Domain Adaptation in Unmanned Aerial Vehicles Navigation and Obstacle Avoidance using Deep Reinforcement Learning

Hemanth Kandula

Abstract

Recent advancements in deep reinforcement learning (RL) inspired end-to-end learning of Unmanned Aerial Vehicles (UAV) navigation. However, they can be slow to train and require lots of interactions with the environment, as these reinforcement learning algorithms have no prior knowledge about the environments or tasks. Transfer learning was shown to be useful to help in some problems in transferring knowledge from a source task to a target task. But most RL problems direct TL with fine-tuning might not be the best solution to transfer knowledge between tasks, environments. Our work presents an adversarial domain adaption method for UAV navigation and obstacle avoidance. We align state representations of pre-trained source domain with target domains and demonstrate in a realistic drone simulator that initialization with domain adaption showed significant performance improvements over RL task learned from scratch. Code for this method for drone RL task is available on GitHub:https://github.com/hemanthkandula/Drone-Navigation-Domain-Adaption.git

1 Introduction

The usage of artificial intelligence (AI) in unmanned system technologies increases drastically. Despite numerous advancements in sensor technologies and motion planning algorithms, autonomous navigation of micro air vehicles is still a challenging problem due to payload weight, size and power constraints which leads to limited sensors . Recent advancements in deep reinforcement learning [15] inspired end-to-end learning of Unmanned Aerial Vehicles (UAV) navigation, mapping directly from monocular images to actions. A modified fitted Q-iteration to train a policy only in simulation using deep reinforcement learning was proposed in [19]. Many other approaches were proposed to solve the UAV control problems using deep reinforcement learning approaches [5, 12, 13, 18, 9]

Despite this recent progress, in most these approaches significant improvements are made to improve single reinforcement learning tasks, improving data-efficiency and stability but still these deep RL solutions are learning the neural

networks from scratch[11].Training such algorithms take significant amount of time. This type of learning doesn't reuse the skills that are already learnt which can limits the adaptability to different task and versatility to different environments and constrain future advances to build efficient systems. For UAV learning agents this is one of the critical aspects is ability to learn how to generalize its behavior in a task efficiently and apply this generalized knowledge new domain or environment.

This problem is addressed in using transfer learning [11, 4, 10] nevertheless reuse of skills remains most significant open research problems for UAVs. This project aims to solves this by adapting state space representations from source task to target task. Hidden layers of the neural network contains these state space representation in deepRL . Domain adaption is a well know technique in Machine learning to do transfer learning to learn the common feature space in source and target domains. Adversarial domain adaptation for RL technique from [22] will be utilized to model the skills learnt in source domain to target domain. This method utilizes strategy from generative adversarial networks to transfer knowledge from source to target domain.

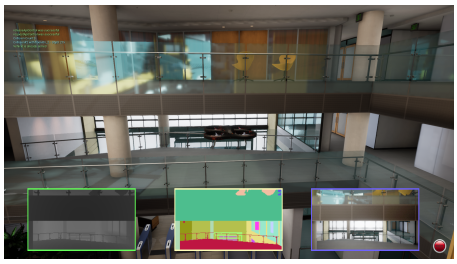
We applies this method on UAV in a simulator environment called airsim to boost the performance and training time compared to training the target task from scratch by random initialization. All our experiments were conducted in a physically and visually realistic simulation from Airsim simulator Fig. 1.Our architecture is shown in Figure **??** illustrates how a target samples from target encoder is sent to a discriminator network into believing it was from the source distribution. We show how this approach can be successfully applied to the UAV reinforcement learning problem across two domains to provide an performance boost over random initialization of the neural network.

2 Background:

This project involves concepts of UAV Perception-Based control, Reinforcement Learning, Deep-Q-learning algorithms, Domain Adaptation, Adversarial networks and AutoEncoders

2.1 UAV Perception-Based control with Deep Reinforcement Learning:

Reinforcement Learning is about learning from interaction how to behave in order to achieve a goal. The learner and decision-maker is called agent, while the thing it interacts with and therefore everything outside of the agent, is called the environment ε. The interaction takes place at each of a sequence of discrete time steps t. At each time step, the agent receives a state S_t from the state space S and selects an action A_t from the set of possible actions in the action space $A(S_t)$. One time step later the agent gets a numerical reward R_{t+1} from



(a) Indoor settings 1

(b) Indoor settings 2

(c) outdoor settings

Figure 1: Airsim environments

the environment as consequence of the previous action. Now the agent finds itself in a new state S_{t+1} The goal is to choose a policy so that it maximizes the total amount of reward. cumulative reward over time is called G_t

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+l+1}$$

In the above task we consider our RL objective as autonomous flight and obstacle avoidance. Agents interact with environment actions, observations and reward calculations. At each time step t, agents gets it's state of environment from a single camera frame. It takes an action A_t from the set of possible actions in the action space $A(S_t)$, which leads to new state S_{t+1}. Agents gets reward for each action section if it moved in the right direction if it receives reward accordingly.

DeepMind's DQN (deep Q-network) [14] was one of the first breakthrough successes in applying deep learning to RL. It used a neural net to learn Q-functions for classic Atari games such as Pong and Breakout, allowing the model to go straight from raw pixel input to an action.

Deep Q-Network: The overall goal of Deep Q-Network (DQN) is to use

a deep convolutional neural network to approximate the optimal action-value function, defined as:

$$Q^*(s,a) = \max_{\pi} \mathbb{E}[r_t + \gamma t_{t+1} + \gamma^2 t_{t+2} + ...|s_t = s, a_t = a]$$

if the optimal value $Q^*(s`, a`)$ of the sequence $s`$ at the next time-step was known for all possible actions $a`$ $Q^*(s,a) = \max_{\pi} \mathbb{E}_{s`\sim\epsilon}[r + \gamma Q^*(s`, a`)|s, a]$ neural network function approximator with weights θ as a Q-network. A Q-network can be trained by minimising a sequence of loss functions $L_i(\theta_i)$ that changes at each iteration i,

$$L_i(\theta_i) = \mathbb{E}_{s`,a\sim p(.)}[(y_i - Q(s,a;\theta_i)^2]$$

where

$$y_i = \begin{cases} R_T & \text{for terminal state } s_T \\ R_{t+1} + \gamma \max_{a`} Q(s_{t+1}, a`) & \text{for non terminal state } s_t \end{cases} \quad (1)$$

Differentiating the loss function with respect to the weights we arrive at the following gradient step,

$$\nabla_{\theta_i} L(\theta_i)) = \mathbb{E}_{s`,a\sim p(.),s`\sim\varepsilon[(R_{t+1}+\gamma \max_{a`} Q(s_{t+1},a`)-Q(s,a,\theta_i))\nabla_{\theta_i} Q(s,a,\theta_i)]}$$

In previous work [13] trained agents Three DQN algorithms were tested: DQN [14], double-DQN [23] and Dueling [24]. The best results were obtained by double-DQN , which showed to be at the level of a human tester. So Double DQN is chosen as the base algorithm for source task training. The Double DQN algorithm remains the same as the original DQN-algorithm, except replacing the target the estimated return as defined as DQN-target

2.2 Domain Adaptation:

Domain Adaptation is a transfer learning approach that seeks to align knowledge gained on a supervised source task with an unlabelled (or limited availability of labels) target dataset from a different domain. These datasets usually share prediction labels so it is only the representation of the information that has changed. Many methods seek to align the representation vector of the source and target data. This can be done in a supervised or unsupervised manner, depending on labelling assumptions, however target data is usually limited if it does exist. Aligning representations can be done in many ways; perhaps the simplest is to impose a constraint between the predicted representation of source and target data. These approaches show much promise, however, aligning samples from source and target must be considered. There are various approaches to this, from using supervised data [25], to assumed correspondences [10], to unsupervised approaches [22, 10].

2.3 Adversarial Domain Adaptation:

Generative adversarial networks (GANs) [8] are an exciting recent innovation in machine learning. GANs are generative models: they create new data instances that resemble your training data.In this framework there's a min-max adversarial game between two neural networks –. The Generator (G) is seeking to learn a generative model and the Discriminator (D) learns to distinguish between samples from the Generator's distribution and the True distribution. The loss function is as follows

$$\min_G \max_D E_{x\sim p_{data}}[logD(x)] + E_{z\sim p_z}[log(1 - D(G(z)))]$$

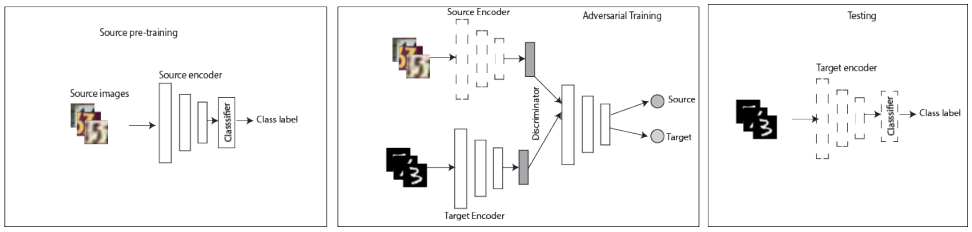


Figure 2: Adversarial Discriminative Domain Adaptation (ADDA) approach [22]. First pre-train a source encoder CNN using labeled source image examples. Next, perform adversarial adaptation by learning a target encoder CNN such that a discriminator that sees encoded source and target examples cannot reliably predict their domain label. During testing, target images are mapped with the target encoder to the shared feature space and classified by the source classifier. Dashed lines indicate fixed network parameters.

The Adversarial Discriminative Domain Adaptation (ADDA) [22] approach the new domain data passed through a network initialized from the source network and regularising the generated representations back towards those of the source task through the use of an adversarial domain classifier. The framework of ADDA is illustrated in figure 2 Domain adaption in ADDA is achieved by optimizing following functions

$$\min_{M_s,C} \mathcal{L}_{cls}(X_s, Y_s) = -\mathbb{E}_{(x_s,y_s)\sim(X_s,Y_s)} \sum_{k=1}^{K} \mathbb{1}_{[k=y_s]} logC(M_s(x_s))$$

$$\min_D \mathcal{L}_{adv_D}(X_s, X_t, M_s, M_t) = -\mathbb{E}_{x_s\sim X_s}[logD(M_s(x_s))] - \mathbb{E}_{x_t\sim X_t}[log(1 - D(M_t(x_t)))]$$

$$\min_{M_s,M_t} \mathcal{L}_{adv_M}(X_s, X_t, D) = -\mathbb{E}_{x_t\sim X_t}[logD(M_t(x_t))]$$

3 Related Work:

Since our overall obejctive is to tranfer of knowledge betwen environments. Transfer learning(TL) is a well established approach of transferring any prior knowledge to a new tasks, domains or environments. TL has widely been used in machine learning problems, most commonly Imagenet weights are used to initialize the network for prior knowledge transfer and fine-tuned with targets. TL was known to solve many issues with low data in target distributions which can ac hive a faster convergence in less training steps [21, 6, 16].

Cad2rl [19] shows models trained for RL simulated environments and d then deployed on new unknown environments.[3] showed a Transfer Learning (TL) based approach to reduce on-board computation required RL navigation problem for a target algorithmic performance. They trained in various simulator environments and meta-weights are then used as initializers to the network in a test environment and fine-tuned for the last few fully connected layers to transfer the domain knowledge to test environments.

But in most of RL problems fine-tuning might not always work, as training and inference have high co-relations. [2] learns a CNN with regressors using supervised learning to drone RL problem to follow a pre-determined path but fails to perform if the environment changes.

Our approach of using adversarial domain adaption to RL problems is closely related to [4] demonstrated how adversarial autoencoder approach can be used for domain adaptation to improve performance on the difficult task of learning to play Atari games.

4 Technical Approach / Methodology / Theoretical Framework

The approach to solve the transfer learning task with Double DQN algorithm [23] combined with ADDA architecture. We use Double DQN algorithm because it was tested to be performing better than DQN for UAV control and navigation, this was shown in [13]

To transfer knowledge from source and target domains this approach utilizes concepts from Adversarial Discriminative Domain Adaptation (ADDA) [22]. This approach seeks to transfer knowledge in this way as a pre-training phase which aligns the representations of a trained agent with that of an initialization network that is trained to efficiently encode the target task. The alignment can be achieved in an unsupervised manner by utilising an ADDA style architecture and is depicted in Figure 2. The initialization network will then be used as the starting weights for training an RL agent using standard deep RL approaches.

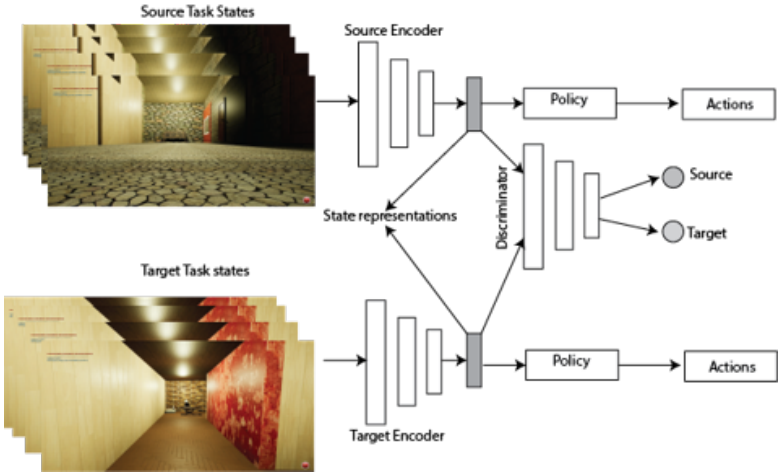


Figure 3: Architecture for domain adaptation, combining ADDA with UAV RL problem. A fully trained source task is depicted on top, target task encoder in the bottom.

Here model comes from the source policy which was trained using the Double DQL algorithm. The architecture of this approach is illustrated in figure 3

4.1 Source training

Since the source network is fixed during adaption. Gradient step on the loss function for source task is as follows:

$$\nabla_{\theta_i} L(\theta_i)) = \mathbb{E}_{s`,a\sim p(.),s`\sim\varepsilon[(R_{t+1}+\gamma \max_{a`} Q(s_{t+1},a`)-Q(s,a,\theta_i))\nabla_{\theta_i} Q(s,a,\theta_i)]}$$

Source agent is trained ahead of time to generate a data-set of state feature representations which can be sampled. A data-set of target task observations is also generated using the random policy.

4.2 Target transfer:

Target state feature representations network is trained by ADDA approach target encoder initially loadded with a pre-trained source state feature enoder model Training procedure after pre-trained model is loaded has two updates per batch:

- The Discriminator is updated to separate source and target domain state feature representations
- Target encoder is updated to fool the Discriminator that the samples states are from source distribution.

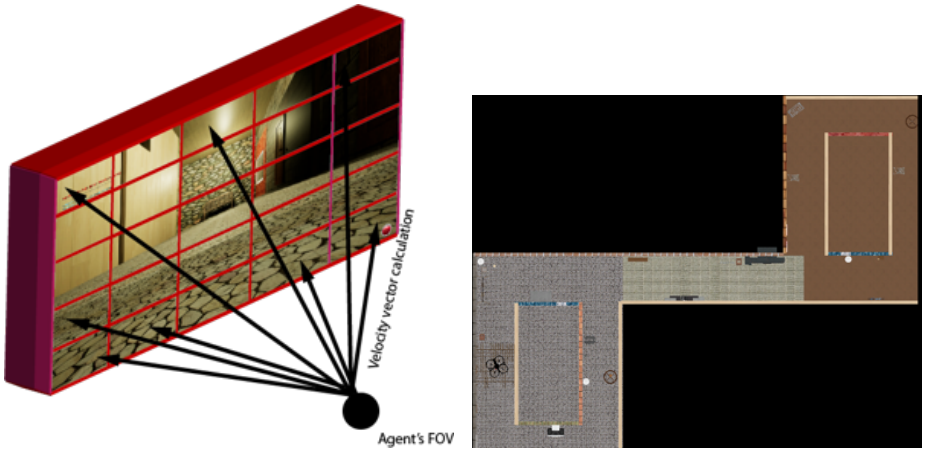
4.3 Target Policy Training:

Finally target model is initialized by the target adapted feature representations network and train it using DDQL. policy and value function learned from scratch. Source task action classifier network is directly used into target network. In this project, value function and policies are learnt from scratch and are not mapped from source and target.

5 Experiments

5.1 Framework Implementations:

Experiments investigate the applicability of Adversarial Domain Adaptation to the UAV reinforcement learning problem. These experiments will be carried out in a simulator called **AirSim** [20] is an open-source plugin for Unreal Engine [7] developed by Microsoft for agents (drones and cars) with physically and visually realistic simulations. In order to interface between Python and the simulated environment, AirSim has Application programming interfaces (APIs) to retrieve data and control vehicles in a platform independent way this also enables to collect training data that can boost source training time. These APIs collect sensor and ground truth data from the drones, as well as various images.



(a) Action Space (b) Indoor environment in 2D

Figure 4: Air Environment in 2D and Action space setting

Following data is received by Airsim environment at every time step:

- v_x, v_y - agent's velocities in x and y directions
- d_x, d_y, d_t agent's distances to goal in x and y and total
- ψ yaw angle relative to initial orientation

- *collided* - collision info

(a) Source Task (b) Target Task

Figure 5: Drone's field of view for source and target tasks in indoor environemts

PEDRA [3] is a programmable engine for Drone Reinforcement Learning (RL) applications. It's proves various environments and interfaces an open source interface to drone reinforcement learning tasks. It is compatible with any numerical computation library, such as TensorFlow [1] and PyTorch [17]. This library has a collection of environments to test RL algorithms with drone. These environments have a shared interface which allows to write general algorithms.

(a) Complex Task: Outdoor environment (b) Target Task

Figure 6: Complex task with 3-D navigation with a goal position to drone

In the selected simulated indoor environment Fig. 4b , a drone agent is equipped with a front-facing camera to capture images, which we use as the state information. The images have dimension 103(height) 103(width) 3(color). Given a state s, the agent chooses from 25 actions to control its flight as shown in Fig. 4a corresponding to the drone controlling its yaw and pitch by various angles. Reward is calculated based on dynamic windowing of the simulated depth map, and is designed to encourage the drone to stay away from obstacles. Most of these setting are default from [3]

We select a indoor environments on the PEDRA platform: indoor updown. Our source task is upper portion of the environment Fig. 5a and target task is Fig. 8a right and left sides of environments in Fig. 4b They contain widely different lighting conditions, wall colors, furniture objects, and hallway structures

We respect to deep learning algorithm implementations proposed framework was implemented in both tensorflow and pytorch. CNN models in tensorflow are 3C2D and Resnet50(Imagenet initialized) in pytorch.

We follow the following procedure to train our entire framework:

1. Train source task agent with DDQN from scratch
2. If model converge or reached acceptable performance, Extract state-action pairs 20k samples
3. In Task 2, run the algorithm with random actions collected states 20k samples
4. Train the ADDA domain adaptation architecture.
5. Load the adapted target encoder and train the target policy network for target task

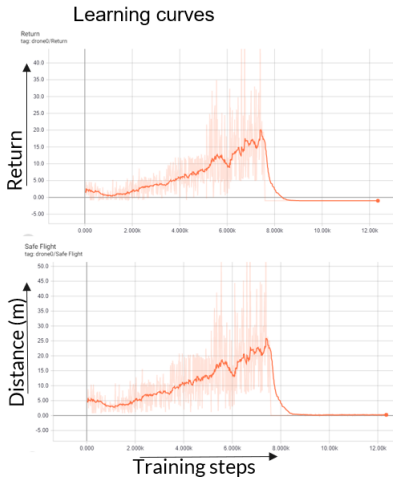
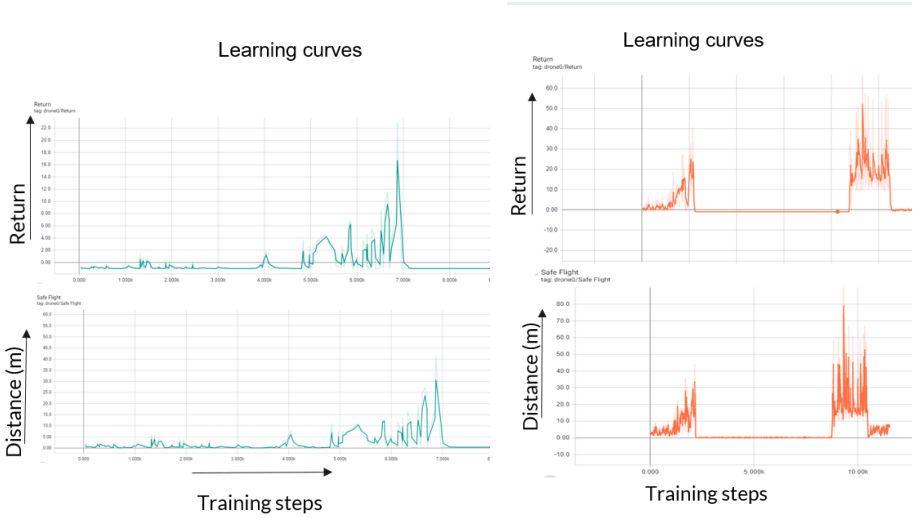


Figure 7: Learning curves for source task returns and distance travelled before collison of each episode

5.2 Results:

We evaluated UAV goal navigation performance in change in environments. Transfer of knowledge occurs from an environments with few variations from source and target such as in one type of indoor setting to another type of indoor settings. Experimentation was carried out on a workstation with a GTX1080Ti GPU.

Initially, a complex navigation task is designed it of moving in a constrained area from Position A to Position B along a road lane in a straight path in an city outdoor environment. The task constraints are if the drone has a collision reward is -100 and resets the episode, Reward varies with distance from position B closer better reward. If the drone leaves a contained 3D cuboid then reward is -10. Results from Fig. 8 indicate that their's no learning with even with huge amount of training. So we simplified our task to basic obstacle avoidance without goal and limiting to yaw and roll movement only without any altitude variation.



(a) Target Task (b) outdoor settings

Figure 8: Learning curves for target task from scratch vs ADDA adapted

We evaluate the performance of the UAV with domain adaption approach and training the target task from scratch. Performance measurement is a plot Rewards vs number of training steps. Safe Flight Distance (SFD) was used to meaning-fully quantify the performance of the learned networks in the respective environment. SFD is the average distance traveled by the agent, in meters, before a collision.

Learning curves from Fig. 7 and Fig. **??** SFD of source task is 80m+(loop) so for source we can observe that agent was able to learn the environment very well and never crashed. Now for agent target task trained from scratch compared with ADDA adapted agent we observe that adapted agent was able to travel significantly more distance and learnt a collision free navigation in fewer training steps compared with than agent trained from scratch.

Thought all the indicated results indicate ADDA performace is much significant from scrach agent, we should further investigate where we are observing the

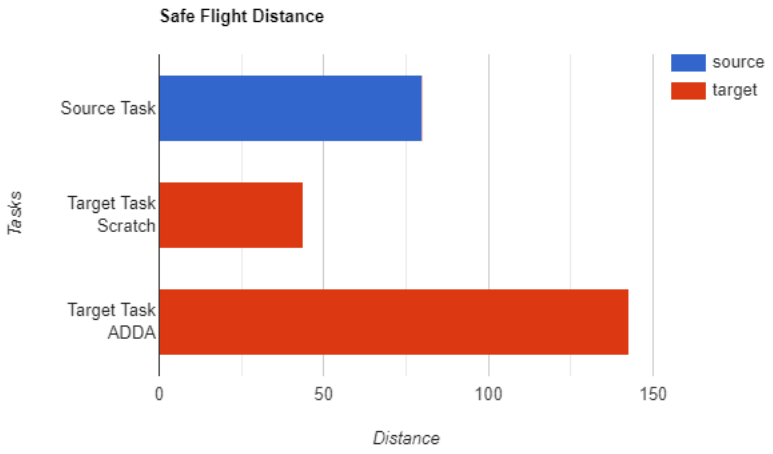


Figure 9: Safe Flight distance for Source task, Target task with trained from scrach and apdapted from ADDA

performance gain by doing an ablation study and doing multiple experiments with random initial positions and harder tasks. We chose source and target distribution discrepancy to be close because training time constrains.

6 Conclusion and Future Work

In this work, we presented an UAV obstacle avoidance using adversarial method for knowledge transfer in reinforcement learning. We demonstrated domain adaptation can be used to transfer the knowledge between tasks and environments. Adapted agent performed significantly better in training time and safe flight distance.

There might be few experiments to interpret if Pretrained source weights vs ADDA performance because ADDA also utilizes source weights. So we further needs to investigate if performance boost we due to ADDA adaption or due to knowledge in pretrained weights.

Future work may also include change action space and reward function in adaption task. Fully explore these methods which can be integrated with multi-task learning or lifelong learning and understand where they fail or succeed

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] Karim Amer, Mohamed Samy, Mahmoud Shaker, and Mohamed ElHelw. Deep convolutional neural network-based autonomous drone navigation. *arXiv preprint arXiv:1905.01657*, 2019.

[3] Aqeel Anwar and Arijit Raychowdhury. Autonomous Navigation via Deep Reinforcement Learning for Resource Constraint Edge Nodes using Transfer Learning. *arXiv e-prints*, page arXiv:1910.05547, Oct 2019.

[4] Thomas Carr, Maria Chli, and George Vogiatzis. Domain adaptation for reinforcement learning on the atari. *arXiv preprint arXiv:1812.07452*, 2018.

[5] Ender Çetin, Cristina Barrado, Guillem Muñoz, Miquel Macias, and Enric Pastor. Drone navigation and avoidance of obstacles through deep reinforcement learning. In *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, pages 1–7. IEEE, 2019.

[6] Felipe Leno Da Silva and Anna Helena Reali Costa. Transfer learning for multiagent reinforcement learning systems.

[7] Epic Games. Unreal engine.

[8] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Advances in neural information processing systems*, 3(06), 2014.

[9] Anna Guerra, Francesco Guidi, Davide Dardari, and Petar M Djurić. Reinforcement learning for uav autonomous navigation, mapping and target detection. In *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pages 1004–1013. IEEE, 2020.

[10] Abhishek Gupta, Coline Devin, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Learning invariant feature spaces to transfer skills with reinforcement learning. *arXiv preprint arXiv:1703.02949*, 2017.

[11] Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018.

[12] Victoria J Hodge, Richard Hawkins, and Rob Alexander. Deep reinforcement learning for drone navigation using sensor data. *Neural Computing and Applications*, pages 1–19, 2020.

[13] Kjell Kersandt, Guillem Muñoz, and Cristina Barrado. Self-training by reinforcement learning for full-autonomous drones of the future. In *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*, pages 1–10. IEEE, 2018.

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

[15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[16] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[17] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[18] Alejandro Rodriguez-Ramos, Carlos Sampedro, Hriday Bavle, Paloma De La Puente, and Pascual Campoy. A deep reinforcement learning strategy for uav autonomous landing on a moving platform. *Journal of Intelligent & Robotic Systems*, 93(1-2):351–366, 2019.

[19] Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

[20] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics*, pages 621–635. Springer, 2018.

[21] Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(7), 2009.

[22] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7167–7176, 2017.

[23] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *arXiv preprint arXiv:1509.06461*, 2015.

[24] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003, 2016.

[25] Fuzhen Zhuang, Xiaohu Cheng, Ping Luo, Sinno Jialin Pan, and Qing He. Supervised representation learning: Transfer learning with deep autoencoders. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.