Air Learning: An End to End Learning Gym for Aerial Robots

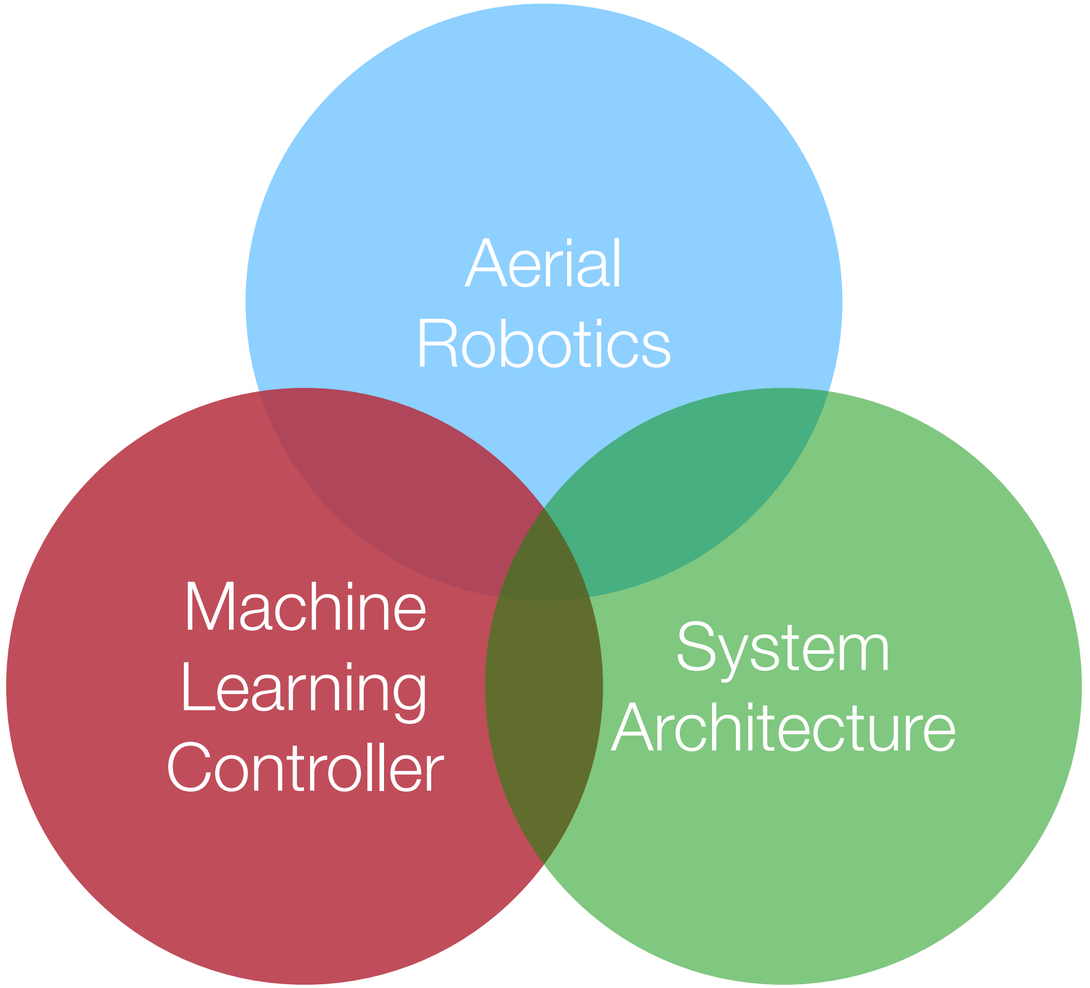
Srivatsan Krishnan[1], Colby Banbury[1], Bardienus Duisterhof[2,1], Aleksandra Faust[3], Vijay Janapa Reddi[1]

1- Harvard University
2- TU Delft
3- Google Research

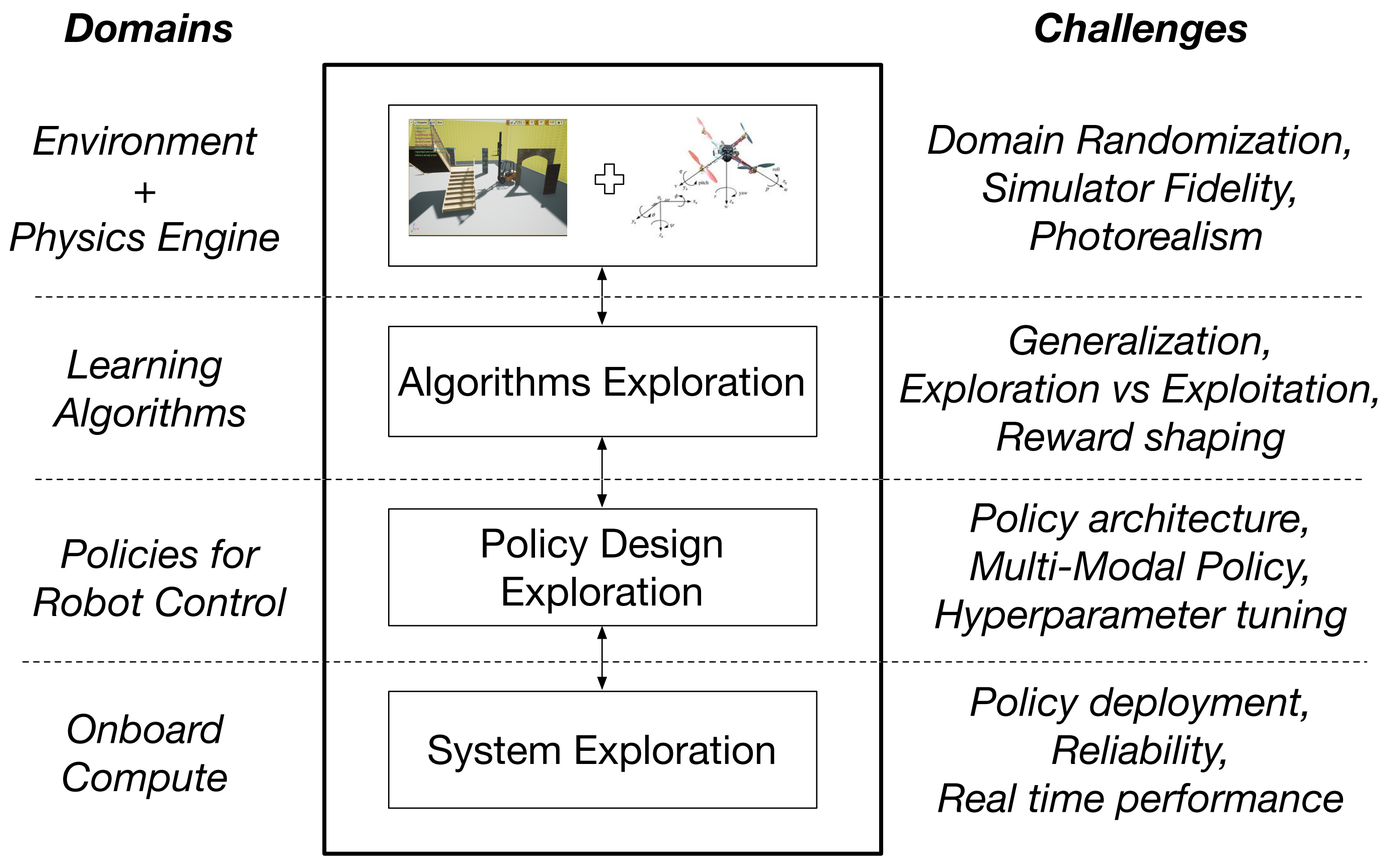
Scan me:

Harvard John A. Paulson School of Engineering and Applied Sciences

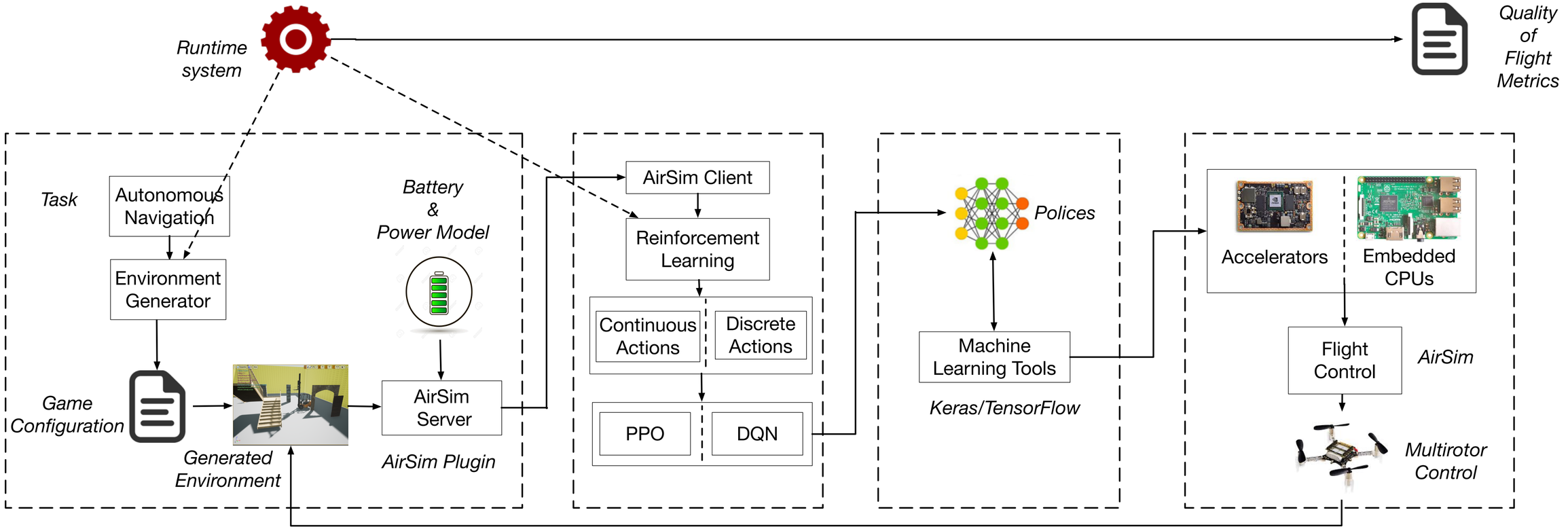
Interdisciplinary Nature of Aerial Robot Learning



Challenges with Cross-Domain



Air Learning Gym Infrastructure



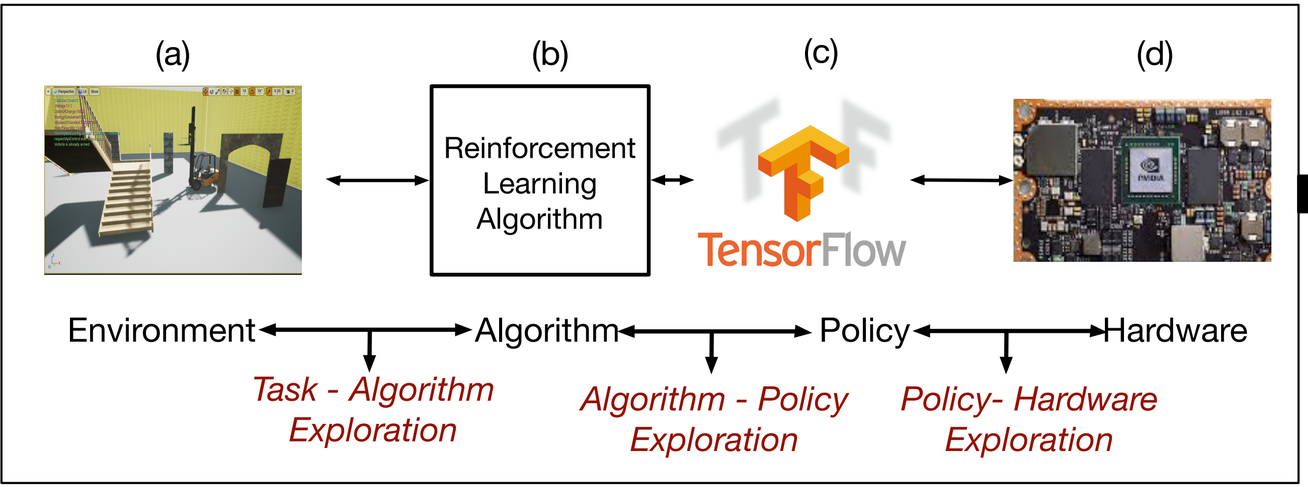
Task Definition and Environment Generation ↔ **Algorithm Exploration** ↔ **Policy Design Exploration** ↔ **Hardware Exploration**

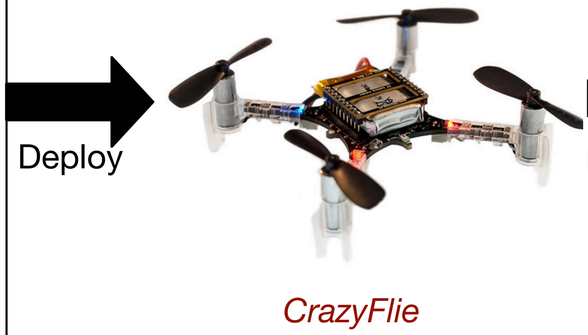
Task - Algorithm Exploration

Algorithm - Policy Exploration

Policy - Hardware Exploration

Demonstration





Demonstration video:https://www.youtube.com/watch?v=cvO5YOzI0mg

Source code/project: https://github.com/harvard-edge/airlearning

AIR LEARNING: AN END-TO-END LEARNING GYM FOR AERIAL ROBOTS

Srivatsan Krishnan [1] **Colby Banbury** [1] **Bardienus Duisterhof** [2] **Aleksandra Faust** [3] **Vijay Janapa Reddi** [1]
https://github.com/harvard-edge/airlearning

ABSTRACT

Air Learning is an interdisciplinary, open-source research infrastructure that aims to soften the boundaries between aerial robotics, machine learning, controls, and systems architecture. It provides the tooling and various components necessary for developing an end-to-end learning-based application for aerial robotics starting from simulation to deployment on a real aerial robot. By having all the key components tightly integrated, we hope researchers can use this tool to develop novel solutions to several open problems in these domains. We also hope researchers can use it to explore and understand various trade-offs of their solution due to cross-domain interactions between algorithms and system. Also since the infrastructure is open-source, the research community can add new features thus modifying it according to their own requirements and use cases.

1 AIR LEARNING

Air Learning research infrastructure facilitates the holistic study of end-to-end learning algorithms for aerial robotics. Autonomous aerial robot design involves navigating the design space across various boundaries spanning from modeling the environment down to the choice of onboard computer platform available in the robot. Air Learning infrastructure aims to provide researchers with a cross-domain infrastructure that allows them to holistically study and evaluate reinforcement learning algorithms for autonomous aerial machines. Air Learning is open sourced and the source code can be found online at http://bit.ly/2JNAVb6. Here is an demonstration of practical application built using Air Learning infrastructure: https://bit.ly/2sEdmdK

There are four key components to Air Learning:

Air Learning Environment Generator: Availability of high-quality data is essential for learning algorithms. Air Learning provides high fidelity photorealistic UE4 based random and parametrizable environment generator (Figure 1 (a)). The AirSim plugin is integrated within the project which models the drone physics and provides different sensing modalities. The Air Learning environment generator is exposed as an OpenAI gym environment for the easy development of reinforcement learning algorithms. The list of parameters that can be configurable in the current version of Air Learning is listed in Table 1. Here is a video link that demonstrates some of the features of environment generator: https://bit.ly/2R4Fxfq

Air Learning RL: The environment generator is exposed as an OpenAI gym interface to facilitate the development of reinforcement learning algorithms. Stable-baselines (Hill et al., 2018), an OpenAI baseline fork, is integrated into Air Learning which allows researchers to use high-quality implementation of state of the art reinforcement learning algorithms (Figure 1(b)). This integration allows researchers to evaluate different RL algorithms for aerial robots with few lines of code change. The current version of Air Learning is seeded with DQN (Mnih et al., 2013), PPO (Schulman et al., 2017), DDPG (Lillicrap et al., 2015), and SAC (Haarnoja et al., 2018). Here is a video link that demonstrate the environment generation part with reinforcement learning training: https://bit.ly/2Nyzxcs

Machine Learning Backend: Policy architecture plays a crucial role in the performance of the reinforcement learning algorithm. Air Learning uses Tensorflow/Keras backend for evaluating different policy architectures. This allows the researchers to use a wide variety of sensing modalities provided by AirSim to explore multimodal policies.

"Hardware-in-the-Loop" Evaluation: Air Learning enables the "Hardware-in-the-loop" approach that allows researchers to experiment and evaluate how their algorithm will perform when onboard compute platform changes. "Hardware-in-the-loop" methodology can also help system architects to benchmark or even design efficient systems targetted for aerial robots. Here is a video demonstration where the RL policy is evaluated in Ras-Pi3 real-time as if the aerial robot has Ras-Pi3 as the onboard compute platform: https://bit.ly/2NDRjex

RL Serving/Deployment: Air Learning also provides necessary tooling and methodology flows for deploying the trained RL policy on a real aerial robot. Once ported, the researchers can perform a flight test to determine how the

[1]Harvard University [2]TU Delft [3]Robotics at Google. Correspondence to: Srivatsan Krishnan <srivatsan@seas.harvard.edu>.

Proceedings of the 2nd SysML Conference, Austin, TX, USA, 2020.

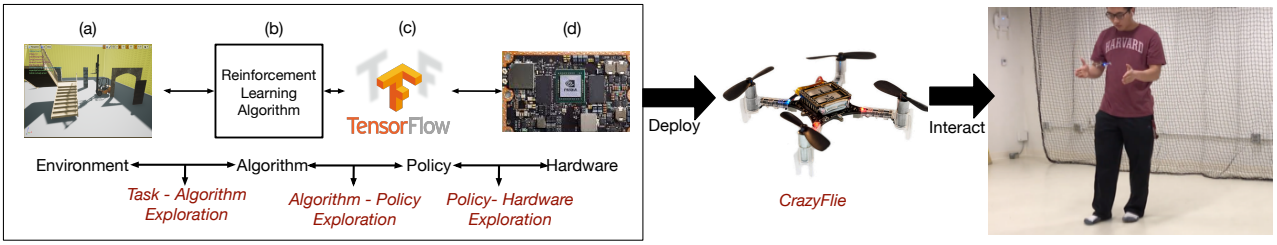


Figure 1. Demonstration of Air Learning infrastructure. For the demonstration, we train a DQN algorithm for dynamic obstacle avoidance. We explore a tiny DQN policy to fit into the memory available in the crazyyflie. Once we determine the best policy for the crazyflie hardware, we verify the functionality of the policy in the Air Learning environment. After verification of the functionality in the simulator, we deploy the learned policy on a real Crazyflie drone to see if it avoids dynamic obstacles. The attendees will get to interact with the Crazyflie similar to the demonstrator in the video: `https://bit.ly/2TAaVnG`

RL policy performs in the real world. Currently, our flow is validated on CrazyFlie nano-drone platform.

2 Live Interactive Demo

Demo Application: We propose an interactive demonstration where we take a simple obstacle avoidance task for a resource-constrained nano-drone and show how various components in the Air Learning infrastructure can be used to create a domain randomized environment for the task, train a RL algorithm, test, and deploy them on a real nano-UAVs.

Through the avoidance task, the attendees will get to experience two scenarios in our demo: (1) explore the simulation infrastructure components in Air Learning. (2) explore the deployment of the learned RL policy onto a CrazyFlie.

Air Learning Simulation Infrastructure: In this part of the demo, the attendees will get to experience the following: (1) demonstration of domain randomization and various parameters available in Air Learning environment generator as listed below in Table 1 and (2) training an aerial robotics task with different reinforcement learning algorithms.

Parameter	Format	Description
Arena Size	`[length, width, height]`	Spawns an arena of "length" x "width" x "height".
Wall Colors	`[R, G, B]`	Wall colors in [Red, Green, Blue] color format.
Asset	`<folder name>`	Allow any UE4 asset to be imported.
# Static Obstacles	Scalar Integer	Static obstacles in the arena.
# Dynamic Obstacles	Scalar Integer	Dynamic obstacle in the arena.
Seed	Scalar Integer	Seed value used in randomization.
Minimum Distance	Scalar Integer	Min. distance between arena obstacles.
Goal Position	`[X, Y, Z]`	X, Y and Z goal coordinates.
Velocity	$[V_{max}, V_{min}]$	Dynamic obstacle velocity.
Materials	`<folder name>`	UE4 material assigned to UE4 asset.
Textures	`<folder name>`	UE4 Texture assigned to UE4 asset.

Table 1. List of configurations available in environment generator.

RL Deployment: In this section of demo, attendees can flash a pre-trained RL policy trained using Air Learning infrastructure onto a CrazyFlie nano-drone. Once the RL policy is flashed the attendee will get to fly the drone and witness how it can avoid humans in close proximity.

3 Equipments

The presenter will supply a laptop with HDMI interface. For the simulation component, the presenters will demo the content through the laptop. The presenter will also carry a couple of CrazyFlie and its accessories for the audience to experience the demo. A pre-trained RL model will be flashed by the presenter/audience to showcase how the Air Learning platform can be used for developing RL (end-to-end learning algorithms), train, test, and finally deploy on a Crazyflie nano-drones.

References

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Hill, A., Raffin, A., Ernestus, M., Gleave, A., Kanervisto, A., Traore, R., Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., and Wu, Y. Stable baselines. `https://github.com/hill-a/stable-baselines`, 2018.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.