s) \text{ If the index of the frame is divided by the ¡slowing factor¿} \\ NONE \text{ Otherwise} \end{cases}$$

This results in slowing down the relative speed between the paddle and the ball by the speed factor. Fig. 5 shows the results of the training of Tennis with the different Pong games variations for pretraining. As can be seen, changing the relative speeds before applying transfer using the data transfer by pretrainig did not improve the Tennis training time. This is probably due to the fact that there are many more factors that affects the transformation validity.

For some pairs of games the reward function is inherently different, which makes the agent optimize on the wrong objective. Such a problem arises in the pairs of Pong and Breakout and Tennis and Breakout, where there is a fundamental difference in the reward function. While in Breakout the objective is to break the most bricks before the player loses, in Pong and Tennis the objective is to make the other player lose. While both reward functions require the paddle of the player to hit the
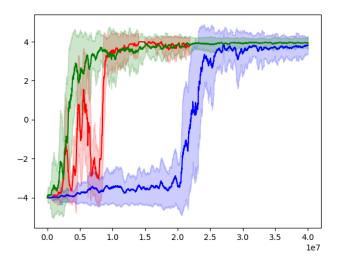
Figure 5: A comparison of the training logs for the game Tennis pretrained with different speeds of Pong, Blue is the baseline (no transfer) green is Tennis pretrained with regular speed pong, and red is Tennis pretrained with slow pong (closer speed to tennis speed).

ball, they are very different in the way they want the ball to behave after being hit and they oppose each other in the amount of time they aim for the game to last – in Breakout the more time you play the better the score is (since you have a better chance at breaking more bricks), while in Pong and Tennis the more time you play, the higher your chance of losing since every time you have to hit the ball is a time where your opponent succeeded in hitting the ball.

In order to make the pairs more similar we designed a different version of Pong where the reward function is the number of times the player hit the ball successfully.

$$R_{new\ pong} = \begin{cases} 1 \text{ If the ball crosses the middle from bottom to top} \\ 0 \text{ Otherwise} \end{cases}$$

Fig. 6 shows the difference in training time of games trained with the Data transfer for pretraining TL method where the pretraining on the source is done using our improved reward or the original reward. It can be seen from the graph that changing the reward function does improve the start of the training and reaches the asymptotic level of performance considerably faster than both transferring from the original game and the baseline. However, the asymptotic success is not significantly improved.

A more general view of the failure to transfer would conclude that the policy π_s overfits the source game and is not able to generalize (RL methods are known to be prone to overfitting). Instead of trying to overcome this by making the source game more similar to the target game, we next try to train a model to fit two source games. Namely, we train a Pong agent in the following way: first we trained a combined model for transferred frames from Tennis to pong and transferred frames from Breakout to Pong. Once the model performed well on both the transfered games, we trained this model on the original Pong frames. Fig. 7 shows that this leads to reaching the asymptotic performance faster than pretraining with any of the two games separately.
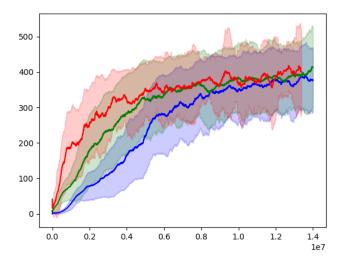
Figure 6: A comparison of the training logs for the game Breakout pretrained with different reward function for Pong, Blue is the baseline green is regular reward, and red is our improved reward.
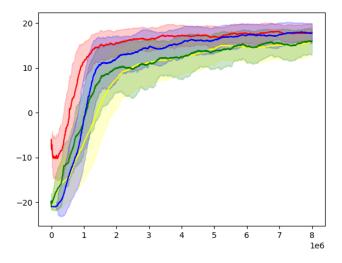

Figure 7: A comparison of the training logs for the game Pong pretrained with breakout or breakout and pong. Blue is the baseline, green is Pong pretrained with breakout, yellow is Pong pretrained with Tennis and red is Pong pretrained with breakout and Tennis.

REFERENCES

Sagie Benaim and Lior Wolf. One-sided unsupervised domain mapping. In *Advances in Neural Information Processing Systems*, pp. 752–762, 2017.

Juan C. Caicedo and Svetlana Lazebnik. Active object localization with deep reinforcement learning. In *ICCV*, pp. 2488–2496, 2015.

Shani Gamrian and Yoav Goldberg. Transfer learning for related reinforcement learning tasks via image-to-image translation. *arXiv preprint arXiv:1806.07377*, 2018.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, pp. 5769–5779, 2017.

Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. *arXiv preprint arXiv:1710.02298*, 2017.

Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jungkwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. *arXiv preprint arXiv:1703.05192*, 2017.

Cody Kwok and Dieter Fox. Reinforcement learning for sensing strategies. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 4, pp. 3158–3163. IEEE.

Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems*, pp. 700–708, 2017.

Yongyi Lu, Yu-Wing Tai, and Chi-Keung Tang. Conditional cyclegan for attribute guided face image generation. *arXiv preprint arXiv:1705.09966*, 2017.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pp. 1928–1937, 2016.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI*, volume 16, pp. 2094–2100, 2016.