Deep Reinforcement Learning for Tactile Robotics: Learning to Type on a Braille Keyboard

Alex Church, John Lloyd, Raia Hadsell, and Nathan F. Lepora

***Abstract*—Artificial touch would seem well-suited for Reinforcement Learning (RL), since both paradigms rely on interaction with an environment. Here we propose a new environment and set of tasks to encourage development of tactile reinforcement learning: learning to type on a braille keyboard. Four tasks are proposed, progressing in difficulty from arrow to alphabet keys and from discrete to continuous actions. A simulated counterpart is also constructed by sampling tactile data from the physical environment. Using state-of-the-art deep RL algorithms, we show that all of these tasks can be successfully learnt in simulation, and 3 out of 4 tasks can be learned on the real robot. A lack of sample efficiency currently makes the continuous alphabet task impractical on the robot. To the best of our knowledge, this work presents the first demonstration of successfully training deep RL agents in the real world using observations that exclusively consist of tactile images. To aid future research utilising this environment, the code for this project has been released along with designs of the braille keycaps for 3D printing and a guide for recreating the experiments.**

***Index Terms*—Force and tactile sensing, reinforecment learning, biomimetics.**

I. Introduction

TOUCH is the primary sense that humans use when interacting with their environment. Deep Reinforcement Learning (DRL) algorithms enable learning from interactions and support end-to-end learning from high-dimensional sensory input to low-level actions. However, most research has focused on vision, while a scarcity of data, sensors and problem domains relating to touch has relegated tactile research to a minor role. This trend continues even when DRL is applied to robots interacting physically with their environment, where most research uses proprioception or vision. Here we attempt to bridge the gap by positioning tactile DRL research in the context of a human-relevant task: learning to type on a braille keyboard.

Manuscript received February 24, 2020; accepted June 26, 2020. Date of publication July 20, 2020; date of current version August 5, 2020. This letter was recommended for publication by Associate Editor J. Falco and Editor D. Popa upon evaluation of the reviewers' comments. This work was supported in part by an Award from the Leverhulme Trust on 'A biomimetic forebrain for robot touch' (RL-2016-39) and in part by an EPSRC CASE Award to AC sponsored by Google DeepMind. *(Corresponding author: Alex Church.)*

Alex Church, John Lloyd, and Nathan F. Lepora are with the Department of Engineering Mathematics and Bristol Robotics Laboratory, University of Bristol, Bristol BS8 1UB, U.K. (e-mail: ac14293@bristol.ac.uk; jl15313@bristol.ac.uk; n.lepora@bristol.ac.uk).

Raia Hadsell is with the Google DeepMind, London N1C 4AG, U.K. (e-mail: raia@google.com).

This letter has supplementary downloadable material available at https://ieeexplore.ieee.org, provided by the authors.

Digital Object Identifier 10.1109/LRA.2020.3010461

The field of DRL has seen rapid progress, which in part is due to benchmark suites that allow new methods to be directly compared. These are mainly simulated environments, such as the Arcade Learning Environment [1], continuous control environments [2], [3] and simulated robot-focused environments [4], [5]. More recently, several robot benchmarking suites have been established [6]–[10] with the intention of moving the field towards physically-interactive environments.

In the area of tactile robotics, there are currently no benchmarks for evaluating and comparing new methods. In this letter, we propose a challenging tactile robotics environment which we intend will serve as a tool for experimentation and fine-tuning of DRL algorithms. Furthermore, the environment challenges the capabilities of the tactile sensor in requiring sufficient sensitivity and spatial resolution and could be used to draw comparisons between different tactile sensors. The environment consists of a keyboard with 3D-printed braille keycaps, combined with a robotic arm equipped with a tactile sensor as its end effector. The primary task in this environment involves learning to type a key or sequence of keys, and has several useful attributes: it is goal driven, uses sparse rewards, contains both continuous and discrete action settings, is challenging in terms of exploration and requires a tactile sensor with both a high spatial resolution and high sensitivity. All of these aspects help to make the task representative of other robotics and tactile robotics challenges. The environment has also been designed to need minimal human intervention and be reasonably fast, necessary characteristics for real-world applications of DRL algorithms. In addition, the components required for creating this environment are either widely available or 3D printed, which allows easier adoption in the tactile robotics community.

This letter makes the following contributions. We define 4 tasks in the braille keyboard environment that progress in difficulty from arrow to alphabet keys and from discrete to continuous actions. In addition to the physical environment, we construct a simulation based on exhaustively sampling tactile data, and use this for initial validation of DRL, including agents trained with Deep Q Learning [11], Twin Delayed Deep Deterministic Policy Gradients (TD3) [12] and Soft Actor Critic (SAC) [13]. We then demonstrate that for the majority of these tasks, DRL can be used to successfully train agents exclusively from tactile observations. However, a lack of sample efficiency for the most challenging task makes it impractical to train learning agents in the physical environment. This work thus leaves open the question of whether new or untested methods can improve this sample efficiency to a point where physical

training is feasible, or whether alternative tactile sensors can represent the required tactile information in a concise enough form to allow for better sample efficiency in training.

II. Related Work

A common approach for applying DRL to robotics is to train on accurate simulations of the robot, then transfer those learned policies to the real world, bridging what is known as the 'reality gap' [14], [15]. However, a serious issue for this approach is that the physical properties of artificial tactile sensors are highly challenging to simulate, as evidenced by a lack of use in the field compared to e.g. simulated visual environments. This issue is compounded as the task complexity increases, particularly with dynamic environments. This necessitates solutions that are capable of being trained directly on a real robot to make progress on DRL for tactile sensing applied to physical interaction.

DRL has been applied to object manipulation, particularly with dexterous robotic hands. However, the most successful examples of learning in this area do not utilise tactile data [15]–[17], but instead deploy simulations to accelerate learning. In these cases, observations are often made up of joint angles and object positions/orientations; in practise, obtaining this information from real-world scenarios has required complicated visual systems. In their work, OpenAI state that they specifically avoid using sensors built into the physical hand, such as tactile sensors, as they are subject to "state-dependent noise that would have been difficult to model in the simulator" [15].

Most prior research applying RL to tactile sensing has used the SynTouch Biotac, a taxel-based tactile sensor the size of a human fingertip. Van Hoof *et al.* [18] demonstrated that tactile data can be used as direct input to a policy network, trained through RL, for a manipulation task on a physical robot. In [19] dimensional reduction is used to simplify the tactile input, resulting in successful regrasping of an object utilising tactile data. In [20], tactile and proprioceptive data are combined to achieve the task of gentle object manipulation, where exploration is improved by using tactile data to form both a penalty signal and as a target for intrinsic motivation.

Recent work has combined tactile image observations from the Gelsight optical tactile sensor with proprioceptive information to use DRL to learn surface following [21]. Optical tactile sensors provide tactile information in a form that is well-matched to modern deep learning techniques that have been honed on computer vision challenges. Given the recent successes of optical tactile sensors with deep learning [22]–[27], the combination of deep reinforcement learning and optical tactile sensing appears a promising avenue for future research.

III. Hardware

A. Custom Biomimetic Tactile Fingertip

The BRL tactile fingertip (TacTip) [28] is a low-cost, robust, 3D-printed optical tactile sensor based on a human fingertip. As the current task involves interpreting braille keycaps, we require a tactile sensor with high spatial resolution that makes contact with an area that can lie inside a standard keycap.

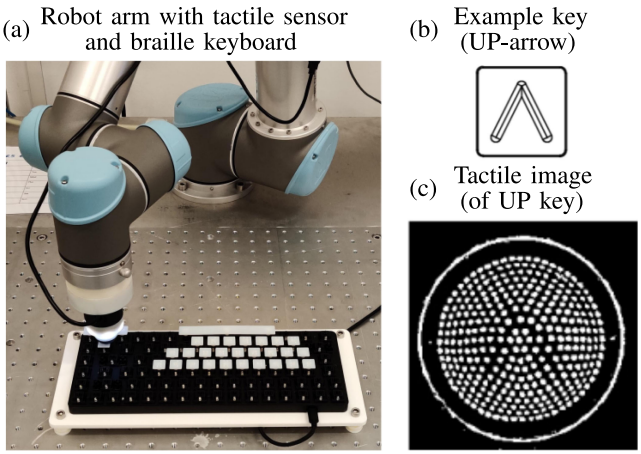

Fig. 1. Robot arm with tactile fingertip (panel a) pressing a 3D-printed UP-arrow key (panel b) resulting in a tactile image (panel c). Notice how the UP-key shape is visible in the spacing of the pins in the tactile imprint.

Conventionally, the TacTip has 127 pins on a 40 mm-diameter tactile dome [28], which is too large for this task. The sensor tip was thus redesigned for this work to fit 331 pins of radius 0.625 mm onto a 25 mm-diameter dome of depth 7.2 mm. The pins are arranged such that there is a sub-mm space between them. An example tactile image obtained from this modified sensor is shown in Fig. 1(c), where adaptive thresholding is used to process the raw tactile image into a binary image that makes the deformation more apparent and mitigates any changes of lighting inside the sensor.

Following recent work with this tactile sensor [22], the pre-processed image captured by the sensor is directly passed into the machine learning algorithms. This removes the need for pin detection and tracking algorithms that were necessary in previous work, and enables the advantages of convolutional neural networks to be applied to tactile data.

B. Braille Keyboard

For the tasks considered here, we chose a good-quality keyboard with a fairly stiff key switch: the DREVO Excalibur 84 Key Mechanical Keyboard with Cherry MX Black switches. These switches have a linear pressing profile and require 0.6 N of force and 2 mm of travel before actuation occurs. Furthermore, Cherry keys have good build quality which improves consistency across keys. The dimensions of our 3D-printed keycaps are shown in Fig. 3 and the full set of keys used throughout this work are shown in Fig. 2. The keyboard was chosen in order to allow the tactile robot to make exploratory contacts with a key before pressing it.

C. Robotic Arm

For these experiments we use a Universal Robotics UR5 robotic arm and its in-built inverse kinematics to allow for Cartesian control. This makes the environment relatively agnostic to the type of robotic arm used, with variation only arising subject to error accumulating from the lack of precision in the inverse kinematics controller. Other than the robotic arm, the

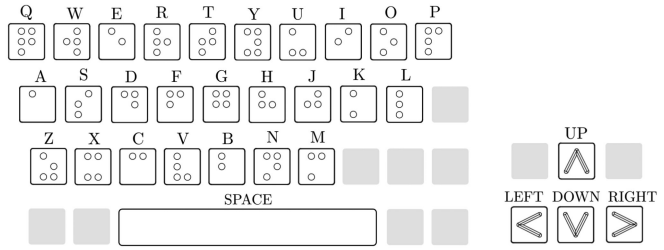

Fig. 2. Braille alphabet used for the tactile keyboard environment. The space bar is a blank character, which will cause deformation of the tactile sensor. Grey squares indicate positions where no deformation of the sensor will occur. A simpler task can be defined using only the 4 arrow keys.

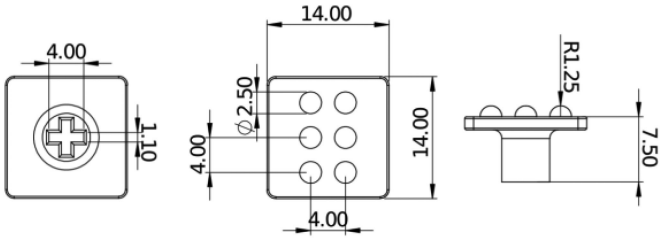

Fig. 3. Dimensions of the braille keycaps in mm, designed for cherry MX mechanical switches.

environment consists of components that are either 3D printed or widely available.

IV. Benchmark Tasks

Within this tactile keyboard environment, we propose 4 tasks of progressive difficulty that have the same underlying principles:

- Discrete actions: Arrow keys.
- Discrete actions: Alphabet keys.
- Continuous actions: Arrow keys.
- Continuous actions: Alphabet keys.

The distinction between arrow and alphabet keys is shown in Fig. 2. Note that the space key is included in alphabet tasks and is represented by a blank character. Continuous actions constitute a positional change in the x, y axis and a tap depth in the z axis, where the sensor performs a downward movement and returns to a predefined height. This is performed sequentially to reduce the potential for damage occurring. An illustration of successful training of a RL agent is depicted in Fig. 4 over the alphabet keys for both discrete and continuous actions.

The proposed task for this environment is to successfully press a goal key, which is randomly selected each episode from a subset of keys on a braille keyboard, and to do so without actuating any other keys. The start location is randomised for each episode. Hence, the agent must first determine its location, then navigate to the target position and finally press the target key. Positions of all the keys are learnt implicitly within the weights of a neural network. A positive reward of $+1$ is given for successfully pressing a goal key; an incorrect key press results in episode termination and a reward of 0. Successful learning of this task will result in an average episodic return close to 1, dependent on the exploration parameters of the RL agent.

The arrow keys offer a natural way to define an easier task, since they are located away from the other keys, are fewer, and span a smaller area, which greatly reduces the size of the state space. An agent should therefore encounter more positive rewards during random exploration. Moreover, the tactile features on the arrow keys are more distinct, easing their interpretation from tactile data.

The alphabet (plus space) keys give a far more challenging task. As the state space is relatively large, learning over the full set of alphabet keys requires an approach such as Hindsight Experience Replay (HER) [29] to improve data efficiency. Also, depending on the position of the sensor, some keys can be indistinguishable from each other, giving subtleties from perceptual aliasing. This feature also makes it harder to learn continuous action policies, since when using discrete actions the sensor can be arranged to be always located above the centre of a key.

V. Initial Assessment Using Supervised Deep Learning

For a first step, we check that the braille keys can be interpreted by the tactile sensor without exerting enough force to activate the button. To test this, we used supervised learning to classify all keys on the braille keyboard (shown in Fig. 2). Overall, we used 100 samples per key for training and 50 samples per key for validation, resulting in 3100 training images and 1550 validation images.

To make the task more representative of how humans perform key presses, we introduced small random perturbations in the action. Each tactile image was collected at the bottom of a tapping motion, where the sensor was positioned $3.5 + \Delta z$ mm above the centre of each key, then moved downwards 5 mm, finishing above the 2 mm activation point for a key press. A random variation Δz sampled from the interval $(-1, 0)$ mm was used to represent uncertainty in the key height, which ranges from 'barely touching' to 'nearly activating' the button. A similar random horizontal (x, y)-perturbation was sampled from ranges $(-1, +1)$ mm and a random orientation perturbation was sampled from $(-10^\circ, 10^\circ)$ to add further variation in the collected data.

Tactile images were cropped and resized down to 100×100 pixels (from 640×480 pixels), then adaptive thresholding was used to binarize the image; in addition to reducing image storage, this also helps accentuate tactile features and mitigate any issues from changes in internal lighting.

The network used in this task follows the architecture used for learning to play Atari games, originally proposed in [11]. The same network architecture is used for all reinforcement learning tasks (details given in Table I). For supervised learning, we perform data augmentation including random shifting and zooming of the images. We also use early stopping, a decaying learning rate, batch normalization on the convolutional layers after the activation and dropout on the fully connected layers.

A near perfect overall accuracy of 99% was achieved, demonstrating that the braille can be interpreted by the tactile sensor without activating the button, even when there is significant uncertainty about how the key is pressed.

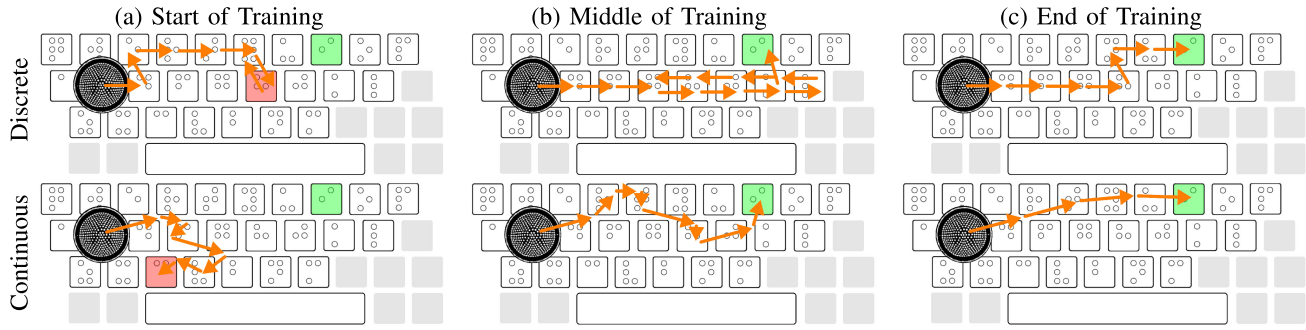

Fig. 4. Visualisation of both the discrete action (top) and continuous action (bottom) tasks. The sensor is initialised in a random position with the task of pressing a randomly initialised goal key (green). Orange arrows indicate the actions taken at each step. Initially, the policy networks cause random actions that results in an incorrect key press (red). After some training, actions that lead to a correct key press are found but may be sub-optimal, often with cyclic movements. Eventually, policies should converge to a near-optimal path between the initial position and goal key. For the discrete case, actions are given by a DQN agent trained for 0, 30, and 100 epochs. The continuous case is an illustrative depiction of successful training.

TABLE I
DRL AND NETWORK HYPERPARAMETERS

	Param	Shared		
Network	Input dim	[100, 100, 1]		
	Conv filters	[32, 64, 64]		
	Kernel widths	[8, 4, 3]		
	Strides	[4, 2, 1]		
	Pooling	None		
	Dense layers	[512,]		
	Output dim	num actions		
	Activation	*ReLU*		
	Initialiser	variance scaling (scale=1.0)		
Control	Epoch steps	250		
	Replay size	10^5		
	Update freq	1		
	n/ Updates	2		
	Batch size	32		
	Start steps	200		
	Max ep len	25		
	Optimizer	Adam		
		DQN	**SAC**	**TD3**
RL	Discount (γ)	0.95	0.6 (disc) 0.95 (cont)	0.95
	Polyak (τ)	0.005	0.995	0.995
	LR (η)	5×10^{-4}	5×10^{-4}	5×10^{-5}
	Alpha LR (η_α)	n/a	1×10^{-3}	n/a
Explore	Initial ϵ	1.0	n/a	1.0
	Final ϵ	0.1	n/a	0.05
	ϵ Decay steps	2000	n/a	n/a
	Target entropy	n/a	0.139 (disc) -6 (cont)	n/a

VI. SIMULATED ENVIRONMENT

Even though our focus in this work is DRL on a physical tactile robot, it is useful to have a simulated task that is similar to the physical environment yet runs much faster. While the policies trained in simulation may not be directly transferable to the physical environment, the parameters found should guide successful training of policies on the physical robot.

In the simulation of the discrete task, actions lead to locations on a virtual keyboard with a known label. The label is then used to retrieve a tactile image of the same label from the data collected for the initial assessment of supervised learning (Section VI). Thus the simulation accurately matches the physical environment, where the label of each key may not be known but a tactile image observation can be collected that will resemble the one obtained from the simulated environment. In practice, this simulated discrete environment is more complicated than the physically interactive environment, because of the perturbations introduced in the data collection that we choose not to introduce during reinforcement learning on the physical robot.

The simulation of the continuous task is more difficult to represent, because we do not have labels for every location on the keyboard. To approximate the physical environment, we collect tactile observations over a dense grid of locations spanning the keyboard, with those location stored (using 3 mm intervals over the x, y-dimensions and 1 mm intervals over z). Along with the tactile image observations, we also record whether a key has been activated or not. During simulation, the position of the virtual sensor is known, which allows for a tactile image to be sampled from the collected data with the minimum Euclidean distance from the virtual sensor position.

In both cases, the simulated environment ignores effects such as error that accumulates over long sequences of robot arm moves, sensor drift, changing lighting conditions, movement of the keyboard and any other information that is not represented during the data collection stage. However, whilst these effects can occur on a real robot, the simulated tasks still offer useful information for hyper-parameter tuning and initial validation of RL algorithms.

VII. TACTILE REINFORCEMENT LEARNING

A. Reinforcement Learning Formulation

To define the problem, we represent the proposed tasks in a standard Markov Decision Process (MDP) formulation for reinforcement learning. An agent interacts with an environment at discrete time steps t, consisting of a state s, action a, resulting reward r, resulting state s' and terminal signal d. Actions a are sampled from policy distribution $\pi(a|s)$ represented by a neural network.

States s are represented as a combination of a tactile image observation o from the sensor, goal g and previous action a_{t-1}. Goals g are selected randomly per episode and represented as a one-hot array, where the 'hot' value represents the class label of a target key. The previous action a_{t-1} is required to avoid an agent

becoming stuck in cyclic patterns when no tactile deformation is present on the sensor. Both are concatenated into the policy networks after any convolutional layers. Reward r is sparse and binary, with $r = 1$ when a correct button is actuated, otherwise $r = 0$. Activation of any button, correct or incorrect, results in a terminal signal $d = 1$ that resets the environment with the sensor moving to a new random location on the keyboard.

In the discrete tasks, actions are selected from the set $\mathcal{A} =$ {`UP, DOWN, LEFT, RIGHT, PRESS`}. For each movement action the tactile sensor is re-positioned above the centre of a neighbouring key where a tap action (which does not activate a key) is performed and a tactile image for the next state s' is gathered. The `PRESS` action moves the sensor in the z axis by -8 mm to activate a key.

In the continuous tasks, each action is selected from $\mathcal{A} = \{\Delta x,\ \Delta y,\ \Delta z\}$ where Δz corresponds to a tapping depth. For practical reasons, actions are bound to a finite range dependent on task. For the arrow task, the enforced Δx and Δy bounds are between ± 10 mm and for the alphabet task between ± 20 mm. In both the alphabet and arrow tasks, Δz is bound to the range [2,8] mm to ensure safe key actuation.

B. Reinforcement Learning Algorithms

There are several popular DRL algorithms in common use [11], [12], [30]–[34]. However, there are two major problems holding back the application of DRL to physical robotics: poor sample efficiency and brittleness with respect to hyper-parameters. Generally, on-policy methods such as Trust Region Policy Optimisation (TRPO) and Proximal Policy Optimisation (PPO) sacrifice sample efficiency to gain stability and robustness to hyper-parameters, making them difficult to apply to physical robots. Since its introduction, Deep Q-Learning (DQN) has had various improvements to address these issues, some of which are amalgamated in RAINBOW [35]. Off-policy entropy-regularised reinforcement learning has also separately addressed these issues with Soft Actor Critic (SAC) [33] and Maximum a Posterior Policy Optimisation (MPO) [34]. Both of these offer good sample efficiency, robustness to hyper-parameter selection and work with either continuous or discrete action spaces. SAC has led to the most follow-up research and a modified version has demonstrated successful training when applied to physical robots [13]. Twin Delayed Deep Deterministic Policy Gradients (TD3) [12] was developed concurrently to SAC and offers similar performance.

For discrete-action tasks, we use DQN with the double [36] and dueling [37] improvements, and SAC adapted for discrete actions. For continuous-action tasks, we use TD3 and SAC; since SAC is applicable to both types of action space, some hyper-parameters are transferable over all tasks.

Another barrier to the application of DRL to robotics is when environments require challenging exploration combined with sparse rewards, as convergence is then very slow. A common technique commonly used to alleviate this is to use dense and shaped rewards [8], [10], [13], [38]. However, this can require domain-specific knowledge and bias learning algorithms towards sub-optimal policies. Hindsight Experience Replay (HER) helps address this problem by replaying episodes stored in a buffer while varying the goal from what was initially intended. Here we find HER significantly improves performance and sample efficiency for all considered tasks.

Several other adjustments were also made to optimise performance. For all algorithms except TD3, the ratio of optimisation steps per environment step was increased to $2:1$ to improve efficiency because of the greater cost of environment versus optimisation steps; however, this hindered the stability of TD3. Additionally for both DQN and TD3 we linearly decayed the exploration parameter ϵ for an initial number of exploration steps. For SAC and TD3, polyak averaging was used for target networks, with a coefficient of $\tau = 0.995$, although we did find that SAC could also learn well with hard target updates.

VIII. Evaluation Metrics

To quantify the performance when training an agent on this benchmark, we introduce several evaluation metrics particular to this task. These can be used to compare results of this benchmark when using alternative algorithms or alternative sensors. The evaluation metrics introduced are:

- **Convergence Epoch:** Measures the first epoch in which 95% of the maximum performance across the full training run is achieved and serves as an indication of sample efficiency during training. This is subject to some noise and works best when results are averaged over multiple seeds or when test results are averaged over a relatively high number of episodes (>10).
- **Typing Accuracy:** Measures the accuracy when typing example sentences or sequences of keys. As a reward of 1 is given for correct key presses this metric is directly correlated with the average return.
- **Typing Efficiency:** Measures the number of steps taken to press a key and gives an indication as to whether the policies learnt are near optimal.

When measuring typing accuracy and efficiency for the arrow task all 24 permutations of {`UP, DOWN, LEFT, RIGHT`} are evaluated. For the alphabet task, we provide 10 randomly generated permutations of the alphabet. This tests that a mapping between all keys is stored implicitly within the weights of the policy neural networks. During evaluation, the sensor is initialised in a random starting position for each sequence and the sensor position is not randomly reset after episode termination when a single key is pressed unlike during training. Furthermore, during evaluation we used deterministic actions to avoid mispressed keys from unnecessary exploration. This procedure gives a more consistent value in comparison with using average episodic return, which allows better comparisons to be drawn.

IX. Results

A. Results for Discrete Tasks

For the discrete arrow key and alphabet key tasks, we find that both DQN and Discrete SAC rapidly converge to accurate performance in simulation and on the physical robot. Testing

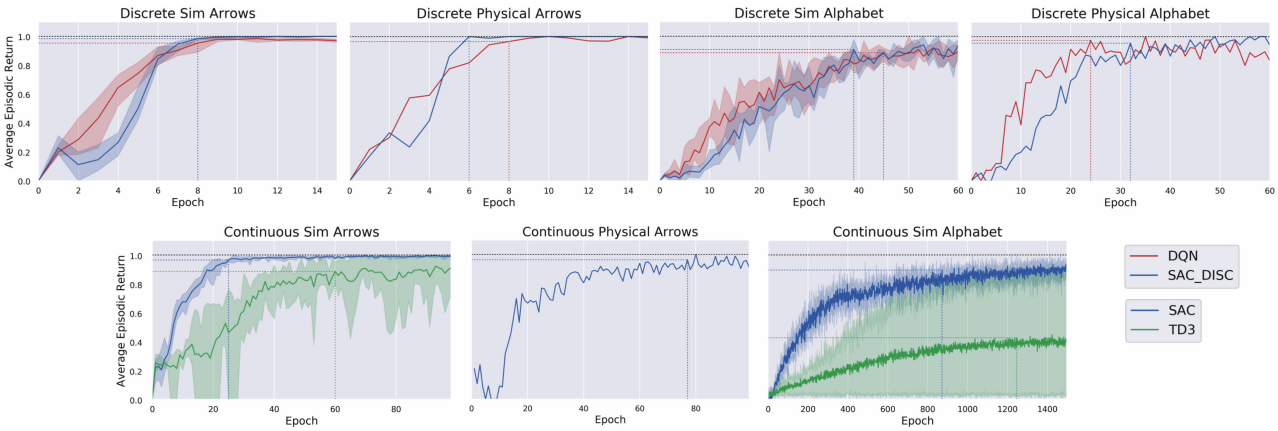

Fig. 5. Training curves showing the average episodic return for all tasks, discrete task results are shown on the top row, continuous task results are shown on the bottom row. Each epoch corresponds to 250 steps in the environment. For the simulated tasks results are averaged over 5 seeds, the physical tasks show results for 1 seed. Coloured dashed lines represent the first point at which episodic return reaches 95% of the maximum reached throughout training.

over multiple seeds in simulation shows that this learning is stable and consistently converges to an average episodic reward of near 1. The training curves for the discrete tasks for both the arrow and alphabet key environments (Fig. 5) show that the task can be learned in all cases.

For the arrow task, asymptotic performance is achieved within 15 epochs, which is approximately equal to 1 hour of training time on the physical robot. For the alphabet task, convergence is longer, with asymptotic performance achieved within 60 epochs, or approximately 4 hours of training time on the physical robot. Similar sample efficiency and final performance is found for both discrete SAC and DQN across all trained agents. Whilst final performance appears slightly higher for discrete SAC, this can be explained by the target entropy causing lower levels of exploration when the agent nears convergence. Decaying the exploration parameter (ϵ) to a lower value during training for the DQN agents, or evaluating with deterministic actions, results in similar final performance to discrete SAC.

B. Results for Continuous Tasks

The continuous control tasks are far more challenging due to their larger state spaces. For the arrow task in simulation, we find SAC is still able to achieve an asymptotic performance of near 1.0 within 50 epochs. TD3 is also able to achieve performance of near 1.0 within 100 epochs; however, training is less stable and is not robust to hyper-parameter changes. Due to this instability we do not evaluate TD3 in the physical environment. The training curves are shown in Fig. 5 (bottom panels) for all continuous tasks that we were able to run to completion.

For the continuous arrow task, continuous control is not as well represented in simulation, this is shown by the left and middle panels in Fig. 5 (bottom) not being accurate matches. We find convergence to a slightly lower average episodic return of 0.95 after about 76 epochs compared to 25 epochs in simulation.

The alphabet task again offers an increase in task complexity. For a single-seeded run in simulation, we achieved a final average episodic return of ~ 0.9 for the continuous SAC algorithm and ~ 0.8 for TD3. This final performance takes significantly longer to achieve than all other tasks, with convergence occurring around 872 epochs for SAC and 1246 epochs for TD3. This will correspond to approximately 60 hours of physical robotic training time and is currently not feasible given laboratory operating constraints.

TABLE II
RESULTS FOR TRAINED AGENT EVALUATION

	Task	Algorithm	Steps	Accuracy	Convergence Epoch
Simulation	Disc Arrow	DQN	230	1.0	8
		SAC_DISC	234	1.0	8
	Disc Alpha	DQN	1722	0.981	45
		SAC_DISC	1649	0.940	39
	Cont Arrow	TD3	140	0.906	60
		SAC	133	1.0	25
	Cont Alpha	TD3	1169	0.811	1246
		SAC	1193	0.933	872
Physical	Disc Arrow	DQN	246	1.0	8
		SAC_DISC	246	1.0	6
	Disc Alpha	DQN	1722	0.992	24
		SAC_DISC	1649	0.985	32
	Cont Arrows	SAC	364	0.938	76

C. Performance on Evaluation Metrics

An overview of the results across all algorithms used within this work are given in Table II. These can be used as a comparative point for future work using this benchmark. For discrete tasks, we find that high typing accuracy is possible across both simulated and physical environments. In evaluation, both SAC_DISC and DQN have comparable results for typing accuracy and typing efficiency. In simulation, continuous tasks display a large reduction in the number of steps taken to activate a goal key due to the more efficient path across the keyboard that the sensor can take. However, this comes at the cost of a reduction in accuracy. These more-efficient policies are not observed in the physical task, likely due to more variety in tactile observations causing less certainty in the selected actions. In the simulated

continuous alphabet task, TD3 gave high performance. However, these results were difficult to achieve consistently, making TD3 impractical for application to the physical robot.

X. Discussion

This work proposes a benchmarking tool aimed at developing the area of reinforcement learning for tactile robotics. Four tasks of varying difficulty are proposed, together with a representative simulated environment to allow for easier exploration of algorithmic adjustments. Evaluation metrics are also provided to allow for a quantitative comparison when using this benchmark with alternative algorithms or alternative sensors.

We evaluate the performance of several learning-based agents on the proposed benchmark tasks, both in simulation and on the physical robot. We demonstrate that successful learning is possible across all tasks within simulation and across 3 of 4 tasks on the physical robot. Currently, training the physical agent with continuous actions for the full alphabet task has not been achieved within the time allowed by operating constraints in our laboratory. Some example techniques that have not yet been implemented include using prioritised experience replay [39] to improve efficiency, and scheduling the ratio of optimisation steps per environment step throughout learning.

During development, we found some techniques were crucial to achieving successful learning on a physical robot. For example, HER gave sample efficiency improvements up to a factor of 10, along with boosting final performance. This efficiency was particularly evident on the alphabet tasks where rewards are less frequently encountered under random exploration. Training a DQN agent on the simulated discrete alphabet task with HER achieved final performance of ~ 0.98 within 50 epochs; comparatively, this same task without HER achieved a similar level of performance after ~ 500 epochs. Thus, HER was required for feasible learning of the tasks on a physical system. When using RL to solve physical robotics problems, designing the tasks to be goal orientated can allow for more general behaviour to be learnt whilst taking advantage of the benefits HER gives.

We also found that learnt optimal policies were sensitive to factors other than the algorithm hyper-parameters. For example, when using large action ranges on the continuous arrow tasks, the policies tended towards large movements in the direction of a goal key with low dependence on the current tactile observation. This behaviour arises because a relatively high average return can be found with this method alone, which causes agents to become stuck in a local optimum. Reducing the action ranges to lower values minimised this effect because the sensor was less likely to jump directly to the correct key. Thus, the average return from following this sub-optimal policy was reduced, which ultimately resulted in better policies.

Another useful technique was to create a simulation that is partially representative of the final task. Even with recent advancements, DRL is notoriously sensitive and brittle with respect to hyper-parameters, and so a fast, simulated environment helped find parameter regions that allowed for successful and stable training. For example, TD3 was so sensitive that a bad starting seed could cause minimal learning of the task. If attempting to find stable hyper-parameter regions on the physical task, multiple-seeded runs took hours or days of lab operation time. Therefore, where possible, simulating a simplified version of the problem provides valuable information for the physical task.

The braille task is designed to be representative of a multitude of tactile robotics tasks in which it may not be practically feasible to create a simulated environment via exhaustive sampling. Therefore, a main aim of this study was to achieve training from scratch in the physical environment. That said, in some circumstances, it would be interesting to explore the benefits of transfer learning from simulation to reality. For a preliminary investigation in simulation, we attempted to capture an important aspect of switching from a simulation to the physical robot, by artificially increasing the step size of the data in the continuous alphabet task from 3 mm to 6 mm (mimicking that the 3 mm intervals are a discrete approximation of the continuous physical environment). We found that learning on the higher density task could be accelerated by transferring policies trained on the lower density task. Whilst this is not entirely aligned to the problem of transfer learning between simulation and reality, these preliminary results demonstrate the potential for large sample-efficiency improvements. However, there are multiple methods of transferring trained policies from simulation to reality, opening up an avenue of future work that uses this platform to examine these sim-to-real approaches.

Previous research on DRL and tactile data has either used taxel-based sensors [18]–[20], [40], or has combined optical tactile images with proprioceptive information [21]. To the best of our knowledge, this work presents the first demonstration of successfully training DRL agents in the real world, with observations that exclusively comprise high-dimensional tactile images. Though this work has only presented an evaluation of the TacTip sensor, the proposed experiments could offer valuable comparisons between alternative tactile sensors, particularly in the context of applications using recent DRL techniques. It is possible that alternative tactile sensors could represent the tactile information in a more concise form that allows for more sample-efficient learning, which is the most limiting factor found during this work and an interesting topic for future investigation. To aid with future work, the code used for this project has been released along with designs of the braille keycaps for 3D printing and a guide for recreating experiments and evaluating trained agents (available at https://github.com/ac-93/braille_RL).

References

[1] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *J. Artif. Intell. Res.*, vol. 47, pp. 253–279, 2013.
[2] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *Proc. Int. Conf. Mach. Learn.*, Apr. 2016, pp. 1329–1338.
[3] Y. Tassa *et al.*, "DeepMind control suite," Jan. 2018, *arXiv:1801.00690*.
[4] M. Plappert *et al.*, "Multi-goal reinforcement learning: Challenging robotics environments and request for research," Feb. 2018, *arXiv:1802.09464*.
[5] G. Brockman *et al.*, "OpenAI gym," Jun. 2016, *arXiv:1606.01540*.
[6] M. Ahn *et al.*, "ROBEL: Robotics benchmarks for learning with low-cost robots," in *Proc. 3rd Annual Conf. Robot Learn.*, 2019, vol. 100, pp. 1300–1313.

[7] A. Kumar, T. Buckley, Q. Wang, A. Kavelaars, and I. Kuzovkin, "OffWorld gym: Open-access physical robotics environment for real-world reinforcement learning benchmark and research," Oct. 2019, *arXiv:1910.08639*.
[8] B. Yang, J. Zhang, V. Pong, S. Levine, and D. Jayaraman, "REPLAB: A reproducible low-cost arm benchmark platform for robotic learning," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 8691–8697.
[9] A. R. Mahmood, D. Korenkevych, B. J. Komer, and J. Bergstra, "Setting up a reinforcement learning task with a real-world robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Mar. 2018, pp. 4635–4640.
[10] A. R. Mahmood, D. Korenkevych, G. Vasan, W. Ma, and J. Bergstra, "Benchmarking reinforcement learning algorithms on real-world robots," in *Proc. 2nd Annu. Conf. Robot Learn.*, 2018, vol. 87, pp. 561–591.
[11] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.
[12] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in Actor-Critic methods," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1582–1591.
[13] T. Haarnoja *et al.*, "Soft Actor-Critic algorithms and applications," Dec. 2018, *arXiv:1812.05905*.
[14] S. James *et al.*, "Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit.*, Dec. 2018, vol. 2019, pp. 12 619–12 629.
[15] OpenAI *et al.*, "Learning dexterous in-hand manipulation," *Int. J. Robot. Res.*, vol. 39, no. 1, pp. 3–20, Aug. 2018.
[16] A. Rajeswaran *et al.*, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," *Robot.: Sci. Syst. XIV*, 2018.
[17] OpenAI *et al.*, "Solving Rubik's cube with a robot hand," Oct. 2019, *arXiv:1910.07113*.
[18] H. van Hoof, N. Chen, M. Karl, P. van der Smagt, and J. Peters, "Stable reinforcement learning with autoencoders for tactile and visual data," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2016, pp. 3928–3934.
[19] Y. Chebotar, K. Hausman, Z. Su, G. S. Sukhatme, and S. Schaal, "Self-supervised regrasping using spatio-temporal tactile features and reinforcement learning," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2016, vol. 2016, pp. 1960–1966.
[20] S. H. Huang *et al.*, "Learning gentle object manipulation with curiosity-driven deep reinforcement learning," Mar. 2019, *arXiv:1903.08542*.
[21] C. Lu, J. Wang, and S. Luo, "Surface following using deep reinforcement learning and a GelSightTactile sensor," Dec. 2019, *arXiv:1912.00745*.
[22] N. F. Lepora, A. Church, C. De Kerckhove, R. Hadsell, and J. Lloyd, "From pixels to percepts: Highly robust edge perception and contour following using deep learning and an optical biomimetic tactile sensor," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 2101–2107, Apr. 2019.
[23] W. Yuan, C. Zhu, A. Owens, M. A. Srinivasan, and E. H. Adelson, "Shape-independent hardness estimation using deep learning and a gelsight tactile sensor," *Sensors*, vol. 17, no. 12, Apr. 2017, Art. no. 2762.
[24] R. Calandra *et al.*, "The feeling of success: Does touch sensing help predict grasp outcomes?" in *Proc. 1st Annual Conf. Robot Learn.*, Nov. 13–15, 2017, vol. 78, pp. 314–323.
[25] R. Calandra *et al.*, "More than a feeling: Learning to grasp and regrasp using vision and touch," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3300–3307, May 2018.
[26] F. R. Hogan, M. Bauza, O. Canal, E. Donlon, and A. Rodriguez, "Tactile regrasp: Grasp adjustments via simulated tactile transformations," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Mar. 2018, pp. 2963–2970.
[27] N. F. Lepora and J. Lloyd, "Optimal deep learning for robot touch: Training accurate pose models of 3D surfaces and edges," *IEEE Robot. Autom. Mag.*, vol. 27, no. 2, pp. 66–77, 2020, doi: 10.1109/MRA.2020.2979658.
[28] B. Ward-Cherrier *et al.*, "The TacTip family: Soft optical tactile sensors with 3D-printed biomimetic morphologies," *Soft Robot.*, vol. 5, no. 2, pp. 216–227, Apr. 2018.
[29] M. Andrychowicz *et al.*, "Hindsight experience replay," in *Proc. Annual Conf. Neural Inf. Process. Syst.*, 2017, pp. 5048–5058.
[30] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," in *Proc. Int. Conf. Mach. Learn.*, Feb. 2015, pp. 1889–1897.
[31] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," Jul. 2017, *arXiv:1707.06347*.
[32] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *Proc. 4th Int. Conf. Learn. Representations*, 2016.
[33] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.
[34] A. Abdolmaleki, J. T. Springenberg, Y. Tassa, R. Munos, N. Heess, and M. Riedmiller, "Maximum a posteriori policy optimisation," in *Proc. 6th Int. Conf. Learn. Representations*, 2018.
[35] M. Hessel *et al.*, "Rainbow: Combining improvements in deep reinforcement learning," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 3215–3222.
[36] H. v. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 2094–2100.
[37] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, "Dueling network architectures for deep reinforcement learning," in *Proc. 33nd Int. Conf. Mach. Learn.*, 2016, vol. 48, pp. 1995–2003.
[38] I. Popov *et al.*, "Data-efficient deep reinforcement learning for dexterous manipulation," Apr. 2017, *arXiv:1704.03073*.
[39] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *Proc. 4th Int. Conf. Learn. Representations*, 2016.
[40] B. Wu, I. Akinola, J. Varley, and P. K. Allen, "MAT: Multi-fingered adaptive tactile grasping via deep reinforcement learning," in *Proc. 3rd Annu. Conf. Robot Learn.*, 2019, vol. 100, pp. 142–161.