Mutual Alignment Transfer Learning

Markus Wulfmeier[*]
Oxford Robotics Institute
University of Oxford
markusw@robots.ox.ac.uk

Ingmar Posner
Oxford Robotics Institute
University of Oxford
ingmar@robots.ox.ac.uk

Pieter Abbeel
Berkeley AI Research (BAIR)
University of California Berkeley
OpenAI, San Francisco
pabbeel@cs.berkeley.edu

Abstract: Training robots for operation in the real world is a complex, time consuming and potentially expensive task. Despite significant success of reinforcement learning in games and simulations, research in real robot applications has not been able to match similar progress. While sample complexity can be reduced by training policies in simulation, such policies can perform sub-optimally on the real platform given imperfect calibration of model dynamics. We present an approach – supplemental to fine tuning on the real robot – to further benefit from parallel access to a simulator during training and reduce sample requirements on the real robot. The developed approach harnesses auxiliary rewards to guide the exploration for the real world agent based on the proficiency of the agent in simulation and vice versa. In this context, we demonstrate empirically that the reciprocal alignment for both agents provides further benefit as the agent in simulation can optimize its behaviour for states commonly visited by the real-world agent.

Keywords: Transfer Learning, Simulation, Robotics, Adversarial Learning

1 Introduction

Recent work in reinforcement learning has led to significant successes such as outperforming humans on a multitude of computer games [1, 2] and surpassing the best human players in the games of Chess [3] and Go [4]. The principal commonality between these settings is the availability of virtually unlimited training data as these systems can be trained in parallel and significantly faster than real-time, real-world executions.

However, training agents for operation in the real world presents a significant challenge to the reinforcement learning paradigm as it is constrained to learn from comparatively expensive and slow task executions. In addition, limits in the perception system and the complexity of manually providing informative real world rewards for high complexity tasks often result in only sparse and uninformative feedback being available, further increasing sampling requirements. As a result, many tasks which involve physical interaction with the real world and are simple for humans, present insurmountable challenges for robots [5].

While there has been significant progress towards fast and reliable simulators [6, 7, 8], they do not represent exact replications of the platforms and environments we intend to emulate. Systematic model discrepancies commonly prevent us from directly porting policies from simulation to the real platform.

The principal differences between simulation and real world are based on the type of system observations as well as discrepancies between system dynamics. Recent developments aim at designing visually more similar environments [9, 10] and current research targets adapting policies to be invariant with respect to differences between the observation spaces of simulator and real platform [11, 12, 13, 14].

[*] Work done as visiting scholar at Berkeley Artificial Intelligence Lab (BAIR), UC Berkeley

1st Conference on Robot Learning (CoRL 2017), Mountain View, United States.

Fine tuning pretrained policies from simulation on the real platform is a straightforward approach to address discrepancies between both systems' dynamics. However, as policies trained via reinforcement learning will learn to exploit the specific characteristics of a system – optimizing for mastery instead of generality – a policy can overfit to the simulation. The resulting initialization can prove unfavourable to random initializations for further training on the real platform as it might inhibit exploration and lead to the optimization process getting stuck in local optima. A phenomenon which we demonstrate in the experiments in Section 4.5. On the other hand, in cases where fine tuning improves performance it can be straightforwardly combined with the presented approach as described in Section 4.

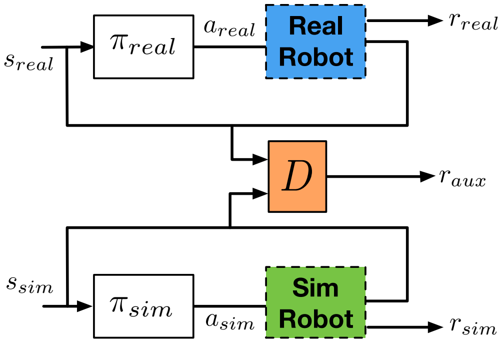


Figure 1: Simplified schema for Mutual Alignment Transfer Learning. Both systems are trained to not only maximize their respective environment rewards but also auxiliary alignment rewards that encourages both systems to occupy similar distributions over visited states. Furthermore, the simulation policy can be trained orders of magnitude faster than the real platform solely based on its environment reward.

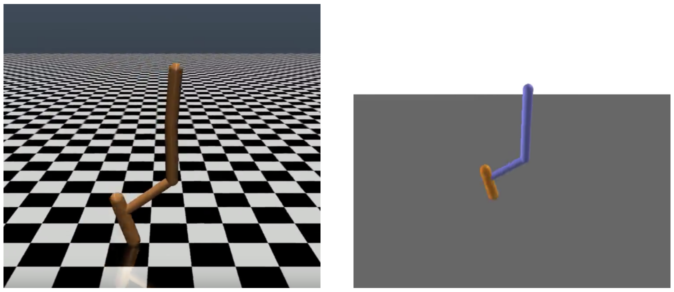
While the actions performed by the simulator policy can fail to accomplish the task on the robot, the sequence of states visited by the agent in simulation represents its task under limited variation in the system dynamics. We propose Mutual Alignment Transfer Learning (MATL), which instead of directly adapting the simulation policy, guides the exploration for both systems towards mutually aligned state distributions via auxiliary rewards. The method is displayed in Figure 1 and employs an adversarial approach to train policies with additional rewards based on confusing a discriminator with respect to the originating system for state sequences visited by the agents. By guiding the target agent on the robot towards states that the potentially more proficient source agent visits in simulation, we can accelerate training. In addition to aligning the robot policy to adapt to progress in simulation, we extend the approach to mutually align both systems which can be beneficial as the agent in simulation will be driven to explore better trajectories from states visited by the real-world policy.

We evaluate the method developed on a set of common reinforcement learning benchmark tasks [15] to transfer between simulations with differences in system parameters such as density, dampening and friction. Furthermore, we extend the experiments to address additional challenges relevant in the context of real platforms such as sparse and uninformative rewards. The final experiments investigate transfer between different simulation engines with unknown discrepancies as a stronger proxy for real robot experiments.

We demonstrate that auxiliary rewards, which guide the exploration on the target platform, improve performance in environments with sparse rewards and can even guide the agent if only uninformative or no environment rewards at all are given for the target agent. Furthermore, the approach proves to be capable of guiding training given scenarios with significant discrepancies between the system dynamics when direct transfer or fine tuning approaches fail as investigated in Section 4.5.

2 Related Work

Given significant advances when training machine learning models and in particular reinforcement learning policies in simulations, the field of transfer learning has gained increased relevance in recent years. In some cases the simulator-given environment represents the final application environment like e.g. when solving computer games, such as various Atari games [1] and Ms. Pacman [2], as



(a) OpenAI gym - MuJoCo

(b) DartEnv - DART

Figure 2: Hopper2D task build atop MuJoCo [7] and DART [6] simulation engines. Both reinforcement learning environments are provided by OpenAI gym [27] and DartEnv [28] respectively. The environments have the same state and action spaces - however differ in the underlying system dynamics and in particular contact modelling.

well as board games, such as Chess [3] and Go [4]. In most cases however, the direct application of models trained in simulation to real world tasks results in significantly decreased performance. This can be due to two possible types of discrepancies: different observation model or different system dynamics.

The problem of different observation distribution has been addressed generally in the framework of unsupervised domain adaptation [14, 13] as well as with particular application to robotics [16, 11]. Rusu et al. [17] tackles the transfer task by reusing features learned for simulation and focusing on learning residual representations needed to adapt to the real world task. Furthermore, recent work addresses the task via visual domain randomization, showing that exact simulation might not be needed when instead training for various variations of textures and colors [12, 18] without any training data required from the final application system.

The second challenge is based on differences of the system dynamics of simulator and real task, which results in different optimal policies even when given the same observation model, is the focus of our work. Christiano et al. [19] address the challenge by learning an inverse dynamics model for the real platform to determine the correct real world action based on the trajectory in simulation. Gupta et al. [20] learn invariant feature spaces which are employed to transfer task knowledge even between systematically different robot architectures. However, the approach relies on prior information through proxy tasks to determine the invariant spaces. Another approach is given by Rajeswaran et al. [21], who train robust policies based on sampling from an ensemble of simulated source domains while focusing on simulations with reduced performance. Additional methods from [22, 23] focus on exploiting inaccurate simulations to improve real world performance with the latter employing bayesian optimisation to improve the split between simulator and real world experiments.

Our method builds on a different idea of transfer learning which has its roots in imitation learning where we align the distributions over visited states between two agents. The alignment procedure is realized by training in an adversarial framework [24] to confuse a discriminator that itself is learning to classify states based on which system they originated in. The approach is conceptually similar to adversarial imitation learning methods [25, 11, 26] which train for one-sided alignment between the agent's policy and a set of demonstration trajectories. Instead our method trains both – simulator and robot – policies with an auxiliary reward based on the mutual alignment of visited state distributions between both systems as described in greater detail in Section 3.1.

3 Method

We consider a simulation to real robot transfer learning scenario for reinforcement learning with the setup of two agents acting in the respective source and target environments, each represented as Markov Decision Process (MDP). State space $s \in \mathcal{S}$ and action space $a \in \mathcal{A}$ are equal in both environments. However, the difference lies in the underlying system dynamics represented as $p_S(s_{t+1}|s_t, a_t)$ for the simulator and $p_R(s_{t+1}|s_t, a_t)$ for the real platform. The reward functions $r_S(s_t, a_t)$ and $r_R(s_t, a_t)$ which both agents optimize, can be the same for both environments, but can also differ as it is possible that we do not have access to the full reward function in the real world. However, both agents are intended to solve is the same task. Both agents, simulation/source and robot/target, act according to their respective stochastic policies $\pi_\theta(a_t|s_t)$ and $\pi_\phi(a_t|s_t)$, which are parameterized as neural networks by θ and ϕ. In the following sections the terms simulator and robot, or respectively source and target are used interchangeably.

3.1 Mutual Alignment Transfer Learning

Mutual Alignment Transfer Learning (MATL) leverages information gained while training in simulation to improve performance on a target robot with potentially different system dynamics via the introduction of auxiliary reward functions to guide both systems to visit similar states or state sequences. The approach is visualized in Figure 1. When the simulator policy cannot be directly applied due to systematic differences, the simulation trajectories still contain information about how to solve the task on a different system given limited variation in the underlying system dynamics.

The method trains policies with auxiliary alignment rewards by running reinforcement learning algorithms simultaneously on both systems, here Trust Region Policy Optimization (TRPO) [29]. As training in simulation can potentially be performed orders of magnitude faster, we run the updates for the simulator policy at M times higher rate with only the environment reward being used for these updates. Both policies, $\pi_\theta(a|s)$ and $\pi_\phi(a|s)$, are trained via TRPO with the method's gradient step computed following Equation 1 based on their respective reward signals $r(s, a)$.

$$\mathbb{E}_{\pi,p}[\nabla \ \log \pi(a_t|s_t)\, r(s_t, a_t)] \tag{1}$$

Auxiliary rewards for the alignment process are generated in an adversarial setting with the objective to confuse the discriminator D_ω which is trained to classify the system of origin for state sequences ζ_t from simulation and robot. As displayed in Equation 2, state sequences can be subsampled to ensure significant change between successive states. In addition to aligning the robot policy to adapt to progress in simulation, the reciprocal alignment of the simulation policy can be beneficial as the agent in simulation will be driven to explore better behaviours from states visited by the robot agent.

The discriminator D_ω is parameterized as neural network with weights ω and trained according to Equation 2 to classify the originating system (simulation-robot) of fixed length state sequences ζ_t.

$$\begin{gathered}\mathcal{L}_\mathcal{D} = \mathbb{E}_{\pi_\theta}[\log(D_\omega(\zeta_t))] + \mathbb{E}_{\pi_\phi}[\log(1 - D_\omega(\zeta_t))] \\ \zeta_t = s_t, s_{t+k}, s_{t+2k}, ..., s_{t+nk} \quad \text{with } n \in \mathbb{N}_0; k \in \mathbb{N}\end{gathered} \tag{2}$$

Additionally to the simulator and robot environment feedback, respectively $r_S(s_t, a_t)$ and $r_R(s_t, a_t)$, the full reward includes the auxiliary rewards ρ_S and ρ_R, for simulator policy and robot policy, as given in Equation 3. By training both policies towards mutual alignment, not only does the robot policy learn from the progress in simulation but also is the simulator policy pushed to explore better behaviour for states visited in the robot environment. The specific formulation in Equations 4 and 5 builds upon the idea of maximizing the confusion loss formulation of the GAN framework [24] , which was found empirically to be better suited for the transfer task than the original min-max formulation. The confusion objective for adversarial training addresses a principal shortcoming of the original formulation by which the gradients for the generating module (represented here by the policies) vanish when the discriminator performance is maximized. Hereinafter, the subscripts R and S stand for reference to robot and simulator systems respectively.

$$r(s_t, a_t) \quad = \quad \begin{cases} r_R(s_t, a_t) + \lambda\, \rho_R(s_t) & \text{: robot agent} \\ r_S(s_t, a_t) + \lambda\, \rho_S(s_t) & \text{: simulator agent} \end{cases} \tag{3}$$

$$\rho_S(s_t) \quad = \quad -\log(D_\omega(\zeta_t)) \tag{4}$$

$$\rho_R(s_t) \quad = \quad \log(D_\omega(\zeta_t)) \tag{5}$$

In conclusion, the full gradient steps for TRPO in the MATL framework are obtained by combining Equations and 1, 3, 4 and 5. Both updates for the simulator and robot policies are given in Equations 6 and 7 respectively, with the complete training procedure in Algorithm 1.

$$\mathbb{E}_{\pi_\theta}[\nabla_\theta \, \log \pi_\theta(a_t|s_t) \, (r_S(s_t, a_t) - \lambda \, \log(D_\omega(\zeta_t)))] \tag{6}$$

$$\mathbb{E}_{\pi_\phi}[\nabla_\phi \, \log \pi_\phi(a_t|s_t) \, (r_R(s_t, a_t) + \lambda \, \log(D_\omega(\zeta_t))] \tag{7}$$

Algorithm 1: Mutual Alignment Transfer Learning

```
Input  : environments MDP$_{R,S}$, alignment weight $\lambda$, iterations outer loop $N$ and inner loop $M$,
         rollout horizon $T$
Output: target policy $\pi_\phi$
$\pi_\theta, \pi_\phi, D_\omega \leftarrow$ initialize
for $i \leftarrow 1$ to $N$ do
    $(s_{0..T}, a_{0..T}, r_{0..T})_S \leftarrow$ rollout $\pi_\theta$ on MDP$_S$
    $(s_{0..T}, a_{0..T}, r_{0..T})_R \leftarrow$ rollout $\pi_\phi$ on MDP$_R$
    $D_\omega \leftarrow$ gradient update following Eq. 2
    $\pi_\theta, \pi_\phi \leftarrow$TRPO updates following Eqs. 6 & 7
    for $j \leftarrow 1$ to $M$ do
        $(s_{0..T}, a_{0..T}, r_{0..T})_S \leftarrow$ rollout $\pi_\theta$ on MDP$_S$
        $\pi_\theta \leftarrow$TRPO update following Eq. 6 with $\rho_S(s_t, a_t) = 0 \; \forall \; t \in T$
    end
end
```

4 Experiments

We evaluate MATL on transfer learning scenarios based on various common reinforcement learning tasks including environments taken from rllab [15], OpenAI gym [27] and DartEnv [28]. For Sections 4.2 to 4.4 we use variations of the same simulation environment with strongly varied parameters for the system dynamics. To evaluate our approach for application with significantly inaccurate knowledge of system dynamics, we severely alter the parameters for joint dampening, friction and densities between both systems. The reader is referred to the additional documentation[2] for detailed information about algorithm and environment parameterization. To extend the evaluation, we address transfer learning between two different simulation engines, MuJoCo [7] and DART [6], in Section 4.5.

The evaluation focuses six different approaches in the following experiments:

- independent - Independent training of the robot policy without auxiliary rewards.
- direct_transfer - Direct application of the simulator policy on the real platform.
- fine_tuning - Transfer of the fully trained simulator policy and subsequent fine tuning based only on robot environment rewards.
- MATLu - Unilateral alignment training with environment rewards but without auxiliary reward for the simulation policy.
- MATL - Mutual alignment training with environment rewards and both auxiliary rewards.
- MATLf - Combining MATL with fine tuning. Training via MATL but starting from a transferred fully trained simulator policy.

All diagrams focus on the number of real world iterations as additional iterations in simulation can be obtained at significantly higher rate and (in comparison) negligible effort. The performance for each experiment is normalized between zero and one.

4.1 Guiding Questions

- Are MATL's auxiliary alignment rewards suited to guide exploration in environments with sparse rewards? (Section 4.2)
- Do the auxiliary rewards provide enough feedback on how to solve a task in a situation where the real world agent has only access to an uninformative reward (such as a reward for not falling)? (Section 4.3)

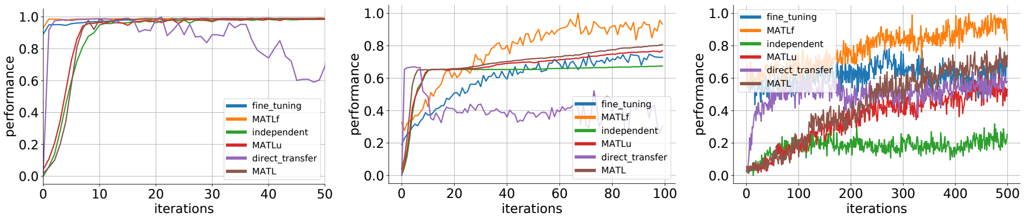
[2] Additional information to environment and algorithm parameterization can be found under sites.google.com/view/matl

- Can we succeed in training towards a task with no reward given in the target environment other than the auxiliary rewards (additionally deactivating all environment rewards in the target environment including the reward for staying alive)? (Section 4.4)
- How important is the mutual alignment vs a unilateral alignment of just the target policy? (Section 4.2 to Section 4.5)
- How well does the approach handle transfer between different simulation frameworks with unknown differences in system dynamics? (Section 4.5)

4.2 Sparse Rewards

The first section of our evaluation focuses on the application of MATL to environments with sparse rewards. In these experiments, both environments - simulation and real - only have access to sparse feedback which is given when the agent is situated within a distance ϵ to the target position[2] instead of a dense, distance based reward.

Sparse feedback renders these situations more complex than scenarios with dense rewards as the agents only seldomly get feedback and learning and progress is delayed. The auxiliary rewards given by MATL however are non-sparse and can help guide the real world agent to solve the task, given it has learned to solve the task in simulation. For these simpler tasks it was found empirically sufficient to apply the discriminator to single states.



(a) Cartpole (b) Cartpole Swingup (c) Reacher2D

Figure 3: Transfer learning with sparse rewards in both environments. The dense auxiliary rewards from MATL help to accelerate training. While fine tuning already learns to perform optimally after a few iterations, MATLf accelerates training even further.

The direct transfer of the simulation policy is unsuited for application on the target system in both tasks as displayed in Figure 3. While independent training already learns fast and fine tuning adapts even better, the best performance is achieved by combining MATL with fine tuning.

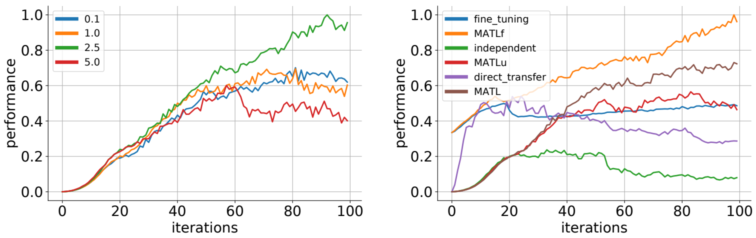
4.3 Uninformative Rewards

We furthermore investigate scenarios limited to an uninformative reward for locomotion tasks. In these cases, the only remaining component is a cost for falling in the target environment. The agent in simulation is still provided with access to the full reward including a forward-guiding component. We evaluate these scenarios based on how well the agent learns to move forward and therefore the capability of MATL to guide to forward motion. The performance is now given as metric based on the average final distance of the agent in the direction of locomotion.

Similar to Stadie et al. [11] we exploit in this section the sequential structure of these tasks and apply the discriminator to state pairs of time-steps t and t+4, which has been shown to work well across a variety of different tasks.

The most robust policy is simply standing still as moving forward increases the risk of falling. This situation renders the task more complex as the auxiliary reward has to overcome a potentially conflicting reward signal. Figure 4a displays that the alignment parameter λ is highly relevant to encourage stable walking behaviour. The conflict between the reward components results in a more conservative, ankle-based running style as can be found in the videos as part of the additional material[2].

Figure 4 shows that independent training results in very limited motion as reward signal encourages the agent to stand still. Fine tuning as well as direct transfer of the simulation policy perform slightly better, but are significantly surpassed by all versions of MATL.



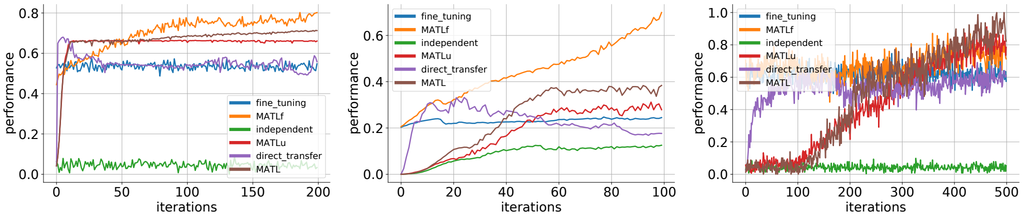
(a) Hopper2D - Alignment weight (b) Hopper2D - Methods

Figure 4: Transfer learning - Hopper2D task with uninformative rewards. Left: Transfer learning performance in dependency of alignment weight. The context of potentially conflicting rewards renders the hyperparameter choice more critical for efficient transfer. Right: Comparison against baselines with MATLf surpassing the other methods.

4.4 Without Robot Environment Reward

To evaluate if transfer is possible without any environment reward in the target environment we run a set of experiments with only auxiliary reward based updates for the robot policy. The changes in parameterization between simulation and robot environments are equal to the settings for Sections 4.2 and 4.3.

Contrary to the other sections, the performance is given for these experiments as ratio of the maximum performance achieved in the same environment with available environment reward from Section 4.2 and 4.3. This metric is chosen to evaluate the relative performance reduction for all methods when access to environment rewards is prevented.



(a) Cartpole Swingup (b) Hopper2D (c) Reacher2D

Figure 5: Transfer learning - Scenario without environment reward on the real platform

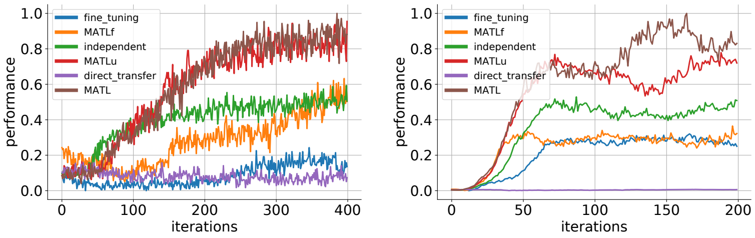
In all scenarios, MATLf – the combination of fine tuning and auxiliary rewards – outperforms the other methods. This fine-tuning of a simulator-trained policy via MATL improves from a well-suited initialization, often exceeding other versions of MATL in its maximally achieved reward. As expected, these environments result in no significant progress for methods training only on target rewards such as fine tuning, which keeps the same performance as the pretrained policy.

4.5 MuJoCo to DART

While the earlier experiments are based on the MuJoCo simulator and varied parameters for different system properties, we extend the evaluation of our approach towards differences between two simulation software packages, namely from MuJoCo [7] to DART [6]. An additional challenge of this transfer experiment is that not only parameter values vary but the underlying algorithms differ in terms of parameter types and in particular contact modeling [28].

The confusion loss formulation for the original GAN framework did not result in improvement for these experiments and presented results are based on an adaptation of the Wasserstein GAN (WGAN) loss [30], which is described in greater detail as part of the additional materials.

As the DART environments differ significantly from their MuJoCo counterparts, direct transfer as well as fine tuning result in low performance. In particular, it can be seen that pretraining in one environment results to a low quality initialization and independently training on the target environment surpasses the fine tuning approach. This inherently also leads to low performance for MATLf which builds on the pretrained policy, but does not affect MATLu and MATL.



(a) Reacher (b) Hopper2D

Figure 6: Transfer learning - MuJoCo to DART simulator

5 Discussion

MATL has been shown to work under significant differences in system dynamics between source and target platform as demonstrated in Section 4, including situations when direct transfer of the simulator policy fails to show good performance. A current shortcoming is the potential instability of the adversarial training framework and connected effort in tuning the hyperparameters. The alignment weight parameter λ is of particular importance in the context of potentially conflicting rewards as is represented by the uninformative rewards in Section 4.3. The weight has to be increased above the value of 0.1 which is used for most other experiments as the safety based environment rewards will prevent exploration.

Different simulation engines, as given in Section 4.5, provide a particular challenge for the transfer learning methods. We show that in these cases, the simulation based policy can overfit and result in providing an unsuitable initialization for fine tuning which performs worse during training than standard random initializations. Nevertheless, MATL demonstrably accelerates training under these conditions. Mutual alignment additionally increases performance consistently across all locomotion tasks (Sections 4.2 and 4.4) while being commensurate with unilateral alignment on the tasks with sparse rewards given in Section 4.2.

While distribution based alignment has been demonstrated to work well in the experiments in Section 4, evaluations based on the straightforward approach of direct alignment between the states along trajectories of each system only lead to limited performance improvements. Auxiliary rewards based on state-wise alignment of simulator and real world trajectories perform adequate mostly in low dimensional tasks with near-deterministic behaviour of the completely trained agent. The task of moving a ball towards a goal in 2D by applying forces in 2 directions serves as an example for this kind of scenario, where the optimal trajectory is - independent of initialization - a straight line between start and target.

6 Conclusions and Future Directions

We present an approach for transfer learning under discrepancies in system dynamics for simulation to robot transfer. The approach relies on parallel training of both agents and employs auxiliary rewards to align their respective distributions over visited states and can be straightforwardly supplemented with ideas based on fine tuning.

Guiding robot exploration via alignment of state distributions between both systems has been shown to be beneficial for accelerating training and potentially lead to better final performance in scenarios with sparse or uninformative rewards for the target platform.

All experiments included in this paper are concerned with transfer learning between two simulations with either different parameterizations or completely different simulation engines to create situations of misaligned and unknown system dynamics. Future work will address the full simulation to robot transfer scenario. As MATL employs partially trained policies on the real platform, further development will have to address methods for safe application of RL on real world platforms [31, 32]. Furthermore, one of the principal challenges going forward is the weighting of auxiliary rewards and original environment rewards in particular in situations where both can lead to conflicting behaviours.

Acknowledgments

The authors would like to acknowledge the support of the UKs Engineering and Physical Sciences Research Council (EPSRC) through the Doctoral Training Award (DTA) as well as the support of the Hans-Lenze-Foundation.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[2] H. van Seijen, M. Fatemi, J. Romoff, R. Laroche, T. Barnes, and J. Tsang. Hybrid Reward Architecture for Reinforcement Learning. *CoRR*, abs/1706.04208, 2017. URL `http://arxiv.org/abs/1706.04208`.

[3] M. Campbell, A. J. Hoane, and F.-h. Hsu. Deep blue. *Artificial intelligence*, 134(1-2):57–83, 2002.

[4] Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587): 484–489, jan 2016. ISSN 0028-0836.

[5] H. Moravec. *Mind children: The future of robot and human intelligence*. Harvard University Press, 1988.

[6] Dart: Dynamic animation and robotics toolkit. `https://github.com/dartsim/dart/`, 2017.

[7] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.

[8] E. Coumans. Bullet physics simulation. In *ACM SIGGRAPH 2015 Courses*, page 7. ACM, 2015.

[9] Nvidia isaac: Virtual simulator for robots. `https://www.nvidia.com/en-us/deep-learning-ai/industries/robotics/`, 2017.

[10] S. Shah, D. Dey, C. Lovett, and A. Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, 2017. URL `https://arxiv.org/abs/1705.05065`.

[11] B. C. Stadie, P. Abbeel, and I. Sutskever. Third-person imitation learning. In *Proceedings of the 33rd International Conference on Learning Representations (ICLR)*, 2017.

[12] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *arXiv preprint arXiv:1703.06907*, 2017.

[13] M. Long, J. Wang, and M. I. Jordan. Deep transfer learning with joint adaptation networks. *CoRR*, abs/1605.06636, 2016. URL `http://arxiv.org/abs/1605.06636`.

[14] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, V. Lempitsky, U. Dogan, M. Kloft, F. Orabona, and T. Tommasi. Domain-Adversarial Training of Neural Networks. *Journal of Machine Learning Research*, 17:1–35, 2016.

[15] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel. Benchmarking deep reinforcement learning for continuous control. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, 2016.

[16] E. Tzeng, C. Devin, J. Hoffman, C. Finn, X. Peng, S. Levine, K. Saenko, and T. Darrell. Towards adapting deep visuomotor representations from simulated to real environments. *arXiv preprint arXiv:1511.07111*, 2015.

[17] A. A. Rusu, M. Vecerik, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell. Sim-to-real robot learning from pixels with progressive nets. *arXiv preprint arXiv:1610.04286*, 2016.

[18] F. Sadeghi and S. Levine. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

[19] P. Christiano, Z. Shah, I. Mordatch, J. Schneider, T. Blackwell, J. Tobin, P. Abbeel, and W. Zaremba. Transfer from simulation to real world through learning deep inverse dynamics model. *arXiv preprint arXiv:1610.03518*, 2016.

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

[21] A. Rajeswaran, S. Ghotra, S. Levine, and B. Ravindran. Epopt: Learning robust neural network policies using model ensembles. *arXiv preprint arXiv:1610.01283*, 2016.

[22] P. Abbeel, M. Quigley, and A. Y. Ng. Using inaccurate models in reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 1–8. ACM, 2006.

[23] A. Marco, F. Berkenkamp, P. Hennig, A. P. Schoellig, A. Krause, S. Schaal, and S. Trimpe. Virtual vs. real: Trading off simulations and physical experiments in reinforcement learning with bayesian optimization. *CoRR*, abs/1703.01250, 2017. URL `http://arxiv.org/abs/1703.01250`.

[24] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[25] J. Ho and S. Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573, 2016.

[26] K. Hausman, Y. Chebotar, S. Schaal, G. Sukhatme, and J. Lim. Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets. *arXiv preprint arXiv:1705.10479*, 2017.

[27] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[28] Dartenv: Openai gym environments transferred to the dart simulator. `https://github.com/DartEnv/dart-env/wiki/OpenAI-Gym-Environments`, 2017.

[29] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1889–1897, 2015.

[30] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[31] J. Achiam, D. Held, A. Tamar, and P. Abbeel. Constrained policy optimization. *arXiv preprint arXiv:1705.10528*, 2017.

[32] G. Kahn, A. Villaflor, V. Pong, P. Abbeel, and S. Levine. Uncertainty-aware reinforcement learning for collision avoidance. *arXiv preprint arXiv:1702.01182*, 2017.