Visual Transfer for Reinforcement Learning via Wasserstein Domain Confusion

Josh Roy and George Konidaris
Brown University
joshnroy@gmail.com, gdk@cs.brown.edu

Abstract

We introduce Wasserstein Adversarial Proximal Policy Optimization (WAPPO), a novel algorithm for visual transfer in Reinforcement Learning that explicitly learns to align the distributions of extracted features between a source and target task. WAPPO approximates and minimizes the Wasserstein-1 distance between the distributions of features from source and target domains via a novel Wasserstein Confusion objective. WAPPO outperforms the prior state-of-the-art in visual transfer and successfully transfers policies across Visual Cartpole and both the easy and hard settings of of 16 OpenAI Procgen environments.

Introduction

Deep Reinforcement Learning (RL) has enabled agents to autonomously solve difficult, long horizon problems such as Atari games from pixel inputs (Sutton and Barto 2018; Mnih et al. 2015). However, in high risk domains such as robotics and autonomous vehicles, RL agents lack the ability to learn from their mistakes without destroying equipment or harming humans. Instead, it would be preferable to train in simulated domains and transfer to the real world. However, Deep model-free Reinforcement Learning agents trained on one environment often fail on environments that require solving the same underlying problem but different visual input (Cobbe et al. 2019b,a). When transferring from a simulated source domain to a real-world target domain, this is known as the reality gap (Sadeghi et al. 2018; Tobin et al. 2017; Sadeghi and Levine 2016). This challenge is known as visual generalization in RL and is studied by transferring between simulated domains (Cobbe et al. 2019b,a; Zhang et al. 2018; Justesen et al. 2018; Juliani et al. 2019).

The main reason for such difficulty in generalization is the tendency of deep networks to overfit to a single task (Zhang et al. 2018; Justesen et al. 2018; Cobbe et al. 2019b; Dubey et al. 2018). For example, an agent that learns to balance the cartpole depicted on the left of Figure 1 will fail to generalize to the cartpole on the right due to visual differences alone. In deep supervised learning, this can be addressed by smoothing the learned function using methods such as data augmentation (Simard et al. 2003), dropout (Srivastava et al. 2014), and regularization (Ng 2004). However, these methods are insufficient for generalization in Deep RL (Cobbe et al. 2019b). Unlike supervised learning, where all datapoints are drawn from the same distribution, transferring between differing RL tasks is akin to that of supervised domain adaptation where datapoints from source and target tasks are drawn from different distributions (Tzeng et al. 2017; Ganin et al. 2016; Tzeng et al. 2015).

Previous work such as Domain Randomization or Meta-Learning takes a brute force approach to this problem, training on many source domains before fine tuning on the target domain for 0 or k episodes, referred to as zero-shot or k-shot transfer, respectively. These methods require a large number of source domains before they can transfer or adapt to the target domain. Domain Randomization learns a representation sufficient to solve all source domains, implicitly aligning the distributions of features extracted from each domain. It then assumes the features extracted from the target domain will fall into a similar distribution (Tobin et al. 2017; Andrychowicz et al. 2020; Akkaya et al. 2019). Some recent work has attempted to more explicitly align the feature distributions across domains but assumes access to pairwise correlations of states across domains (Slaoui et al. 2019; Gupta et al. 2017a; Tzeng et al. 2020). Other work relies upon style-transfer networks to "translate" target domain states to source domain states before processing (Zhang et al. 2019). This adds additional computational complexity during both training and inference and relies upon the quality of the translation network. Further work attempts to learn the causal structure underlying such a family of visual Markov Decision Processes (MDPs), leading to a causal representation independent of domain and successful transfer (Zhang et al. 2020). Both translation and causal methods require modeling of variables unrelated to the task given, leading to additional complexity (Zhang et al. 2019, 2020).

We introduce a novel algorithm that explicitly learns to align distributions of extracted features between a source and target task without adding additional computation during inference or assuming access to pairwise correlations of observations. By simultaneously training an RL agent to solve a source task and minimize the distance between distributions of extracted features from the source and target domains, our algorithm enables seamless transfer to the target domain. Specifically, we train an adversary network to

approximate the Wasserstein-1 distance (Arjovsky, Chintala, and Bottou 2017) between the distributions of features from source and target tasks, and an RL agent to minimize this distance via a novel Wasserstein Confusion objective while solving the source task. The Wasserstein-1 distance between distributions can be intuitively described as the effort required to align two probability distributions by transporting their mass (Rubner, Tomasi, and Guibas 2000). As shown in Arjovsky, Chintala, and Bottou (2017), minimizing this distance allows adversarial alignment methods to succeed where other distance metrics, such as Jensen-Shannon divergence, fail. Our algorithm outperforms the prior state-of-the-art in visual transfer and successfully transfers policies across Visual Cartpole, a visual variant of the standard cartpole task where colors are changed across domains (Brockman et al. 2016), and both the easy and hard settings of 16 OpenAI Procgen environments (Cobbe et al. 2019a).

Background and Related Work

RL is concerned with sequential decision making (Sutton and Barto 2018): an agent exists within a world (environment) and must take an action a based on some information about the world (state) s. This causes the environment to provide the agent with the next state s' and a reward r. The agent's goal is to learn a policy π mapping states to actions that maximizes the expected sum of rewards $\mathbb{E}[\sum_t \gamma^t r_t]$, where $\gamma \in [0, 1)$ is a discount factor that weights the importance of short and long term rewards. Return is defined as the sum of cumulative rewards. The interactions between the agent and the environment are modeled as a Markov Decision Process (MDP) defined as a 5-tuple(S, A, T, R, γ) where S is the set of states, A is the set of actions, T is a function from state and action to next state, R is a function from state and action to reward, and γ is the discount factor.

Model-free Deep RL uses neural networks to predict both the value (expected future reward) of a state and the optimal action to take. Proximal Policy Optimization (PPO) is a state-of-the-art model-free policy gradient algorithm for Deep RL (Schulman et al. 2017). It parameterizes a policy $\pi_\theta(a|s)$ and a value function $V_\theta(s)$ that share the majority of their weights, splitting after the "feature extraction" section of the network. The value network is trained to minimize the mean square error $\mathcal{L}_{\text{value}} = \frac{1}{n}\sum_{i=1}^{n}(V(s) - R)^2$, where R is the return. The policy network is trained to minimize $\mathcal{L}_{\text{policy}} = -\hat{\mathbb{E}}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t]$, where $\hat{\mathbb{E}}$ is the empirical expected value at timestep t, $\hat{A}_t$ is the empirical advantage at timestep t, $r(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the ratio of taking action a_t given state s_t between the current and previous policies, and ϵ is a small hyperparameter. The two function approximators are trained jointly, and their combined loss is $\mathcal{L}_{\text{PPO}} = \mathcal{L}_{\text{policy}} + \mathcal{L}_{\text{value}}$.

Transfer in Reinforcement Learning

Transfer in RL has been a topic of interest far before the recent advent of Deep Neural Networks and Deep RL. Work on transfer is separated into questions of dynamics, representation, and goals with environments differing in their states, actions, transition function, or reward function, respectively (Taylor and Stone 2009; Lazaric 2012). A policy that transfers perfectly trains on a source MDP and achieves target reward equivalent to an agent trained on a target MDP.

The most popular approach in k-shot transfer for RL is Meta RL. These methods optimize a Deep RL agent's parameters such that it can rapidly learn any specific task selected from a set of tasks. Specifically, the agent first trains on n source tasks and then trains on the $n+1$th task for k episodes before measuring performance on the $n+1$th task (Finn, Abbeel, and Levine 2017; Nichol and Schulman 2018). There are also a number of other approaches (Carr, Chli, and Vogiatzis 2018; Gupta et al. 2017b; Wulfmeier, Posner, and Abbeel 2017) that transfer between domains by utilizing the ability to train in the target domain. Though they can transfer from one source to one target domain, training or fine tuning in the target domain is a very strong assumption. For example, training on a real robot arm will likely result in unstable behavior and possibly damage, but collecting image observations will be safe and predictable.

Domain Randomization is the most popular approach to zero-shot transfer in RL. In Domain Randomization, an agent trains on a set of n source tasks, implicitly learning a representation and policy sufficient to zero-shot transfer to the $n+1$th task (Tobin et al. 2017; Andrychowicz et al. 2020; Akkaya et al. 2019). For successful transfer, all $n+1$ tasks must be drawn from the same distribution. Furthermore, n is required to be sufficiently large that the agent is unable to memorize domain-specific policies.

Some work directly addresses dynamics transfer where the transition function changes, but states, actions, and rewards are identical. Some work (Killian et al. 2017; Doshi-Velez and Konidaris 2016; Yao et al. 2018) parameterizes the transition function of an MDP and learning a conditional policy that can transfer between such tasks. Other work generates a curriculum to learn generalizable policies that can adapt to MDPs with differing dynamics (Mysore, Platt, and Saenko 2019). Some approaches (Wulfmeier, Posner, and Abbeel 2017; Joshi and Chowdhary 2019) uses distributional alignment. This work solves a different problem than WAPPO which focuses on appearance-based transfer.

Visual Transfer in Reinforcement Learning

Visual transfer takes place within a family $\mathcal{M}$ of related Block MDPs $M \in \mathcal{M}$ each defined by a 6-tuple $(S, \mathcal{A}, \mathcal{X}, p, q, R)$ where S is an unobserved state space, $\mathcal{A}$ is an action space, $\mathcal{X}$ is an observation space, $p(s'|s, a)$ is a transition distribution over the next state s' based on the previous state s and action a, $q(x|s)$ is an emission function that represents the probability of an observation x based on a hidden state s, and $R(s, a)$ is a reward function that maps from a state and action to a reward (Du et al. 2019; Zhang et al. 2020). The emission function q and the observation space $\mathcal{X}$ are the only quantities that change across Block MDPs within a family. Block MDPs are a subset of POMDPs which have emission functions rather than observations functions (Du et al. 2019; Monahan 1982). Both functions map from hidden states to observations but emission functions generate observations that are definitionally

Markov and uniquely identify their corresponding underlying states (Du et al. 2019). We make the decision to use Block MDPs and focus on visual transfer just as dynamics transfer approaches focus on solving transfer between MDPs with differing transition functions but identical states, actions, and rewards. Furthermore, the Block MDP formulation extends to real-world problems such as simulation to reality transfer for robotic reinforcement learning. Modern robotic simulators have accurate physical modeling of robot arms, turtlebots, and other robots, but not photorealistic graphics. To transfer from simulation to reality, an agent must transfer between MDPs with identical underlying states (ex: gripper and object location), actions (ex: motor position setpoints), transitions, and rewards but different observations. This exactly fits the Block MDP definition.

To transfer across different visual domains, Robust Domain Randomization (RDR) (Slaoui et al. 2019) aims to learn a domain-agnostic representation by training an agent to solve n source domains while minimizing the Euclidian distance between internal representations across domains. It then attempts to zero-shot transfer the $n + 1$th domain. This method shows success in tasks where the background color is the only varied property, but the visual randomization of complex tasks such as OpenAI Procgen (Cobbe et al. 2019a) is far higher. When the training data contains pairs of samples with the same semantic content in each domain, minimizing the Euclidian distance will align the distributions of representations for each domain. Without paired data, this will incorrectly align samples with different hidden states, leading to unreliable representations.

Other approaches assume either the labeling or existance of pairs of images from source and target domains and minimize paired distance, aligning the feature distributions across domains (Gupta et al. 2017a; Tzeng et al. 2020). The probability of collecting pairs of source and target domain samples where underlying states are identical decreases rapidly with state space complexity. Obtaining such data is often impractical or infeasible in practice. Though this work attempts to solve visual transfer for RL, it makes restrictive assumptions that limit its scope.

Other work uses causal inference to learn Model-Irrelevance State Abstractions (MISA) (Zhang et al. 2020) and transfer between Block MDPs. MISA can transfer between RL tasks with low-dimensional states and visual imitation learning tasks with changing background color (Zhang et al. 2020) but not visual RL tasks. Since MISA learns to reconstruct observations, it must model factors irrelevant to the task. Additionally, MISA minimizes mean squared error between observations and reconstructions, which ignores small objects due to their low effect on the error signal (Oh et al. 2015). In visual RL, small objects such as the player character are essential to solving the task.

Work in supervised domain adaptation (Hoffman et al. 2018) and style transfer (Zhu et al. 2017; Jing et al. 2019; Gatys, Ecker, and Bethge 2015) translates images to different "styles" such as those of famous painters by training Generative Adversarial Networks (GANs) to map from an input image to a semantically equivalent but stylistically different output image. VR Goggles for Robots (Zhang et al. 2019) aims to transfer RL policies across visually differing domains by using a style transfer network to translate target domain images into the source domain. This enables an RL agent trained in the source domain to act in the target domain but adds additional computational and algorithmic complexity during inference time and relies heavily upon the success of the style-transfer network (Zhang et al. 2019) as the policy does not train on translated images. As the style transfer network minimizes the mean absolute error and mean squared error between real and generated images, it prioritizes modeling factors that utilize a higher number of pixels. In complex visual RL tasks, important factors, such as the player character, occupy a small number of pixels and recieve low priority (Oh et al. 2015). While style-transfer adds computational complexity and relies upon image reconstruction, other work in GANs and supervised domain transfer aligns distributions without these drawbacks.

Adversarial Distribution Alignment

Prior work in GANs (Goodfellow et al. 2014) shows that adversarial methods can align arbitrary distributions of data. Given samples drawn from a "real" distribution P_r a GAN learns a mapping from noise sampled from a Gaussian distribution P_z to samples from a "fake" distribution P_f. It jointly trains an adversary network to classify samples as real or fake and a generator network to "fool" the adversary, minimizing the Jensen-Shannon divergence (JS divergence) between the distributions (Goodfellow et al. 2014).

Some domain adaptation methods in supervised learning uses a GAN-like adversary to align a classifier's internal representations across domains and solve the supervised classification analogue of visual transfer in RL (Tzeng et al. 2017; Ganin et al. 2016; Tzeng et al. 2015). They each introduce and minimize different distribution alignment objectives based on minimizing the classification accuracy of the adversary network (Tzeng et al. 2015, 2017).

GAN-like adversarial algorithms align distributions but are unstable and prone to failures such as mode collapse (Arjovsky, Chintala, and Bottou 2017). Furthermore, minimizing the JS divergence between two distributions has been shown to fail in certain cases, such as aligning two uniform distributions on vertical lines (Arjovsky, Chintala, and Bottou 2017). Wasserstein GANs (Arjovsky, Chintala, and Bottou 2017) solve both of these problems by changing the adversarial classifier into an adversarial critic f that estimates the Wasserstein-1 distance $W(P_r, P_f)$ between real and fake distributions P_r and P_f. Minimizing Wasserstein-1 distance is more stable and empirically shown to avoid mode collapse (Arjovsky, Chintala, and Bottou 2017). Though directly estimating the Wasserstein-1 distance is infeasible, the Kantorovich-Rubinstein duality (Villani 2008) provides a reparameterization that is directly computable. The gradient of the generator G with weights θ is defined as $\nabla_\theta W(P_r, P_f) = -E_{z\sim P_z}[\nabla_\theta f(G(z))]$.

Wasserstein Adversarial Proximal Policy Optimization

Our novel algorithm Wasserstein Adversarial Proximal Policy Optimization (WAPPO) transfers from any source task M_s to any target task M_t where both M_t, M_s are Block MDPs drawn from a family $\mathcal{M}$. For any two such tasks, the observation spaces $\mathcal{X}_s, \mathcal{X}_t$ and emission functions q_s, q_t are different, but the hidden state space S, action space $\mathcal{A}$, transition function p, and reward function R are identical. The Visual Cartpole environments shown in Figure 1 illustrate difference in emission function; the source domain has a green cart and pole as on a pink background and blue track while the target domain has a brown cart with a green pole on a green background and yellow track. However, both environments are defined by the same hidden states (position of the cart and pole), actions (push the cart left or right), rewards ($+1$ for keeping the pole upright at each timestep), and transitions (simulation of physics). Furthermore, the optimal policy that balances the pole is solely a function of the hidden states representing the position of the cart and pole and not of the colors of the objects.

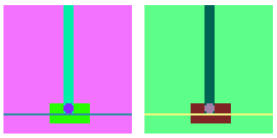


Figure 1: Example Visual Cartpole Source Domain (Left) and Target Domain (Right). The two domains contain the same hidden states, actions, rewards, and transitions but differ in their emission functions and observations.

In model-free Deep RL, the agent does not learn to approximate the transition and reward functions of its MDP. Instead, it learns a policy that maximizes its reward and representation function h_θ parameterized by the first few layers of the RL network that maps from observations $x \in \mathcal{X}$ to internal representations $\mathfrak{r} \in \mathfrak{R}$. In the Block MDPs defined above, an optimal policy depends solely on the hidden state space S. Thus, an optimal representation function h^* will map observations $x \in \mathcal{X}$ to internal representations $\mathfrak{r} \in \mathfrak{R}$ that contain no more information than their corresponding hidden states.

Similarly, there exist optimal representation functions h^*_s, h^*_t for source and target tasks $M_s, M_t \in \mathcal{M}$. Since the hidden state spaces of M_s, M_t are identical, the optimal policies and representation spaces $\mathfrak{R}_s, \mathfrak{R}_t$ are identical. Thus, there exists a representation function $h^*_{(s,t)}$ that maps from both $\mathcal{X}_s$ and $\mathcal{X}_m$ to a representation space $\mathfrak{R} = \mathfrak{R}_s = \mathfrak{R}_t$ that is sufficient to solve both source and target tasks.

The goal of Wasserstein Adversarial PPO is to learn a representation function $h_{\theta,(s,t)}$ that approximates the optimal representation function $h^*_{(s,t)}$. It is given the ability to train within M_s and access a buffer of observations sampled from M_t. Note that the observations from M_t are not paired with observations from M_s and are not sequential. Further, WAPPO does not have access to rewards from M_t. Given that Block MDPs M_s, M_t both belong to the family $\mathcal{M}$, they share hidden states, transition function, and reward function but differ in observations. Therefore, an agent with a domain-agnostic representation will be able to learn a policy in M_s and transition seamlessly to M_t. We define the representation function h_θ as the first few layers of the RL network with parameters θ. To learn a domain-agnostic representation, Wasserstein Adversarial PPO jointly learns to solve M_s while using an adversarial approach to align the distributions of representations from M_s and M_t.

Specifically, Wasserstein Adversarial PPO trains on a dataset D defined as $\{(x_s, x_t, a_s, r_s)_1, ...(x_s, x_t, a_s, r_s)_m\}$ where m is the length of the dataset, $x_s \in \mathcal{X}_s$ is an observation from the source MDP, $x_t \in \mathcal{X}_t$ is an observation from the target MDP, a_s is an action from the source MDP, and r_s is a reward from the source MDP. Note that x_s, a_s, r_s correspond to the same timestep from the source MDP: the agent takes action a_s from state x_s, receives reward r_t, and transitions to the next state. x_t does not correspond to the same timestep as the other variables. Instead, it is drawn randomly from a buffer of observations from the target MDP. WAPPO uses the output of an intermediate layer, denoted by $h_\theta(x)$ as a latent representation $\mathfrak{r}$ of the observation x. To enforce that this latent representation x is agnostic to domain, WAPPO approximates h^* by minimizing the Wasserstein-1 distance between the distributions of $h_\theta(x_s)$ and $h_\theta(x_t)$.

Similar to supervised adversarial domain adaptation algorithms (Tzeng et al. 2017; Ganin et al. 2016; Tzeng et al. 2015), WAPPO consists of two networks: the RL network and the adversary network shown in the Appendix. The RL network learns a latent representation which is used to compute the best next action and the value of each observation. This latent representation should be identical across source and target domain. The adversary network takes the latent representation as input and is trained to distinguish between source and target tasks. The policy network both maximizes performance on the source task and minimizes the adversary's ability to identify the domain. Specifically, the RL network minimizes

$$\mathcal{L}_{\text{WAPPO}} = \mathcal{L}_{\text{PPO}} + \lambda \mathcal{L}_{\text{Conf}},$$

where $\mathcal{L}_{\text{PPO}}$ is the PPO loss, $\mathcal{L}_{\text{Conf}}$ is the loss term that maximally confuses the critic and aligns the distributions of source and target observations, and λ is a constant weight.

The Wasserstein GAN (Arjovsky, Chintala, and Bottou 2017) approximates the Wasserstein distance between real and fake distributions using a neural network. Additionally, it defines a derivative for this distance that is used in optimizing the loss term for the generator, demonstrating higher stability than the standard GAN loss.

While the Wasserstein GAN loss term seems to align exactly with $\mathcal{L}_{\text{Conf}}$, it has one key difference. It assumes that one distribution is fixed, which is not true of domain adaptation. The goal of domain adaptation is to align two distributions both parameterized by θ. Specifically, we wish to align the distribution of extracted features from the source domain $P_s^{h_\theta}$ with the extracted features of the target domain $P_t^{h_\theta}$. Note that it is not possible to directly sample from $P_s^{h_\theta}, P_t^{h_\theta}$. Instead, we sample from these distributions by first sampling from the distribution of observations, P_s, P_t and then map-

ping to the representation by applying h_θ. Thus, the Wasserstein distance is defined as

$$W(P_s, P_t) = E_{x\sim P_s}[f(h_\theta(x))] - E_{x\sim P_t}[f(h_\theta(x))], \quad (1)$$

where f is the adversarial critic and the gradient is defined as:

$$\begin{aligned}\nabla_\theta W(P_s, P_{f_t}) &= \nabla_\theta[E_{x\sim P_s}[f(h_\theta(x))] - E_{x\sim P_t}[f(h_\theta(x))]] \\ &= \nabla_\theta E_{x\sim P_s}[f(h_\theta(x))] - \nabla_\theta E_{x\sim P_t}[f(h_\theta(x))] \\ \nabla_\theta W(P_s, P_t) &= E_{x\sim P_s}[\nabla_\theta f(h_\theta(x))] - E_{x\sim P_t}[\nabla_\theta f(h_\theta(x))]. \end{aligned} \quad (2)$$

Moving the gradient inside the expectation is shown to be correct in the proof of Theorem 3 of Arjovsky, Chintala, and Bottou (2017).

Experimental Results

We validate our novel Wasserstein Confusion loss term and WAPPO algorithm on 17 environments: Visual Cartpole and both the easy and hard versions of 16 OpenAI Procgen environments. To evaluate the ability of the Wasserstein Confusion loss term to align distributions of features across environment and enable successful transfer, we examine how an RL agent trained using WAPPO on a source domain performs on a target domain. For each environment evaluated, the agent trains using WAPPO with full access to the source domain and a buffer of 5000 observations from the target domain. The agent does not have access to rewards from the target domain. We compare WAPPO's transfer performance with that of three baselines: PPO, PPO with the feature matching loss as described by Robust Domain Randomization, the prior state of the art for feature alignment in RL (Slaoui et al. 2019), and PPO with VR Googles, the prior state of the art for domain adaptation in RL (Zhang et al. 2019).

Robust Domain Randomization originally trains on n source domains while matching their features before attempting to transfer zero-shot to the target domain. It's main contribution is a feature alignment loss term. By minimizing this term and aligning the distributions of features extracted from the n source domains, it hopes that the distribution of features from the target domain will also be aligned. We directly evaluate RDR's ability to match distributions of features by training an RL agent on one source domain and evaluating on one target domain while minimizing it's feature alignment loss using observations from each domain. Like zero-shot setting, the agent's performance on the target domain is proportional to the alignment of features from source and target domains. This enables a direct comparison between the feature alignment loss used in RDR and our feature alignment loss, Wasserstein Confusion.

VR Goggles trains an RL agent on the source domain and a style-transfer network between target and source domain (Zhang et al. 2019). During evaluation on the target domain, it translates images to the style of the source domain before applying the RL agent's trained policy. As VR Goggles utilizes a pre-trained source agent rather than a new RL algorithm, we report the target performance on Figures 2a, 3, and 4 as a horizontal line. We use the baseline PPO agent as the pre-trained source agent.

We utilize the PPO implementation and hyperparameters provided with (Cobbe et al. 2019a). We use these same hyperparameters for the other methods tested and do not perform any hyperparameter searches. Switching ReLU activations to Leaky ReLU activations to ease adversarial training is the only change made to the code. For further details, please see the appendix and the code [1] distributed with Cobbe et al. (2019a). Our adversarial critic is 9 dense layers of width 512 separated by Leaky ReLU functions.

Each experiment's performance is reported across multiple trials with identical random seeds across algorithms. Visual Cartpole and Procgen Easy are evaluated across 5 trials and Procgen Hard is evaluated across 3 trials. There is one source domain and one target domain per trial.

Visual Cartpole

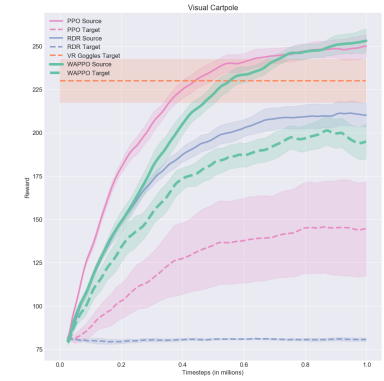
We first demonstrate transfer on a simple environment, Visual Cartpole, a variant on the standard Cartpole task (Brockman et al. 2016) where observations are images of the Cartpole instead of position and velocity. Color of the cart, pole, background, and track are varied across domains, converting Cartpole into a family of Block MDPs where different emission functions correspond to different colors. Figure 1 shows sample source and target environments.

As shown in Figure 2a, Wasserstein Adversarial PPO far outperforms both PPO and PPO using RDR's feature matching loss. As shown in Figure 2b, the target features extracted by both PPO and RDR lie within a very small subset of their corresponding source features. This dense clustering implies that the target features are not expressive compared to the source features, leading to low transfer performance. When using Wasserstein Adversarial PPO, the target features are clustered at similar density with the source features, demonstrating their distributional alignment and leading to higher reward. The density plots in Figure 2b are generated by reducing the features space to 2 dimensions via Principle Component Analysis (PCA) (Wold, Esbensen, and Geladi 1987) and estimating a density via Gaussian kernel density estimation (Scott 2015). While VR Goggles outperforms WAPPO and the other algorithms in Visual Cartpole, it does not in complex visual environments such as Procgen (Cobbe et al. 2019a). VR Goggles is able to perform well in Visual Cartpole as the background color is the only visual distraction and the cartpole system occupies a majority of the visual observation. We do not visualize distributions of features for VR Goggles as it operates on the observation space rather than the image space.

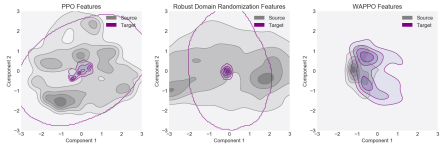
OpenAI Procgen

The remaining 16 environments are from OpenAI's Procgen Benchmark (Cobbe et al. 2019a). This benchmark originally provides agents n source domains to train on and tests their generalization to a different target domain. The domains differ according to their emission functions, dynamics, and states. As this work focuses on visual domain adaptation in Reinforcement Learning, we modify the environments such that each environment is a family of Block

[1] https://github.com/openai/procgen



(a) Training Graph.



(b) Feature Distributions.

Figure 2: Visual Cartpole Training Graph (a) and Feature Distributions (b): Solid and dashed lines indicate source and target reward, respectively. Green indicates WAPPO, pink indicates PPO, blue indicates RDR, and orange indicates VR Goggles. Shading shows standard deviation across 5 trials. Gray and purple indicate source and target features, respectively. WAPPO has both higher target reward and better alignment of source and target features than other methods.

MDPs. All domains in a particular environment have identical state spaces, transition functions, and reward functions but unique emission functions. This decouples the different types of transfer and solely focuses on evaluating visual transfer. To modify the Procgen environments, we set a fixed emission function for each domain that maps from underlying state to observation. This function decides which sprites, colors, and visual factors to use for each domain. It is not responsible for underlying state factors such as maze shape, platform location, or enemy positions. Unlike the Procgen environments, resseting the environment does not provide the agent with a new domain. Instead it resets the domain, keeps the emission function, and changes underlying states. In an environment such as `maze`, an agent trained on a single domain will see different maze shapes, starting locations, and goal locations, but a single configuration of visual factors including colorscheme and character sprites. This modification both fits the Block MDP formulation and allows us to evaluate visual transfer ability without interference from dynamics, reward, or underlying state transfer.

The training curves and results are not expected to match that of the Procgen benchmark, as they are two separate tasks. In the Procgen benchmark, agents are trained on 200 levels where dynamics, visual factors, state factors, and rewards differ and are tested on held-out levels. In this task, agents are trained on one source domain with access to a fixed number of observations from one target domain, where only visual factors differ, as formalized by Block MDPs.

As in Visual Cartpole, we train the agent in one source domain with access to observations from the target domain and test the agent's performance on the target domain. The environments vary in difficulty of both the source task and generalizing to a new domain. Chaser, Heist, Maze, and Miner focus on navigating in 2-dimensional mazes of varying size. This is a difficult task for model-free deep RL agents as they cannot plan a path betwen maze locations and causes difficulty for all of the agents evaluated (Tamar et al. 2016).

There are two different instantiations of each environment: an easy version and a hard version. The easy version of each game has less variation across all factors, such as colors, layouts, and goal locations (Cobbe et al. 2019a). In certain games, the easy version provides hints that simplify the task, such as an arrow pointing towards the goal.

We report both the reward on each task and the normalized return across all tasks. Normalized return is calculated by averaging the normalized return on each task $R_{\text{norm}} = (R - R_{\text{min}})/(R_{\text{max}} - R_{\text{min}})$, where R is the vector of rewards and $R_{\text{min}}, R_{\text{max}}$ are the minimum and maximum returns for each environment. This quantity measures the overall source and target performance for each algorithm.

As shown in Figure 3, WAPPO's target performance exceeds that of the other algorithms in 14 of the 16 easy environments. In Coinrun and Chaser, the target performance of all algorithms approximately matches their training performance, demonstrating the environments' lack of transfer difficulty. In Chaser, WAPPO's performance exceeds that of RDR and matches that of PPO. In Coinrun, WAPPO matches the performance of RDR and exceeds PPO.

These results are mirrored in the hard versions of the Procgen environments, as shown in Figure 4. WAPPO's target performance exceeds that of the other algorithms in 12 of the 16 hard environments. On Plunder, Chaser, and Leaper, WAPPO matches the performance of PPO and outperforms other algorithms. Chaser has minimal visual variation, allowing all algorithms match their source and target performance. On Maze, WAPPO performs worse than PPO but better than other visual transfer algorithms. Maze tests an agent's ability to solve mazes and has few visual differences across domains. Maze solving is difficult for model-free Deep RL agents due to their lack of ability to plan a path between locations (Tamar et al. 2016). As shown in Figures 3 and 4, WAPPO's normalized return on easy and hard Procgen environments far exceeds that of prior algorithms, demonstrating its superior ability to generalize to a target domain. Aligning source and target distributions allows WAPPO to ignore distracting visual details in the source domain and achieve higher source domain reward.

Conclusion

We address the difficulty in transferring a learned policy across differing visual environments by explicitly learning a representation sufficient to solve both source and target domains. Previous approaches have added additional inference-time complexity (Zhang et al. 2019), relied upon pairwise observations (Gupta et al. 2017a; Tzeng et al.

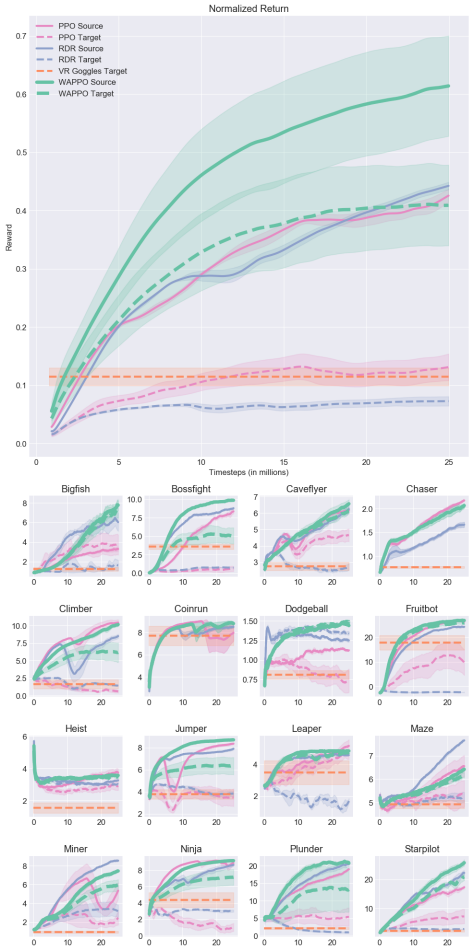


Figure 3: Procgen (Easy) Training Graph. Solid lines and dashed lines indicate source and target reward, respectively. Green indicates WAPPO, blue indicates PPO, pink indicates RDR, and orange indicates VR Goggles. WAPPO matches or outperforms the other algorithms across all environments.

2020), or ignored fine details by relying upon image reconstruction losses (Zhang et al. 2019, 2020). We have introduced a new method, WAPPO, that does not. Instead, it uses a novel Wasserstein Confusion objective term to force the RL agent to learn a mapping from visually distinct domains to domain-agnostic representations, enabling state-of-the-art domain adaptation performance in Reinforcement Learning. We validate WAPPO on 17 visual transfer environments where our agent both achieves higher reward in the target MDP and better matches distributions of representations across domains.

Acknowledgements

This work was supported by the National Science Foundation under grant number IIS-1717569 and DARPA under grant number W911NF1820268.

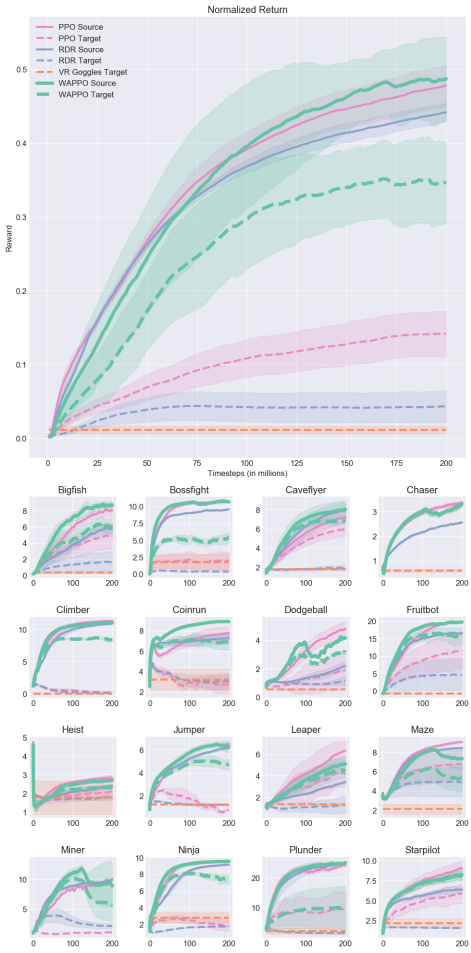


Figure 4: Procgen (Hard) Training Graph. Solid lines and dashed lines indicate source and target reward, respectively. Green indicates WAPPO, blue indicates PPO, pink indicates RDR, and orange indicates VR Goggles. WAPPO matches or outperforms the other algorithms across all environments except Maze which primarily tests path planning ability rather than visual transfer.

References

Akkaya, I.; Andrychowicz, M.; Chociej, M.; Litwin, M.; McGrew, B.; Petron, A.; Paino, A.; Plappert, M.; Powell, G.; Ribas, R.; et al. 2019. Solving Rubik's Cube with a Robot Hand. *arXiv preprint arXiv:1910.07113* .

Andrychowicz, O. M.; Baker, B.; Chociej, M.; Jozefowicz, R.; McGrew, B.; Pachocki, J.; Petron, A.; Plappert, M.; Powell, G.; Ray, A.; et al. 2020. Learning Dexterous In-Hand Manipulation. *International Journal of Robotics Research* 39(1): 3–20.

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein Generative Adversarial Networks. In *Proceedings of the International Conference on Machine Learning*, 214–223.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. OpenAI Gym. *arXiv preprint arXiv:1606.01540* .

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

Cobbe, K.; Hesse, C.; Hilton, J.; and Schulman, J. 2019a. Leveraging Procedural Generation to Benchmark Reinforcement Learning. *arXiv preprint arXiv:1912.01588* .

Cobbe, K.; Klimov, O.; Hesse, C.; Kim, T.; and Schulman, J. 2019b. Quantifying Generalization in Reinforcement Learning. In *Proceedings of the International Conference on Machine Learning*, 1282–1289.

Dhariwal, P.; Hesse, C.; Klimov, O.; Nichol, A.; Plappert, M.; Radford, A.; Schulman, J.; Sidor, S.; Wu, Y.; and Zhokhov, P. 2017. OpenAI Baselines. https://github.com/openai/baselines.

Doshi-Velez, F.; and Konidaris, G. 2016. Hidden Parameter Markov Decision Processes: A Semiparametric Regression Approach for Discovering Latent Task Parametrizations. In *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 2016, 1432.

Du, S.; Krishnamurthy, A.; Jiang, N.; Agarwal, A.; Dudik, M.; and Langford, J. 2019. Provably Efficient RL with Rich Observations via Latent State Decoding. In *Proceedings of the International Conference on Machine Learning*, 1665–1674.

Dubey, R.; Agrawal, P.; Pathak, D.; Griffiths, T.; and Efros, A. 2018. Investigating Human Priors for Playing Video Games. In *Proceedings of the International Conference on Machine Learning*, 1349–1357.

Espeholt, L.; Soyer, H.; Munos, R.; Simonyan, K.; Mnih, V.; Ward, T.; Doron, Y.; Firoiu, V.; Harley, T.; Dunning, I.; et al. 2018. IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. In *Proceedings of the International Conference on Machine Learning*, 1407–1416.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *Proceedings of the International Conference on Machine Learning*, 1126–1135. JMLR. org.

Ganin, Y.; Ustinova, E.; Ajakan, H.; Germain, P.; Larochelle, H.; Laviolette, F.; Marchand, M.; and Lempitsky, V. 2016. Domain-Adversarial Training of Neural Networks. *Journal of Machine Learning Research* 17(1): 2096–2030.

Gatys, L. A.; Ecker, A. S.; and Bethge, M. 2015. A Neural Algorithm of Artistic Style. *arXiv preprint arXiv:1508.06576* .

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, 2672–2680.

Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; and Courville, A. C. 2017. Improved Training of Wasserstein GANs. In *Advances in neural information processing systems*, 5767–5777.

Gupta, A.; Devin, C.; Liu, Y.; Abbeel, P.; and Levine, S. 2017a. Learning Invariant Feature Spaces to Transfer Skills with Reinforcement Learning. *arXiv preprint arXiv:1703.02949* .

Gupta, A.; Devin, C.; Liu, Y.; Abbeel, P.; and Levine, S. 2017b. Learning invariant feature spaces to transfer skills with reinforcement learning. *arXiv preprint arXiv:1703.02949* .

Hoffman, J.; Tzeng, E.; Park, T.; Zhu, J.-Y.; Isola, P.; Saenko, K.; Efros, A.; and Darrell, T. 2018. CyCADA: Cycle-Consistent Adversarial Domain Adaptation. In *Proceedings of the International Conference on Machine Learning*, 1989–1998.

Jing, Y.; Yang, Y.; Feng, Z.; Ye, J.; Yu, Y.; and Song, M. 2019. Neural Style Transfer: A Review. *IEEE Transactions on Visualization and Computer Graphics* .

Joshi, G.; and Chowdhary, G. 2019. Adapt-to-Learn: Policy Transfer in Reinforcement Learning .

Juliani, A.; Khalifa, A.; Berges, V.-P.; Harper, J.; Teng, E.; Henry, H.; Crespi, A.; Togelius, J.; and Lange, D. 2019. Obstacle Tower: A Generalization Challenge in Vision, Control, and Planning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2684–2691. AAAI Press.

Justesen, N.; Torrado, R. R.; Bontrager, P.; Khalifa, A.; Togelius, J.; and Risi, S. 2018. Illuminating Generalization in Deep Reinforcement Learning through Procedural Level Generation. *arXiv preprint arXiv:1806.10729* .

Killian, T. W.; Daulton, S.; Konidaris, G.; and Doshi-Velez, F. 2017. Robust and Efficient Transfer Learning with Hidden Parameter Markov Decision Processes. In *Advances in Neural Information Processing Systems*, 6250–6261.

Kingma, D. P.; and Ba, J. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980* .

Lazaric, A. 2012. Transfer in Reinforcement Learning: a Framework and a Survey. In *Reinforcement Learning*, 143–173. Springer.

Maas, A. L.; Hannun, A. Y.; and Ng, A. Y. 2013. Rectifier Nonlinearities Improve Neural Network Acoustic Models.

In *Proceedings of the International Conference on Machine Learning*, volume 30, 3.

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): 529–533.

Monahan, G. E. 1982. State of the Art—A Survey of Partially Observable Markov Decision Processes: Theory, Models, and Algorithms. *Management Science* 28(1): 1–16.

Mysore, S.; Platt, R.; and Saenko, K. 2019. Reward-guided Curriculum for Robust Reinforcement Learning .

Ng, A. Y. 2004. Feature Selection, L 1 vs. L 2 Regularization, and Rotational Invariance. In *Proceedings of the International Conference on Machine Learning*, 78.

Nichol, A.; and Schulman, J. 2018. Reptile: a Scalable Metalearning Algorithm. *arXiv preprint arXiv:1803.02999* 2: 2.

Oh, J.; Guo, X.; Lee, H.; Lewis, R. L.; and Singh, S. 2015. action-Conditional Video Prediction using Deep Networks in Atari Games. In *Advances in Neural Information Processing Systems*, 2863–2871.

Rubner, Y.; Tomasi, C.; and Guibas, L. J. 2000. The Earth Mover's Distance as a Metric for Image Retrieval. *International Journal of Computer Vision* 40(2): 99–121.

Sadeghi, F.; and Levine, S. 2016. Cad2rl: Real Single-Image Flight without a Single Real Image. *arXiv preprint arXiv:1611.04201* .

Sadeghi, F.; Toshev, A.; Jang, E.; and Levine, S. 2018. Sim2Real Viewpoint Invariant Visual Servoing by Recurrent Control. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4691–4699.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347* .

Scott, D. W. 2015. *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley & Sons.

Simard, P. Y.; Steinkraus, D.; Platt, J. C.; et al. 2003. Best practices for convolutional neural networks applied to visual document analysis. In *Proceedings of the International Conference on Document Analysis and Recognition*, volume 3.

Slaoui, R. B.; Clements, W. R.; Foerster, J. N.; and Toth, S. 2019. Robust Domain Randomization for Reinforcement Learning. *arXiv preprint arXiv:1910.10537* .

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15(1): 1929–1958.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement Learning: An Introduction*. MIT press.

Tamar, A.; Wu, Y.; Thomas, G.; Levine, S.; and Abbeel, P. 2016. Value Iteration Networks. In *Advances in Neural Information Processing Systems*, 2154–2162.

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

Tieleman, T.; and Hinton, G. 2012. Lecture 6.5-RMSProp: Divide the Gradient by a Running Average of its Recent Magnitude. *COURSERA: Neural Networks for Machine Learning* 4(2): 26–31.

Tobin, J.; Fong, R.; Ray, A.; Schneider, J.; Zaremba, W.; and Abbeel, P. 2017. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 23–30. IEEE.

Tzeng, E.; Devin, C.; Hoffman, J.; Finn, C.; Abbeel, P.; Levine, S.; Saenko, K.; and Darrell, T. 2020. Adapting Deep Visuomotor Representations with Weak Pairwise Constraints. In *Algorithmic Foundations of Robotics XII*, 688–703. Springer.

Tzeng, E.; Hoffman, J.; Darrell, T.; and Saenko, K. 2015. Simultaneous Deep Transfer Across Domains and Tasks. In *Proceedings of the IEEE International Conference on Computer Vision*, 4068–4076.

Tzeng, E.; Hoffman, J.; Saenko, K.; and Darrell, T. 2017. Adversarial Discriminative Domain Adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7167–7176.

Villani, C. 2008. *Optimal Transport: Old and New*, volume 338. Springer Science & Business Media.

Wold, S.; Esbensen, K.; and Geladi, P. 1987. Principal Component Analysis. *Chemometrics and Intelligent Laboratory Systems* 2(1-3): 37–52.

Wulfmeier, M.; Posner, I.; and Abbeel, P. 2017. Mutual alignment transfer learning. *arXiv preprint arXiv:1707.07907* .

Yao, J.; Killian, T.; Konidaris, G.; and Doshi-Velez, F. 2018. Direct Policy Transfer via Hidden Parameter Markov Decision Processes. In *Proceedings of the Lifelong Learning: A Reinforcement Learning Approach Workshop*.

Zhang, A.; Lyle, C.; Sodhani, S.; Filos, A.; Kwiatkowska, M.; Pineau, J.; Gal, Y.; and Precup, D. 2020. Invariant Causal Prediction for Block MDPs. *arXiv preprint arXiv:2003.06016* .

Zhang, C.; Vinyals, O.; Munos, R.; and Bengio, S. 2018. A Study on Overfitting in Deep Reinforcement Learning. *arXiv preprint arXiv:1804.06893* .

Zhang, J.; Tai, L.; Yun, P.; Xiong, Y.; Liu, M.; Boedecker, J.; and Burgard, W. 2019. VR-Goggles for Robots: Real-to-Sim Domain Adaptation for Visual Control. *IEEE Robotics and Automation Letters* 4(2): 1148–1155.

Zhu, J.-Y.; Park, T.; Isola, P.; and Efros, A. A. 2017. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 2223–2232.

Appendix

Algorithm

The goal of the RL network is to simultaneously minimize the PPO loss $\mathcal{L}_{\text{RL}}$ and the confusion loss $\mathcal{L}_{\text{Conf}}$. To compute these losses, it samples observations, actions, and rewards from the source environment and observations from the target environment. f is a function that approximates the Wasserstein distance between the source observation and target observation distributions. Thus, it should be trained to convergence for every update of the RL network. As in Wasserstein GAN, it is optimized for n_{critic} steps for each update of the RL network. This process is outlined in Algorithm 1. Note that we use the weight clipping method defined in Arjovsky, Chintala, and Bottou (2017) rather than the gradient penalty method defined in Gulrajani et al. (2017) to directly test the effect of the novel Wasserstein Confusion loss term. Combining Wasserstein Confusion and gradient penalty is a promising direction for future work.

Algorithm 1: Wasserstein Adversarial PPO

for $t = 0, ..., n_{timesteps}$ **do**
  for $j = 0, ..., n_{critic}$ **do**
    Sample $\{\mathfrak{s}_{s,i}\}_{i=1}^{m} \sim P_s$ a batch from the source domain
    Sample $\{\mathfrak{s}_{t,i}\}_{i=1}^{m} \sim P_t$ a batch from the target domain buffer
    $\eth_w \leftarrow \nabla_w[\frac{1}{m}\sum_{i=1}^{m} f_w(h_\theta(\mathfrak{s}_{s,i})) - \frac{1}{m}\sum_{i=1} f_w(h_\theta(\mathfrak{s}_{t,i}))]$
    $w \leftarrow w + \alpha\times$ RMSProp$(w, \eth_w)$
  end
  Sample $\{\mathfrak{s}_{s,i}, a_{s,i}, r_{s,i}\}_{i=1}^{m} \sim P_s$ a batch from the source domain
  Sample $\{\mathfrak{s}_{t,i}\}_{i=1}^{m} \sim P_t$ a batch from the target domain buffer
  $\eth_\theta \leftarrow \nabla_\theta[-\frac{1}{m}\sum_{i=1}^{m} f_w(h_\theta(\mathfrak{s}_{s,i})) + \frac{1}{m}\sum_{i=1} f_w(h_\theta(\mathfrak{s}_{t,i})) + \mathcal{L}_{\text{RL}}(\mathfrak{s}_{s,1}, a_{s,1}, r_{s,1}, ..., \mathfrak{s}_{s,m}, a_{s,m}, r_{s,m})]$
  $\theta \leftarrow \theta - \alpha\times$ RMSProp$(\theta, \eth_\theta)$
end

Implementation Details

There are four algorithms that we implement: PPO, Robust Domain Randomization, VR Goggles, and WAPPO. All four are built off of PPO (Schulman et al. 2017). We use the high quality, open source implementation of PPO provided by OpenAI Baselines (Dhariwal et al. 2017). Furthermore, we use the neural architecture and learning rate provided by (Cobbe et al. 2019a) as a baseline. This architecture consists of the CNN component from the IMPALA network (Espeholt et al. 2018), which is shared between the policy and value prediction components of PPO. The value and policy networks then branch and each have one fully connected layer which outputs the policy or value, respectively. As in (Cobbe et al. 2019a), we use a learning rate of 5×10^{-4}. We evaluated with both the Adam Optimizer (Kingma and Ba 2014) and the RMSProp Optimizer (Tieleman and Hinton 2012) but find negligible difference. The network architecture is outlined in figure 6. The adversarial critic is made of 9 stacked fully connected layers of width 512 separated by Leaky ReLU activations.

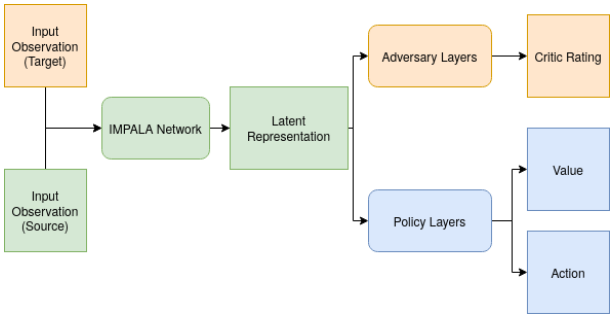


Figure 5: Network architecture. Layers are represented as rounded rectangles. Blue indicates use in training the RL policy, orange indicates use in training the critic, and green indicates use in training both. Note that the network architecture mirrors that of domain confusion and randomization but is modified to work with Reinforcement Learning rather than Supervised Learning (Tzeng et al. 2015, 2017; Ganin et al. 2016). The combination of the green and blue networks is identical in architecture to the IMPALA network used to benchmark the Procgen Environments and mirrors that used in Robust Domain Randomization (Cobbe et al. 2019a; Slaoui et al. 2019). Only the green and blue networks were used when measuring the performance of PPO and Robust Domain Randomization.

For PPO, Robust Domain Randomization, and VR Goggles, the network does not have an adversary. As such, we used the green and blue sections depicted in Figure 5 to train the PPO agent used in both these algorithms. The VR Goggles training process is the same as that used in Zhang et al. (2019). Specifically, we collect a dataset of 2000 source domain images and 2000 target domain images and train the VR Goggles translation network with a learning rate of 2×10^{-4} for 50 epochs. As in Zhang et al. (2019), we found no performance gain when using a larger dataset or training for more epochs. As Zhang et al. (2019) does not provide an implementation, we re-implement this method by adding their novel shift loss to the open source CycleGAN implementation (Zhu et al. 2017). For Robust Domain Randomization, we implement the regularization loss and use a regularization weight of 10 and as described in Slaoui et al. (2019).

For WAPPO, we use the entire network depicted in Figure 5. The green and blue sections are optimized according to the PPO loss $\mathcal{L}_{\text{PPO}}$ and the green and orange sections are optimized according to the Wasserstein Confusion loss $\mathcal{L}_{\text{Conf}}$. Similarly to (Arjovsky, Chintala, and Bottou 2017), we take 5 gradient steps of the adversary network per step of the RL network. The adversarial critic network is made of 8 dense layers of width 512, separated by Leaky ReLU (Maas, Hannun, and Ng 2013) activation function.

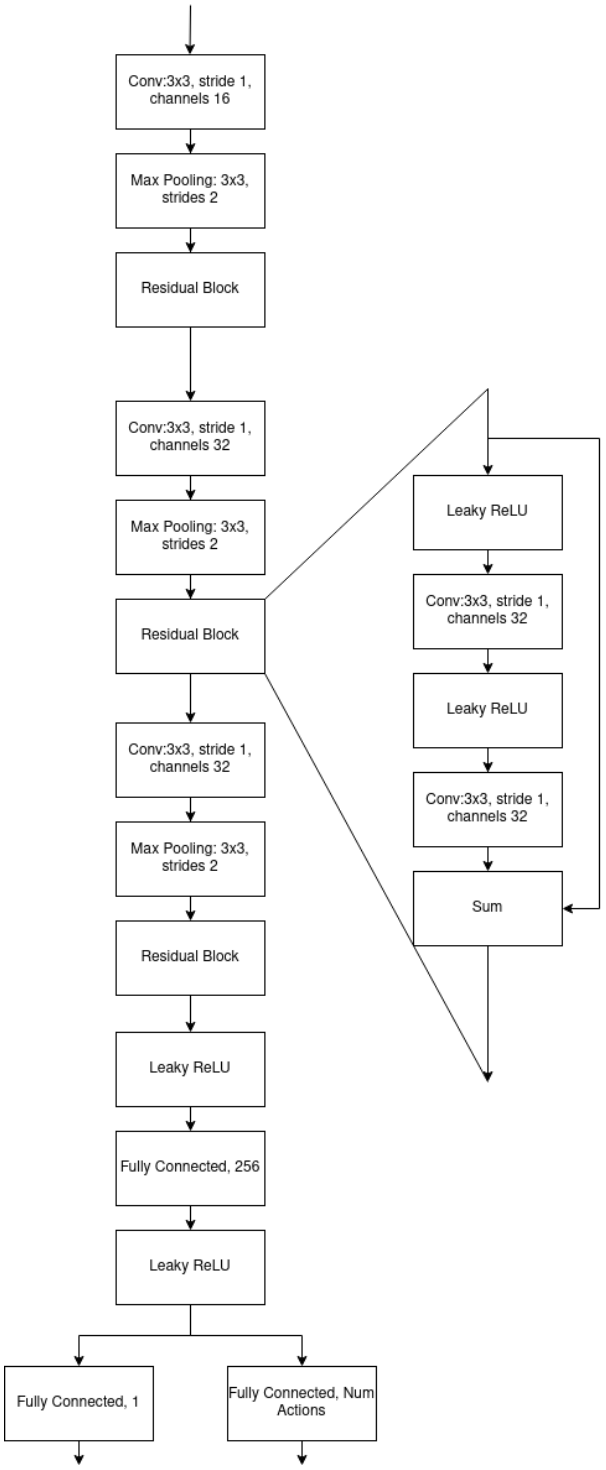


Figure 6: Detailed RL Network architecture. Residual Blocks are detailed on the right and have the same number of channels as their input. The value-specific fully connected layer is on the right and the policy-specific fully connected layer is on the left. Network architecture is taken from Cobbe et al. (2019a) but uses Leaky ReLU activations.