Deep Reinforcement Learning for Biomimetic Touch: Learning to Type Braille

Alex Church *
University of Bristol
ac14293@bristol.ac.uk

John Lloyd
University of Bristol
jl15313@bristol.ac.uk

Raia Hadsell
Google Deepmind
raia@google.com

Nathan F. Lepora
University of Bristol
n.lepora@bristol.ac.uk

Abstract

Tactile robotics and reinforcement learning are two areas that are concerned with interaction between an agent and an environment yet research concerning this overlap has not reached significant milestones. Here we present a challenging tactile robotics environment, learning to type on a braille keyboard, for use with deep reinforcement learning algorithms. Preliminary results show that successful learning can take place directly on a physical robot equipped with a biomimetic tactile sensor (the TacTip), based on the human fingertip.

1 Introduction

Reinforcement Learning (RL) is an algorithm that learns through interaction with an environment and touch is the primary sense that humans and animals use to interact with their environment. Modern Deep Reinforcement Learning (DRL) algorithms allow for more general learning from high dimensional sensory input, yet most research has focused on vision sensors, while touch has received comparatively low levels of interest. This trend continues when DRL is applied to physical robots where most research uses either proprioception or vision to make up an observation. The two fields of reinforcement learning and tactile sensing overlap but the combination has yet to reach significant milestones. Here we will focus on connecting the two fields in the context of a human-relevant task.

The field of deep reinforcement learning has seen rapid progress in part due to benchmark suites allowing new ideas to be directly compared to previous work. The majority of these have focused on simulated environments such as the Arcade Learning Environment [1], continuous control environments [2, 3] and simulated, robotics-focused environments [4, 5]. There have been some calls to establish physical robot benchmarking suites [6], however due to the cost of robotics equipment they are difficult for the field to adopt. In the narrower field of tactile robotics there are no available benchmarks to evaluate and compare new ideas. In this paper we propose a challenging tactile robotics environment which is intended to serve as a tool for experimentation and fine-tuning of DRL algorithms being applied to tactile robotics. The environment consists of a keyboard with braille keycaps, combined with a controllable robotic arm equipped with a tactile sensor. The primary task in this environment involves learning to type a key or sequence of keys, and has a number of interesting attributes: it is goal driven, contains both continuous and discrete action spaces, uses sparse rewards, and will likely be challenging in terms of exploration. All of these features help to make the task representative of other robotics and tactile robotics challenges. This environment also requires minimal human intervention and is relatively fast to run, ideal properties for RL algorithms.

*AC, JL and NL are with the Dept of Engineering Mathematics and Bristol Robotics Lab, Bristol, U.K.

NeurIPS 2019 Workshop on Robot Learning: Control and Interaction in the Real World, Vancouver, Canada

In this study a challenging problem is that the physical properties of the artificial tactile sensors are difficult to simulate; this motivates us to train directly on a physical robot. There are a variety of DRL algorithms in common use [7, 8, 9, 10, 11, 12], but the majority of these are not applicable for learning on the physical robot, either because they suffer from poor sample efficiency, brittleness with respect to hyper-parameters, or are only applicable to discrete action spaces (whereas our task requires both discrete and continuous actions). There are currently two algorithms that meet the criteria for learning directly on a physical robot, SAC [13] and MPO [14], both of which offer good sample efficiency, robustness to hyper parameter selection and work with either continuous or discrete action spaces. Whilst the simulated results of MPO appear to be the strongest, SAC has seen the most following research and a slightly revised version has shown evidence of successful training when directly applied to physical robots even whilst using image observations [15].

2 Biomimetic Tactile Sensing

The TacTip (Figure 1) [16] is a low-cost, robust, 3D-printed optical tactile sensor based on a human fingertip. The human sense of touch corresponds with the deformation of the dermal and epidermal layers of the skin which is detected and relayed through mechanoreceptors [17]. The design of the TacTip builds upon research that showed that the Merkel Cell Complex (MCC) of sensory receptors works in tandem with the morphology of intermediate ridges. This biological structure is mimicked in the TacTip, by replacing intermediate ridges with internal pins, attached to a skin-like membrane. The detection of deformation in the artificial skin is captured as movement in these pins via an internal camera in place of the MCC. The camera view of this sensor design is shown in Figure 1b. Adaptive thresholding is used to preprocess the raw tactile image to a binary image, this makes the effects of deformation of the sensor more apparent (see Figure 1c).

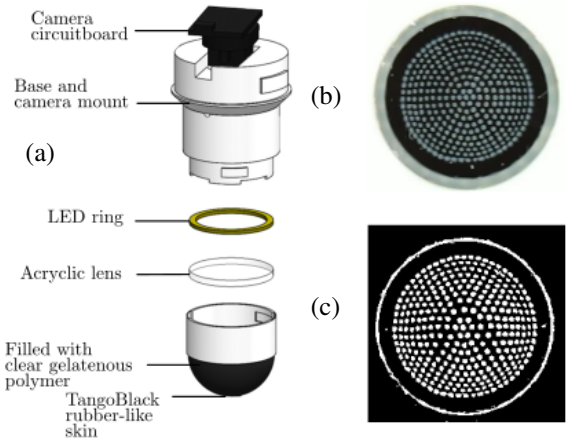

Figure 1: (a) Exploded view of the TacTip sensor design. (b) Camera view of the 331 pins in the dense, 25mm diameter sensor design. (c) Thresholded image from the sensor as it is pressed onto an 'UP' arrow key from the braille keyboard described below.

Most of the initial work and some of the most impressive results in DRL research have used images as the observational spaces [18, 15]. This allows for some of the advances in deep learning for computer vision to be leveraged for use in DRL algorithms. The TacTip offers a high-dimensional artificial tactile sense in the form of images. On top of this, the design of the TacTip transfers the tactile images into a more simplistic form allowing for networks to be relatively small yet capture a significant amount of tactile information. The design of the TacTip has also been demonstrated to cope well with continuous interaction with an environment including tapping and sliding on edges for hours at a time. The application of supervised deep learning to the images obtained from the TacTip has seen recent success, with a single sensor shown to achieve highly robust contour following [19] and multiple sensors, incorporated onto a robotic hand, proposed for grasp success prediction and item classification [20].

3 Braille Learning Task

The primary task is to successfully press a target, or goal key that is randomly selected per episode from a subset of keys on a braille keyboard. A positive reward of $+1$ is given for successfully pressing a goal key, a negative reward of -1 is given for pressing an incorrect key. Currently there is no negative reward associated with movement of the sensor however use of a small penalty per action

could encourage more efficient typing. Successful learning of this task will result in an average episodic return of 1 or very near.

In order to successfully complete a task the DRL algorithms will need to be able to interpret a button before pressing it. To help with this, a keyboard with relatively stiff key switch has been used. We use the DREVO Excalibur 84 Key Mechanical Keyboard with Cherry MX Black switches. These switches have a linear pressing profile and require 60cN of force and 2mm of travel before actuation occurs. On top of this Cherry keys have good build quality making them consistent across keys. The keycaps shown in Figures 2 have been designed to attach to Cherry MX switches.

The tactile keyboard environment can accommodate several different tasks of varying difficulty, a list of proposed tasks can be found in the appendix (Table 1). Due to the positioning on the keyboard the arrow keys can be separated from alphabet (including space) keys. As there are fewer arrow keys and they have a symmetric arrangement the tasks using only arrow keys should be easier than those using the full alphabet keys. Additionally as the state space is significantly smaller for the arrow tasks it should encounter positive rewards during random exploration; however, learning a task that covers the full set of alphabet keys will likely require an approach such as Hindsight Experience Replay (HER) [21] to improve data efficiency. When using the discrete action space the sensor automatically taps a key after each movement. This tapping movement will not activate the button but will ensure there is a tactile image where the sensor is under some deformation.

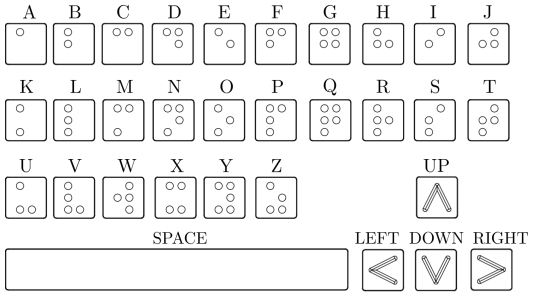

Figure 2: Braille alphabet used for the tactile keyboard environment. Space consists of no tactile information and the arrows allow for a separate, smaller and easier challenge.

Incorporating continuous control is needed to make this task more representative of other tactile robotics tasks. However, giving the robot full continuous control may result in damage occurring to the sensor or keyboard. To aid this we restrict $\Delta x, \Delta y$ actions to movement in a plane above the keyboard, the Δz action then becomes a prediction of tap depth. To aid performance we add a constant to the predicted tap depth, similarly to the discrete case this will ensure there is a tactile image where the sensor is under some deformation.

4 Initial Test with Supervised Learning

An initial step in this paper is to ensure that the braille keys can be interpreted on the TacTip sensor whilst the keys are attached to a keyboard and without exerting enough force to activate the button. To test this we performed a supervised learning task of classifying all keys shown in Figure 2. There were 100 samples collected per key for training and 50 sample collected per key for validation, resulting in 3100 training images and 1550 validation images. Each tactile image was collected at the bottom of a tapping motion. The sensor was positioned 3.5mm above the centre of each key, the tapping motion consists of moving downwards 5mm, this avoids pressing the key passed its 2mm actuation point. A random z variation sampled from the interval $(-1, 0)$mm was used; this moves the position of the sensor away from the keys and ranges from barely touching to nearly actuating the button. A similar random x, y perturbation was sampled from $(-1, +1)$mm and a random orientation perturbation was sampled from $(-10°, 10°)$ to add some variation in the collected data.

Adaptive thresholding is used on the input images to create a binary image with focus directly onto the pins. Data augmentation including random shifting and zooming of the images are used along with early stopping and a decaying learning rate. The network used in this challenge is a relatively simple convolutional neural network that contains 5 convolutional layers and 2 fully connected layers. Batch-normalization is used on the convolutional layers after the activation and dropout is used on the fully connected layers. A near perfect overall accuracy of 99% is achieved demonstrating that the braille is interpretable on the tactile sensor without actuating a button. This shows that using DRL to perform tasks in this environment should be possible. As well as this, the trained supervised networks can be used for initialisation of the networks used in DRL.

5 Preliminary Results with Deep Reinforcement Learning

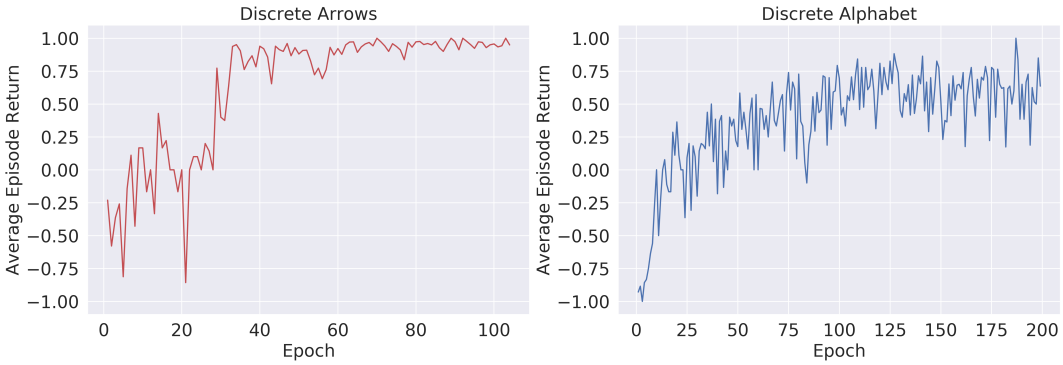

Figure 3: Training curves showing the average episodic return for the discrete arrow task (left) and discrete alphabet task (right). Each epoch corresponds to 250 steps in an environment.

The initial attempt at learning this problem focused on the simpler, discrete action setting. The states in this task consist of a grid layout, where the sensor can be positioned over the centre of each key and some additional empty spaces. The arrow keys are arranged as shown in Figure 2 with empty spaces on the left and right of the UP key. The observations consist of a $100 \times 100 \times 1$ image captured with the TacTip sensor, a threshold is then applied to create a binary image. To boost performance this image is passed through the pre-trained convolutional stages of the network used in the supervised classification task, these weights have been frozen and are not updated during training. This results in a 144 dimensional latent space that contains enough information for interpreting the images. As this task uses a discrete action space, discrete DRL algorithms are viable options. Here we use a modified version of the standard DQN algorithm with both the dueling [22] architecture and double Q learning improvement [23]. A one-hot encoding of both the target key and the previous action are concatenated onto the flattened convolutional output, two 512 node fully connected layers are then attached. These are used to predict a separate state value $V(s)$ and state dependent action advantages $Q(s, a)$, used both in training the algorithm and for deciding which action to take.

For the arrow task the algorithm is able to successfully complete the task, consistently achieving an average episodic return of 1 or near. Figure 3 demonstrates that this successful learning occurs at around 40 epochs (10000 steps/2 hours of training). For the alphabet case the challenge is more difficult and therefore more epochs are necessary for training. After 200 epochs (50000 steps/10 hours of training) near successful training has occurred. Observing the robot at test time shows that near optimal policy has been successfully learnt. With deterministic actions incorrect key presses are avoided however the policy will sometimes be averse to the press action and can result in the sensor hovering above a goal key without activating it. It is likely that this will be resolved with more training time or hyper parameter tuning.

6 Conclusion

Here we have presented a challenging new environment designed for the fine tuning and optimising of deep reinforcement learning algorithms for their application to tactile robotics challenges. The environment contains features such as continuous control and sparse rewards that help to make it representative of other tactile robotics tasks. There are several proposed tasks using this environment, each with varying levels of difficulty. Preliminary results show that successful training can occur directly on a physical robot within a relatively short time period. One of the techniques used that allow for this fast successful training is the pre-training and freezing of convolutional weights used in the Q network. This significantly reduces the complexity of the network whilst capturing the necessary information from an image observation. Whilst supervised learning has been used to pre-train these convolutional weights other solutions are viable; variational auto-encoders and random features are other techniques that warrant further exploration.

References

[1] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research*, 7 2012.

[2] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking Deep Reinforcement Learning for Continuous Control. *International Conference on Machine Learning (pp. 1329-1338)*, 4 2016.

[3] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. DeepMind Control Suite. *arXiv preprint arXiv:1801.00690*, 1 2018.

[4] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, Vikash Kumar, and Wojciech Zaremba. Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research. *arXiv preprint arXiv:1802.09464*, 2 2018.

[5] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 6 2016.

[6] A. Rupam Mahmood, Dmytro Korenkevych, Gautham Vasan, William Ma, and James Bergstra. Benchmarking Reinforcement Learning Algorithms on Real-World Robots. *arXiv preprint arXiv:1809.07731*, 9 2018.

[7] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust Region Policy Optimization. *International conference on machine learning (pp. 1889-1897)*, 2 2015.

[8] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 7 2017.

[9] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. *International conference on machine learning (pp. 1928-1937)*, 2 2016.

[10] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining Improvements in Deep Reinforcement Learning. *Thirty-Second AAAI Conference on Artificial Intelligence*, 10 2017.

[11] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 9 2015.

[12] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing Function Approximation Error in Actor-Critic Methods. *arXiv preprint arXiv:1802.09477*, 2 2018.

[13] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *arXiv preprint arXiv:1801.01290*, 1 2018.

[14] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a Posteriori Policy Optimisation. *arXiv preprint arXiv:1806.06920*, 6 2018.

[15] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic Algorithms and Applications. *arXiv preprint arXiv:1812.05905*, 12 2018.

[16] Benjamin Ward-Cherrier, Nicholas Pestell, Luke Cramphorn, Benjamin Winstone, Maria Elena Giannaccini, Jonathan Rossiter, and Nathan F. Lepora. The TacTip Family: Soft Optical Tactile Sensors with 3D-Printed Biomimetic Morphologies. *Soft Robotics*, 5(2):216–227, 4 2018.

[17] Roland S. Johansson and J. Randall Flanagan. Coding and use of tactile signals from the fingertips in object manipulation tasks. *Nature Reviews Neuroscience*, 10(5):345–359, 5 2009.

[18] 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*, 12 2013.

[19] Nathan F. Lepora, Alex Church, Conrad De Kerckhove, Raia Hadsell, and John Lloyd. From pixels to percepts: Highly robust edge perception and contour following using deep learning and an optical biomimetic tactile sensor. *IEEE Robotics and Automation Letters*, 12 2018.

[20] Alex Church, Jasper James, Luke Cramphorn, and Nathan Lepora. Tactile Model O: Fabrication and testing of a 3d-printed, three-fingered tactile robot hand. *arXiv preprint arXiv:1907.07535*, 7 2019.

[21] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight Experience Replay, 2017.

[22] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. Dueling Network Architectures for Deep Reinforcement Learning. *arXiv preprint arXiv:1511.06581*, 11 2015.

[23] Hado van Hasselt, Arthur Guez, and David Silver. Deep Reinforcement Learning with Double Q-Learning. *Thirtieth AAAI Conference on Artificial Intelligence*, 3 2016.

7 Table of Proposed Tasks used for the Braille Keyboard Environment

Table 1: List of proposed tasks to be learned in the tactile keyboard environment and their expected difficulty. Difficulty levels range from 1-4 with 1 being easiest, 4 being hardest.

Name	**Difficulty**	**Buttons Used**	**Action Space**	**Action List**
Disc-Arrow	1	Arrows	Discrete	{`UP, DOWN, LEFT, RIGHT, PRESS`}
Disc-Alpha	2	Alphabet	Discrete	{`UP, DOWN, LEFT, RIGHT, PRESS`}
Cont-Arrows	3	Arrows	Continuous	$\{\Delta x,\ \Delta y,\ \Delta z\}$
Cont-Alpha	4	Alphabet	Continuous	$\{\Delta x,\ \Delta y,\ \Delta z\}$