Learning Deep Parameterized Skills from Demonstration for Re-targetable Visuomotor Control

Jonathan Chang*1, Nishanth Kumar*1, Sean Hastings1, Aaron Gokaslan1, Diego Romeres2, Devesh K. Jha2, Daniel Nikovski2, George Konidaris1 and Stefanie Tellex1

Abstract—Robots need to learn skills that can not only generalize across similar problems but also be directed to a specific goal. Previous methods either train a new skill for every different goal or do not infer the specific target in the presence of multiple goals from visual data. We introduce an end-to-end method that represents targetable visuomotor skills as a goal-parameterized neural network policy. By training on an informative subset of available goals with the associated target parameters, we are able to learn a policy that can zero-shot generalize to previously unseen goals. We evaluate our method in a representative 2D simulation of a button-grid and on both button-pressing and peg-insertion tasks on two different physical arms. We demonstrate that our model trained on 33% of the possible goals is able to generalize to more than 90% of the targets in the scene for both simulation and robot experiments. We also successfully learn a mapping from target pixel coordinates to a robot policy to complete a specified goal.

I. INTRODUCTION

Recent advances have allowed for deep learning of visuomotor skills from scratch. However, because such skills are based on strictly visual input, they are insufficiently flexible for tasks with multiple potential goals visible in the same scene. For example, an intelligent robot should be able to identify and flip the correct light switch even in the presence of other light switches in the scene; it should be able to pick up the white queen on a chessboard even when there are other pieces, or press a specific button on a remote control. To achieve this behavior, the agent must learn a targetable skill: one that can be instructed to pick up that piece, or flip that switch, by taking as additional input a parameter vector that disambiguates its goal. An important question is then how to build targetable visuomotor skills in a sample-efficient manner.

While effective at learning an end-to-end visuomotor policy with a single goal from raw pixel data, current methodologies cannot be instructed either to a specific goal or towards a target that was not seen in training. Some approaches to solve this have involved either training a unique policy for every possible goal, or creating a learning algorithm that could quickly converge to the desired policy given a few training examples [9, 11, 12]. However, the amount of training data needed becomes prohibitive for related tasks such as pressing an elevator button, where one would need to train a separate skill for each button. Additionally, even though few-shot learning and meta-learning algorithms have drastically reduced the amount of training data needed, these algorithms require the robotic system to have a separate policy to target a specific goal. Alternatively, da Silva et al. [4] introduced parameterized skills that allowed robot agents to target specific goals with a single policy; however, these methods were not end-to-end and used a separate module to infer the goal from raw pixel data.

To address these shortcomings, we present a method to train targetable visuomotor skills from expert demonstrations. Our model consists of three modules: vision, auxiliary task, and control. The vision module takes as input an RGB and depth image and outputs a visual encoding. The auxiliary task module takes the visual encoding and the goal-parameter and tries to predict the final pose of the end-effector. Finally, the control module infers the next time step’s linear velocity from the visual encoding, goal-parameter, and the predicted final state of the end-effector. By training on an informative subset of goal-parameters, we are able to train one policy that can execute its learned visuomotor skill to previously-unseen...
goals.

We evaluate our method via a series of experiments in simulation and on two different robotic arms. In simulation, we use a two dimensional grid representing a $3 \times 3$ button panel to study our method’s ability to generalize to novel goals. We show that the policy conditioned on the goal allows for the agent to generalize better in this multi-goal setting. Moreover, we also demonstrate how conditioning with an informative representation of the goal further improves performance. Finally, we extend our algorithm such that it can generalize to novel orientations of the goals in the scene. We then evaluate our method on two real-world robotic tasks that require significant precision in a continuous space. The KUKA LBR iwa-7 and the the MELFA RV-4FL arms were trained to solve button-pushing and peg-insertion tasks respectively on grid-based goals. In all of our experiments, our method was able to target specific goals and generalize to more than 90% of goals after being trained on a third of the third within the grid. Additionally, we demonstrate a goal-representation that allows our model to generalize to non-grid-based goals with no additional training data. Our open-source code is available on GitHub at https://github.com/h2r/parameterized-imitation-learning.

II. RELATED WORK

We present a method that combines parameterized skill learning [4, 5] and end-to-end deep visuomotor skill learning from demonstration [18, 20]. Parameterized skill learning as described by da Silva et al. [4] aims to learn a mapping from a given task parameter vector to a policy [4, 5, 6, 9, 16, 17, 21, 22, 24]. All of these methods show zero-shot generalization properties to unseen goal parameters and goals in settings with hand-designed compact state descriptors. Unlike prior work, we focus on directly learning from raw pixels while preserving the agent’s ability to generalize from a drastically reduced number of demonstrations to similar tasks with different goals by sharing data across tasks. Other approaches achieve similar generalization through coupled dictionary learning [15], gating functions [20], and modular sub-networks [7]. Different from our model, these methodologies either did not learn an end-to-end policy from raw sensory data or did not demonstrate their parameterization capabilities outside of the simulation domain. Our work aims to learn a single policy for a parameterized skill such that it is sample efficient, is end-to-end, and can be instructed to specific goals.

There has also been work in imitation learning that takes raw sensory data from human demonstrations and learns robot motor controls [2, 8, 10, 18, 19, 23, 25, 26]. Our model specifically takes inspiration from the behavioral cloning algorithm proposed by Zhang et al. [26] which learns to map input RGB/depth sensor data to subsequent linear and angular velocities for the end-effector. While these methods could generalize to unseen variations of the task when there was one possible goal in the visual frame, we found that they failed to generalize in the presence of multiple targets. For example, the behavioral cloning algorithm presented in Zhang et al. [26] would require 30 minutes worth of demonstrations for each button resulting in 270 minutes worth of demonstrations for nine buttons.

III. LEARNING DEEP PARAMETERIZED SKILLS

This section presents a behavioral cloning algorithm that learns parameterized neural network policies. Given a set of skills $S$, let $T = \{\tau^{(s)}\}$ be the set of goal parameters for each specific instance of the skill $s \in S$ and $\Omega = \{(o^{(s)}_t, \tau^{(s)}, u^{(s)}_t)\}$ be the dataset that consists of sets of state observations, corresponding goal parameters, and controls collected for all $s \in S$. Now let $\pi_{\theta}(u_t|o_t, \tau)$ be our neural network policy, parameterized by $\theta$, that learns control policies that are dependent on our goal parameters, $\tau$ at time step $t$. Our neural network architecture closely follows that of Zhang et al. [26] with the following modifications.

A. Task Parameterization

As proposed by da Silva et al. [4], a parameterized skill is a multi-task policy that maps input task parameters to end-effector controls. We learn the mapping $T \rightarrow \theta_S$ where $T$ is the set of task parameters and $\theta_S$ is the multi-task policy parameters for the family of skills $S$. We define $\tau^{(s)} \in T$ as the vector of task parameters injected into our policy for skill $s \in S$. We chose our $\tau$ to be the parameters that described the goal for our tasks.

B. Neural Network Policies

The neural network takes raw sensory data and a goal parameter as input and outputs robot motions. More formally, for time step $t$, the inputs $o_t = (I_t, D_t, P_{t-5:t-1})$ include (1) RGB images $I_t \in \mathbb{R}^{160 \times 120 \times 3}$, (2) depth images $D_t \in \mathbb{R}^{160 \times 120}$, and (3) positions of the end-effector $P_t \in \mathbb{R}^3$ for the 5 most recent steps where $P_{t-5:t-1} \in \mathbb{R}^{15}$. Furthermore, the net also takes as input $\tau$ which is (4) the task parameters.

Given these inputs, the neural network outputs the current control $u_t$ described as the linear velocity, $v_t \in \mathbb{R}^3$ of the end-effector.

The neural network architecture can be decomposed into three modules, $\theta = (\theta_{vision}, \theta_{aux}, \theta_{control})$. The first module consists of convolutional layers with a spatial-softmax layer [10, 18] to extract spatial feature points from the image to generate the current state encoding (Eq. 1). Every convolutional layer is followed by a layer of rectified linear units:

$$f_t = \text{CNN}(I_t, D_t; \theta_{vision}).$$

(1)

This is followed by a small fully connected network that takes as input the state encoding, $f_t$, and the goal parameter, $\tau$, to predict the auxiliary task:

$$a_t = \text{NN}(f_t, \tau; \theta_{aux}).$$

(2)

Finally, given the extracted state encoding $f_t$, end-effector positions $P_{t-5:t-1}$, goal parameters $\tau$, and auxiliary prediction $a_t$, we use a fully-connected network to predict the current time step’s controls:

$$u_t = \text{NN}(f_t, P_{t-5:t-1}, \tau, a_t; \theta_{control}).$$

(3)

For all of the fully-connected layers, we add a layer of rectified linear units.
C. Auxiliary Prediction Tasks

Our network includes an auxiliary prediction task as another means of self-supervision, resembling the approaches shown by Zhang et al. [26]. Similar to their findings, we found that leveraging the extra self-supervisory signals resulted in increased data efficiency. As mentioned in [26], there could be multiple auxiliary tasks, but we found that for our tested tasks, one auxiliary task module was sufficient. We added a module of two fully-connected layers after the spatial-softmax layer (Eq. 2) and fed the final layer of these modules back into the control module (Eq. 3) as shown in Figure 2.

The auxiliary tasks that we chose were limited to using the information that could be inferred from the dataset \( \Omega \). For our experiments, we had an auxiliary module for predicting the final end-effector pose for the task. We also found that generalization across novel goal parameters improved with the goal parameters being fed into this auxiliary prediction. All of these auxiliary prediction tasks were trained concurrently with the rest of the network.

D. Loss Functions

The loss function used for learning is a modification to a commonly used behavioral cloning loss also used in Zhang et al. [26]. Our algorithm uses \( l_1 \) and \( l_2 \) losses to fit to the training trajectories using visual and proprioceptive input. Given an example set of \( (o_t, \tau, u_t) \), we have the losses:

\[
L_{l1} = ||\pi_\theta(o_t, \tau) - u_t||_1 \tag{4}
\]

\[
L_{l2} = ||\pi_\theta(o_t, \tau) - u_t||_2^2. \tag{5}
\]

Furthermore, we add an arc-cosine loss to enforce consistency between the directions of the output and target velocity:

\[
L_{acos} = \arccos \frac{u_t^T \pi_\theta(o_t, \tau)}{||u_t||_2 ||\pi_\theta(o_t, \tau)||_2}. \tag{6}
\]

Finally, for our auxiliary prediction task, we use an \( l_2 \) loss:

\[
L_{aux} = ||\text{NN}(f_t; \theta_{aux}) - a_t||_2^2. \tag{7}
\]

The loss function for the whole algorithm is a weighted sum of the losses described by the above equations:

\[
L = \lambda_{l1} L_{l1} + \lambda_{l2} L_{l2} + \lambda_{acos} L_{acos} + \lambda_{aux} L_{aux}. \tag{8}
\]

\( \lambda = (\lambda_{l1}, \lambda_{l2}, \lambda_{acos}, \lambda_{aux}) \) was tuned for each experiment. For the button pressing task we use \( \lambda = (1.0, 0.01, 0.005, 1.0) \), whereas for the peg-insertion task \( \lambda = (1.0, 10.0, 1.0). \) Policies were optimized using NovoGrad [13] with a learning rate of 0.0005 and batch size of 64. Training was done with randomly sampled batches from the dataset \( \Omega \).

IV. Evaluation

This section evaluates our method’s ability to learn targetable visuomotor skills in both simulated and physical domains.

Fig. 3: The first and second row show the RGB and depth image inputs respectively for the (a) 2D, (b) button pressing, and (c) peg insertion experiments.
A. 2D Button Simulation

We experimented on a two-dimensional representation of the robot button-pressing task. We used a $3 \times 3$ grid of blue squares representing buttons and a black circle representing the agent, shown in Figure 3a. We designed the simulation such that the agent would occlude the squares when it overlapped them.

We collected 100 trajectories for each square where the agent began at a random position along the right and top edges of the scene. As shown in Figure 3b, the depth image input was a black screen. For this experiment, we used the row/column index and later the pixel coordinates of the square as $\tau$. For example, we set $\tau$ as $(0,0)$ and $(200,150)$ for the top-left square, $(2,0)$ and $(600,450)$ for the bottom-left, and so on for the row/column index and pixel coordinates respectively.

We trained our network on random subsets of $\tau$ corresponding to the nine squares to test how well our model generalized. As an example: $\{(0,0), (0,2), (2,0), (2,2)\}$ is a length 4 subset of $\tau$’s corresponding to the 4 corner squares in our grid. For every random subset, we trained our network for 50 epochs and evaluated the performance of our model on 100 trials for each button. The trajectories for the trials were generated by moving the agent in one of eight cardinal directions towards the goal at each time-step. A trial was counted as a success only if the agent slowed to a complete stop at the correct blue square.

We conducted an ablation study to evaluate the effect of our model’s goal parameterization. Our ablation of the goal-parameter was equivalent to the architecture shown in Figure 2 without the goal-conditioning module. In addition, we evaluated a version of our algorithm that uses an unstructured representation for the goal-parameter to study how structure in the choice of representation for $\tau$ affects our performance. Specifically, we used a one-hot vector where a goal corresponded to a randomly-chosen index of a nine-dimensional vector. Finally, we tested another version of our model that used the specific button’s pixel location within the image as $\tau$. For our ablation, one-hot, and pixel-based versions, we repeated the above experiment with the specified model changes. The results are displayed in Figure 4.

Our experiments show that injecting the goal parameter $\tau$ defined by either the relative row/column indices or the pixel coordinates expands the functionality of the original behavioral cloning algorithm. As Figure 4 shows, the ablation algorithm is only able to achieve approximately 11% success rate when trained on only one square and its performance generally degraded with more squares in the training set. We also see that the one-hot vector representation of the goal parameter only allows for the algorithm to successfully reach squares seen in training. The addition of our structured goal-parameter $\tau$ - either as the relative row-column index or pixel location - allowed the network to learn the mapping from $\tau$ to the location of a square, and thus enable it to not only select squares it has seen during training, but also generalize to novel squares based on novel $\tau$ inputs.

Additionally, we found that given our choice of $\tau$, we were able to consistently achieve perfect performance after having trained on roughly 7/9 of the possible targets regardless of the combination of goal-parameters in the training set. We also found that given optimal selection of goal parameters in the training set, we were able to achieve full generalization for the whole button panel with 33% of the goal-parameters represented in the training data. Our experiments show that our methodology is able to learn the entire space of goals after seeing roughly a third of the possible goals provided that the training set represents an informative subset of the entire goal-parameter space.

Finally, we found that using a button’s pixel location as $\tau$ allowed it to generalize well. As shown in Figure 4, we found that this $\tau$ enabled us to achieve perfect generalization to the grid with only two goals in the training set. We also found that parameterizing by the pixel location allowed the model to generalize to arbitrary locations in the scene even
when the visual input was drastically changed. That is, we were able to transfer our learned skill to target buttons in a variety of previously-unseen orientations as shown in Figure 5. We demonstrate this behavior further in our supplementary video.

B. Robot Button-Pressing Task

In this experiment, we show that our method can work robustly on a real-world robotic task. We used a KUKA LBR iiwa-7 equipped with a Schunk gripper to press buttons on a 3D, $4 \times 4$ button panel. Similar to Section IV-A, we parameterized our button-grid with a row/column tuple of the button’s location on the grid. For training, we collected 100 trials of the robot’s end-effector beginning at a random position and following a straight line with noise to the specified button. The end-effector’s final position was varied with noise drawn from a Gaussian distribution such that the robot would press the button differently each time.

For this experiment, we used specific subsets of buttons within a $3 \times 3$ section of the grid as training data. We chose combinations that had been found to generalize well in our two-dimensional simulation. We found that we achieved good performance within 100 epochs of training. We evaluated for three attempts on each button and deemed an attempt successful if the robot pressed the button. Results are displayed in Figure 7.

The performance of the robot on the task was similar to the average performance with the row/column τ for our 2D simulation. The robot always successfully pushed buttons seen during training. After having been trained on just three buttons, the robot successfully generalized to $88\%$ of previously-unseen buttons and $92\%$ of all buttons. Interestingly, the average performance of the robot stayed the same when trained on three to five buttons because the robot failed to press exactly two unseen buttons in each of these cases. However, we qualitatively observed that the robot did get progressively closer to succeeding when trained on more buttons, but was still not close enough to successfully press. When trained on six buttons or more, the robot achieved a $100\%$ success rate on all buttons in the grid. We also evaluated our model again with τ as the pixel coordinates of the goal in the input RGB image. We found that our model exhibited similar generalization properties with the added functionality of being able to handle arbitrary locations of the button board. As shown in Figure 8, we were able to generalize to various different orientations of goals as well as to goals that were visually different from our original button board. This suggests that our model is also able to generalize to any button board that fits within its scene, successfully learning a mapping from raw pixel coordinates to policy parameters. We demonstrate this behavior further in our supplementary video.

C. Robot Peg-Insertion Task

We further evaluated our method on a real-world robotic task that required significant precision. As pictured in Figure 8, we used a MELFA RV-4FL robotic arm to perform peg-insertions in a $3 \times 3$ grid of holes. We performed an experiment similar to the one in Section IV-B, with a different robot, task-setting, and subsets of goals chosen. The collected trajectories are straight lines from random start

Fig. 7: Results from our experiments on the MELFA and KUKA robots.

Fig. 8: Various goal orientations the model generalized to after being trained on the original $3 \times 3$ button grid using goal pixel location as τ.
positions, uniformly sampled on the area above the holes grid, to a waypoint of random heights directly above the hole, and then a straight path downward into the hole to complete the insertion. Shown in Figure 9a, an example three-goal subset that we trained on was $\tau \in \{(1, 1), (1, 2), (2, 1)\}$. During data collection 60 such trajectories were collected for each hole, for a total of 540 insertions. Results of successful insertions for different training combination of holes are displayed in Figure 7.

Our method performed remarkably well on this task. The robot generalized to all nine holes with a 100% success rate after having seen insertions performed on only three holes during training. All executions were performed in stiffness control mode; however, the control outputs were precise enough that compliant motion was not necessary. We found that the outputted control trajectories to new goals were very smooth as shown in Figure 9b. The differences between our performance on the peg-insertion and button-pressing tasks can likely be explained by two differences: the noise in the training trajectories and the subsets of the goals that the robots were trained on. As shown in Figure 9a, the training trajectories for our peg-insertion experiments had no noise on their final positions because the insertion tolerance was too small to induce much noise. However, the training trajectories for our button-pressing experiment had significant noise on the end-effector’s final position. This could have lowered the precision of the model, leading to near-misses for buttons that were not seen during training. In addition, the two tasks did not use the same subsets of goals for training in every case. It is possible that some of the subsets used for the peg-insertion task were more optimal than those used for the button-pressing task.

V. CONCLUSION AND FUTURE WORK

We introduce a method that uses a neural network to learn deep parameterized visuomotor skills. We show that our method is able to learn to perform new instances of a task that were not seen at training time and demonstrate this via experiments on different tasks in a 2D simulation and on two different physical robots. We empirically study our method’s generalization and dependence on user-specified goal parameters and show that it is able to generalize to all possible instances of various tasks after having seen at most six out of nine instances at training, provided a diverse subset of goals. This is summarized in Table I. We also show that depending on the choice of goal parameters, our model is able to generalize to different orientations of multiple goals in the scene with no additional training data.

We hope to investigate various architectural changes to improve our method in the future. Of particular promise is the use of Reinforcement Learning (RL) to precisely learn stopping conditions and intermediate fine-grained movements. Additionally, we hope to extend our idea of goal parameterization to other frameworks for learning from demonstration such as inverse reinforcement learning [11] or generative adversarial imitation learning [14] (GAIL). Recent work [8] has successfully extended goal-parameterization to GAIL and shown encouraging results in simulation. We hope such methods will enable us to represent more complex parameterized skills.

Finally, we hope to investigate different ways of parameterizing the task itself. For instance, we might use natural language commands or even multi-modal encodings as our goal parameter. Studying such different parameterizations could help us formulate additional conditions or guidelines for selecting ‘good’ representations and values of goal parameters to train on.

| Indices | 2D | Button-Pressing | Peg-Insertion |
|---------|----|-----------------|---------------|
| Pixel   | 2  | 4               | -             |

TABLE I: Lowest number of goal parameters, either row/column indices or pixel coordinates, at which we observed perfect 100% generalization to the 3x3 grid of goals in our experiments.
REFERENCES

[1] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In Proceedings of the Twenty-first International Conference on Machine Learning, ICML ’04, New York, NY, USA, 2004. ACM. doi: 10.1145/1015330.1015430.

[2] Pooya Abolghasemi, Amir Mazaheri, Mubarak Shah, and Ladislau Boloni. Pay attention! - robustifying a deep visuomotor policy through task-focused visual attention. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.

[3] Felipe Codevilla, Matthias Müller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 1–9, May 2018. doi: 10.1109/ICRA.2018.8460487.

[4] Bruno da Silva, George Konidaris, and Andrew Barto. Learning parameterized skills. International Conference on Machine Learning, 2. 06 2012.

[5] Bruno da Silva, Gianluca Baldassarre, George Konidaris, and Andrew Barto. Learning parameterized motor skills on a humanoid robot. pages 5239–5244, 05 2014. doi: 10.1109/ICRA.2014.6907629.

[6] Marc Peter Deisenroth, Peter Englert, Jan Peters, and Dieter Fox. Multi-task policy search for robotics. In 2014 IEEE International Conference on Robotics and Automation (ICRA), pages 3876–3881, 2014.

[7] Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. Learning modular neural network policies for multi-task and multi-robot transfer. 2017 IEEE International Conference on Robotics and Automation (ICRA), pages 2169–2176, 2016.

[8] Yiming Ding, Carlos Florensa, Mariano Phielipp, and Pieter Abbeel. Goal-conditioned imitation learning. CoRR, abs/1906.05838, 2019. URL http://arxiv.org/abs/1906.05838

[9] Yan Duan, Marcin Andrychowicz, Bradly C. Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. arXiv preprint arXiv:1703.07326, 2017.

[10] Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel. Deep spatial autoencoders for visuomotor learning. 2016 IEEE International Conference on Robotics and Automation (ICRA), pages 512–519, 2015.

[11] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML’17, pages 1126–1135, 2017.

[12] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. ArXiv, abs/1709.04905, 2017.

[13] Boris Ginsburg, Patrice Castonguay, Oleksii Kuchaiev, Vitaly Lavrukhin, Ryan Leary, Jason Li, Huyen Nguyen, and Jonathan M. Cohen. Stochastic gradient methods with layer-wise adaptive moments for training of deep networks. ArXiv, abs/1905.11286, 2019.

[14] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. CoRR, abs/1606.03476, 2016.

[15] David Isele, Mohammad Rostami, and Eric Eaton. Using task features for zero-shot knowledge transfer in lifelong learning. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI’16, pages 1620–1626, 2016.

[16] Jens Kober, Andreas Wilhelm, Erhan Oztok, and Jan Peters. Reinforcement learning to adjust parametrized motor primitives to new situations. Autonomous Robots, 33:361–379, 11 2012. doi: 10.1007/s10514-012-9290-3.

[17] Andras Gabor Kupcsik, Marc Peter Deisenroth, Jan Peters, and Gerhard Neumann. Data-efficient generalization of robot skills with contextual policy search. In AAAI, 2013.

[18] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. J. Mach. Learn. Res., 17(1):1334–1373, January 2016.

[19] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Diédré Quinn. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. The International Journal of Robotics Research, 37(4-5):421–436, 2018. doi: 10.1177/0278364917710318.

[20] Katharina Muelling, Jens Kober, Oliver Kroemer, and Jan Peters. Learning to select and generalize striking movements in robot table tennis. International Journal of Robotics Research (IJRR), 32(3):263 – 279, March 2013.

[21] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. In 2009 IEEE International Conference on Robotics and Automation, pages 763–768, May 2009. doi: 10.1109/ROBOT.2009.5152385.

[22] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In ICML, 2015.

[23] Avi Singh, Larry Yang, Kristian Hartikainen, Chelsea Finn, and Sergey Levine. End-to-end robotic reinforcement learning without reward engineering. arXiv preprint arXiv:1904.07854, 2019.

[24] Freek Stulp, Gennaro Riai, Antoine Hoarau, Serena Ivaldi, and Olivier Sigaud. Learning compact parameterized skills with a single regression. In 2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids), pages 417–422, 2013.

[25] Mel Vecerik, Oleg Sushkov, David Barker, Thomas Rothrhl, Todd Hester, and Jon Scholz. A practical approach to insertion with variable socket position
using deep reinforcement learning. In 2019 International Conference on Robotics and Automation (ICRA), pages 754–760, May 2019. doi: 10.1109/ICRA.2019.8794074.

[26] Tianhao Zhang, Zoe McCarthy, Owen Jowl, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. 2018 IEEE International Conference on Robotics and Automation (ICRA), 2017.