A Survey of Deep Network Solutions for Learning Control in Robotics: From Reinforcement to Imitation

Lei Tai*[1], Jingwei Zhang*[2], Ming Liu[1], Joschka Boedecker[2], Wolfram Burgard[2],

***Abstract*—Deep learning techniques have been widely applied, achieving state-of-the-art results in various fields of study. This survey focuses on deep learning solutions that target learning control policies for robotics applications. We carry out our discussions on the two main paradigms for learning control with deep networks: *Deep Reinforcement Learning* and *Imitation Learning*. For *Deep Reinforcement Learning* (DRL), we begin from traditional reinforcement learning algorithms, showing how they are extended to the deep context and effective mechanisms that could be added on top of the DRL algorithms. We then introduce representative works that utilize DRL to solve navigation and manipulation tasks in robotics. We continue our discussion on methods addressing the challenge of the *reality gap* for transferring DRL policies trained in simulation to real-world scenarios, and summarize robotics simulation platforms for conducting DRL research. For *Imitation Leaning*, we go through its three main categories, behavior cloning, inverse reinforcement learning and generative adversarial imitation learning, by introducing their formulations and their corresponding robotics applications. Finally, we discuss the open challenges and research frontiers.**

***Index Terms*—Deep Learning, Robotics, Deep Reinforcement Learning, Imitation Learning.**

I. Introduction

A. Deep Learning

Deep learning, as a solution for artificial intelligence that is capable of building progressively more abstract representations of input data, plays an essential role in various fields of study (Goodfellow et al., 2016).

From image classification (Krizhevsky et al., 2012; He et al., 2016; Huang et al., 2017), to semantic segmentation (Long et al., 2015; Chen et al., 2016), from playing Atari games at the human-level with only pixel inputs (Mnih et al., 2015, 2016), to learning policies capable of driving real robotic systems in navigation (Zhu et al., 2017b; Zhang et al., 2017a; Tai et al., 2017) and manipulation (Levine et al., 2016; Yu et al., 2018) tasks, the learning power of deep networks drives the state-of-the-art in various research directions (Schmidhuber, 2015).

Recent years have witnessed a rapidly growing trend of utilizing deep learning techniques for robotics tasks. Replacing hand-crafted features with learned hierarchical distributed deep features, learning control policies directly from high-dimensional sensory inputs, the robotics community is making solid progress towards building fully autonomous intelligent systems.

B. Deep Learning for Robotics: From Perception to Control

Autonomous intelligent robotics systems require two essential building blocks: perception and control.

The perception pipeline can be viewed as a passive procedure: intelligent agents receive observations from the environment, then infer desired properties or detect target quantities from those sensory inputs. We refer readers to Deng (2014) and Guo et al. (2016) for a comprehensive overview of deep learning techniques for perception. Compared with pure perception, the problem of control for autonomous agents goes one step further, seeking to actively interact with or influence the environment by conducting sequences of actions. This active nature leads to the following major distinctions between perception and control, in terms of deep learning based approaches:

Data distribution: When learning perception through supervised learning techniques, the training datasets are collected and labeled before the learning phase begins. In this case, the data points can be viewed as being independently and identically distributed (i.i.d), such that a direct mapping from the input to the labels can be learned via standard stochastic gradient descent methods and variants. In contrast, for control, the datasets are collected in an online manner, which makes the data points sequential in nature: the consecutive observations received by the agent are temporally correlated since the agent actively influences the data distribution by the actions it takes. Ignoring this underlying temporal correlation would lead to compounding errors (Bagnell, 2015).

Supervision signal: The supervision for learning perception is often direct and strong, in that each training sample is provided along with its ground truth label. In control tasks, on the other hand, either only sparse reward signals are available when learning behaviors through *deep reinforcement learning*, or the feedback is often delayed and not instantaneous, even when demonstrations from experts are provided in the scenario of *imitation learning*, since the credit for achieving a certain goal needs to be correctly assigned to all the actions taken along the trajectory.

Data collection: As discussed before, the dataset for perception can be collected off-line, while the dataset for control has

*indicates equal contribution.

[1]Lei Tai and Ming Liu are with The Hong Kong University of Science and Technology. {ltai, eelium}ust.hk

[2]Jingwei Zhang, Joschka Boedecker and Wolfram Burgard are with University of Freiburg. {zhang, jboedeck, burgard}@informatik.uni-freiburg.de

to be collected in an on-line manner, since actions are actively involved in the learning process. This greatly limits the number of samples one can collect, since executing actions in the real world with real robotics systems is a relatively expensive procedure. In cases where the control policies are trained in simulation, the problem of the *reality gap* arises when they are deployed in real-world scenarios, where the discrepancies between the modalities of the synthetic renderings and the real sensory readings impose major challenges.

Recognizing those distinctions, various deep learning based algorithms have been proposed to solve control for robotics. In this survey, we review the deep learning approaches for control tasks based on their underlying learning paradigms, and we carry out our discussion through the following sections:

- **Sec. II** Deep Reinforcement Learning
  - **Sec. II-A** RL Overview
  - **Sec. II-B** RL Algorithms
  - **Sec. II-C** DRL Algorithms
  - **Sec. II-D** DRL Mechanisms
  - **Sec. II-E** DRL for Navigation
  - **Sec. II-F** DRL for Manipulation
  - **Sec. II-G** The Reality Gap: From Simulation to the Real World
  - **Sec. II-H** Simulation Platforms
- **Sec. III** Imitation Learning
  - **Sec. III-A** Behavior Cloning
  - **Sec. III-B** Inverse Reinforcement Learning
  - **Sec. III-C** Generative Adversarial Imitation Learning

II. Deep Reinforcement Learning

Being the first to stabilize large-scale reinforcement learning with deep convolutional neural networks as function approximators, deep Q-networks (DQN) (Mnih et al., 2015) have brought increased research and applications of deep reinforcement learning (DRL) methods. In the following we first review the basic concepts and algorithms in traditional reinforcement learning (RL). Then we continue to the several most influential DRL algorithms and mechanisms, on the basis of which we discuss DRL solutions for robotics control, with an emphasis on navigation and manipulation applications.

A. *RL Overview*

We formalize a robotics task (e.g., navigation, manipulation) as a *Markov Decision Process* (MDP), in which the agent interacts with the environment through a sequence of observations, actions, and reward signals. An MDP is a $5-$tuple $\langle \mathcal{S}, \mathcal{A}, P, \mathcal{R}, \gamma \rangle$:

- $\mathcal{S}$: set of all states.
- $\mathcal{A}$: set of all actions.
- $\mathcal{P}$: the transition dynamics, where $P(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ defines the distribution of the next state $\mathbf{s}'$ by taking action $\mathbf{a}$ in state $\mathbf{s}$, where $\mathbf{s}, \mathbf{s}' \in \mathcal{S}, \mathbf{a} \in \mathcal{A}$. We also denote the initial state distribution $P(\mathbf{s}_0)$ as ρ_0.
- $\mathcal{R}$: set of all possible rewards. In the following, we denote the instantaneous scalar reward received by the agent by taking action $\mathbf{a}_t$ from state $\mathbf{s}_t$ as $R_{t+1}(\mathbf{s}_t, \mathbf{a}_t)$, and use R_{t+1} as short for $R_{t+1}(\mathbf{s}_t, \mathbf{a}_t)$. There also exist other definitions of the reward function that depend only on the state itself, in which $R(\mathbf{s})$ refers to the reward signal that the agent receives by arriving at state $\mathbf{s}$. In some of the following discussions, the negative counterpart of the reward function, the cost function, is used, and is denoted as $c(\mathbf{s})$.
- γ: a discount factor in the range of $[0, 1]$.

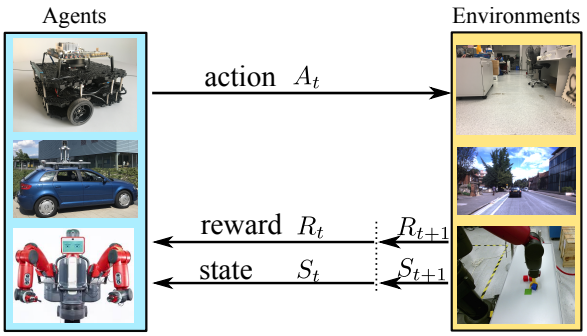

Fig. 1. The reinforcement learning loop in the context of robotics. In state $\mathbf{s}_t$, the autonomous agent takes and action $\mathbf{a}_t$, receives a reward R_{t+1}, and transits to the next state $\mathbf{s}_{t+1}$.

In an MDP, the agent takes an action $\mathbf{a}_t$ in state $\mathbf{s}_t$, receives a reward R_{t+1}, and transits to the next state $\mathbf{s}_{t+1}$ following the transition dynamics $\mathcal{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$. This process in the context of robotics is depicted in Fig.1.

In robotics, we mainly consider *episodic* MDPs, where there exists a *terminal state* (e.g., a mobile ground vehicle reaches a certain goal location, a manipulator successfully grabs a red cup) that, once reached, terminates the current *episode*. Also for an *episodic* MDP with a time horizon of T, an *episode* will still be terminated after a maximum of T time steps, even if by then the *terminal state* has not yet been reached.

Another point worth mentioning is the *partial observability*. In a robotics task, an autonomous agent perceives the world with its onboard sensor (e.g., RGB/depth camera, IMU, laser range sensor, 3D Lidar), receiving one observation per time step. However, simply representing $\mathbf{s}_t$ by $\mathbf{x}_t$ often does not satisfy the *Markov* property: one such sensor reading can hardly capture all the necessary information for the agent to make decisions in the future, in which case the underlying procedure is called a Partial Observeble MDP (POMDP). This is often dealt with by either stacking several (e.g, N) consecutive observations $\{\mathbf{x}_{t-N+1}, \mathbf{x}_{t-N+2}, \ldots, \mathbf{x}_t\}$ to represent $\mathbf{s}_t$, or by feeding $\mathbf{x}_t$ into a *recurrent* neural network instead of a *feed forward* one, such that the past information is naturally accounted with (e.g., by the cell state when using the long short-term memories (LSTMs).

Reinforcement learning agents are designed to learn from interactions how to behave to achieve a certain goal (Sutton and Barto, 1998). More precisely, here the objective of learning is to maximize the *expected discounted return*, where the

discounted return is defined as follows:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{T-t-1} R_T \quad (1)$$

$$= \sum_{k=t}^{T} \gamma^{k-t} R_{k+1}. \quad (2)$$

To solve control, two important definitions are introduced:

- **Policies**: π, μ
  - $\pi(\mathbf{a}|\mathbf{s})$: stochastic policy, where actions are drawn from a probability distribution defined by $\pi(\mathbf{a}|\mathbf{s})$.
  - $\mu(\mathbf{s})$: determinstic policy, where actions are deterministically selected for a given state $\mathbf{s}$.
- **Value functions**: V, Q
  - $V^\pi(\mathbf{s})$: state-value function, defined as the expected return when starting from state $\mathbf{s}$ and following policy π thereafter:

$$V^\pi(\mathbf{s}) = \mathbb{E}_\pi \left[G_t | \mathbf{s}_t = \mathbf{s} \right] \quad (3)$$

$$= \mathbb{E}_\pi \left[\sum_{k=t}^{T} \gamma^{k-t} R_{k+1} | \mathbf{s}_t = \mathbf{s} \right]. \quad (4)$$

  - $Q^\pi(\mathbf{s}, \mathbf{a})$: action-value function, defined as the expected return by taking the action $\mathbf{a}$ from state $\mathbf{s}$, then following π thereafter:

$$Q^\pi(\mathbf{s}, \mathbf{a}) = \mathbb{E}_\pi \left[G_t | \mathbf{s}_t = \mathbf{s}, \mathbf{a}_t = \mathbf{a} \right] \quad (5)$$

$$= \mathbb{E}_\pi \left[\sum_{k=t}^{T} \gamma^{k-t} R_{k+1} | \mathbf{s}_t = \mathbf{s}, \mathbf{a}_t = \mathbf{a} \right]. \quad (6)$$

  - $Q^*(\mathbf{s}, \mathbf{a})$: optimal value function (We omit the case for state-value function V here since the action-value function Q is a much more effective representation for control.):

$$Q^*(\mathbf{s}, \mathbf{a}) = \max_\pi Q^\pi(\mathbf{s}, \mathbf{a}). \quad (7)$$

  - $\pi^*(\mathbf{a}|\mathbf{s})$: optimal policy:

$$\pi^*(\mathbf{a}|\mathbf{s}) = \operatorname*{argmax}_{\mathbf{a}} Q^*(\mathbf{s}, \mathbf{a}). \quad (8)$$

B. RL Algorithms

With the definitions of the core components, we now continue to discuss the different classes of RL algorithms. We emphasize those methods that have been extended with deep learning variants.

*1) **Value-based Methods**:* These methods are based on estimating the values of being in a given state, then extracting the control policies from the estimated values. The recursive value estimation procedures are based on the *Bellman Equations*. Below, we list the *Bellman Expectation Equation* (Eq. 9) and the *Bellman Optimality Equation* (Eq. 10):

$$Q^\pi(\mathbf{s}, \mathbf{a}) = \mathbb{E}_\pi \left[R_{t+1} + \gamma Q^\pi(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) | \mathbf{s}_t = \mathbf{s}, \mathbf{a}_t = \mathbf{a} \right], \quad (9)$$

$$Q^*(\mathbf{s}, \mathbf{a}) = \mathbb{E} \left[R_{t+1} + \gamma \max_{\mathbf{a}'} Q^*(\mathbf{s}_{t+1}, \mathbf{a}') | \mathbf{s}_t = \mathbf{s}, \mathbf{a}_t = \mathbf{a} \right]. \quad (10)$$

Following the formulations of Eq. 9 and Eq. 10 respectively, we have the two most well-known value-based RL methods: *SARSA* and *Q-learning*, which follow the same recursive backup procedures, given as follows:

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) \leftarrow Q^\pi(\mathbf{s}_t, \mathbf{a}_t) + \alpha \delta_t, \quad (11)$$

$$\delta_t = \mathbf{y}_t - Q^\pi(\mathbf{s}_t, \mathbf{a}_t). \quad (12)$$

In this estimation procedure, Q-values are recursively updated by a step size of α towards a target value $\mathbf{y}_t$. δ_t is termed the *td-error* (temporal difference error), and $\mathbf{y}_t$ the *td-target*.

The difference between *SARSA* and *Q-learing* comes in their *td-target's*. Below, we list the *td-targets* for *SARSA* and *Q-learing* in Eq. 13 and Eq. 14 respectively:

$$\mathbf{y}_t^{SARSA} = R_{t+1} + \gamma Q^\pi(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}), \quad (13)$$

$$\mathbf{y}_t^{Q\text{-}learning} = R_{t+1} + \gamma \max_{\mathbf{a}'} Q^\pi(\mathbf{s}_{t+1}, \mathbf{a}'). \quad (14)$$

SARSA updates its Q-value estimates using the transitions generated by following the behavioural policy π, which makes *SARSA* an *on-policy* method; *Q-learning*, on the other hand, is *off-policy*, since its value estimations are updated not towards the behavioural policy, but towards a target optimal policy.

There are also other value-based methods, such as *Monte-Carlo control*, which uses the true return of complete trajectories as its update target instead of bootstrapping from old estimates, and λ-variants, which mix the sample return and 1-step lookahead estimations.

A reformulation of the Q-value function, the *successor representation* (Dayan, 1993), is also studied in the recent literature (Kulkarni et al., 2016; Barreto et al., 2017; Zhang et al., 2017a):

$$R_{t+1}(\mathbf{s}_t, \mathbf{a}_t) = \phi(\mathbf{s}_t, \mathbf{a}_t)^\top \cdot \omega, \quad (15)$$

$$Q^\pi(\mathbf{s}, \mathbf{a}) = \psi^\pi(\mathbf{s}, \mathbf{a})^\top \cdot \omega, \quad (16)$$

where

$$\psi^\pi(\mathbf{s}, \mathbf{a}) = \mathbb{E}_\pi \left[\sum_{k=t}^{T} \gamma^{k-t} \phi(\mathbf{s}_k, \mathbf{a}_k) | \mathbf{s}_t = \mathbf{s}, \mathbf{a}_t = \mathbf{a} \right], \quad (17)$$

is termed the *successor feature*. This line of formulation decouples the task specific reward estimation into the estimation of representative features $\phi(\cdot)$ and a reward weight ω, and the estimation of the expected occurrence of the features $\phi(\cdot)$ under specific world dynamics following a specific policy. It combines the computational efficiency of model-free methods with the flexibility of some model-based methods. We refer readers to Dayan (1993), Kulkarni et al. (2016), Barreto et al. (2017) and Zhang et al. (2017a) for more detailed discussions and extensions.

*2) **Policy-based Methods**:* Unlike value-based methods, policy-based methods do not maintain value estimations, but work directly on policies. When it comes to high-dimensional or continuous action spaces, policy-based methods generally give much more effective solutions than value-based approaches. They can learn stochastic policies instead of just deterministic policies, and have better convergence properties.

Policy-based approaches operate on parameterized policies, and search for parameters that maximize the policy objective function. The policy search can be carried out in two

paradigms: gradient-free (Fu et al., 2005; Szita and Lörincz, 2006) and gradient-based. We focus on the gradient descent methods from the gradient-based family as they remain the method of choice in recent studies. More formally, given policy $\pi_\theta(\cdot)$ with parameters θ, policy optimization searches for the best θ that maximizes an objective function $\mathcal{J}(\pi_\theta)$:

$$\mathcal{J}(\pi_\theta) = \mathbb{E}_{\pi_\theta}[f_{\pi_\theta}(\cdot)]. \tag{18}$$

Here, $f_{\pi_\theta}(\cdot)$ is a *score function*, which judges the goodness of a policy. There are multiple valid choices for the *score function*; we refer readers to Schulman et al. (2015b) for a full discussion.

The *policy gradient* is defined as

$$\nabla_\theta \mathcal{J}(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta \cdot f_{\pi_\theta}(\cdot)\right]. \tag{19}$$

Intuitively speaking, firstly, some actions, experiences or trajectories are sampled following the current policy π_θ and the goodness of those samples is given by $f_{\pi_\theta}(\cdot)$, the *score function* and $\nabla_\theta \log \pi_\theta$ points out the direction in the parameter space that would lead to an increase of the probability of those actions being sampled. Thus, by ascending along the *policy gradient* given in Eq. 19, we end up with policies that are capable of generating samples with higher scores.

The standard REINFORCE algorithm (Williams, 1992), a well-known method in RL, plugs in the sample return as the *score function*:

$$f_{\pi_\theta}(\cdot) = G_t. \tag{20}$$

This algorithm, however, suffers from the very high variance. A common way to reduce the variance of the estimation while keeping it unbiased is by subtracting a *baseline* $b(\mathbf{s})$ from the return:

$$f_{\pi_\theta}(\cdot) = G_t - b_t(\mathbf{s}_t). \tag{21}$$

A commonly used *baseline* is a learned estimate of the *state-value* function $V(\mathbf{s})$. This leads us to the *actor-critic* class of algorithms, since it involves estimating the value functions along with policy search.

Before we go into *actor-critic* methods, several details are worthy of pointing out.

Firstly, directly following the policy gradient might not be desirable in the robotics setting, since hardware constraints and safety requirements should be carefully dealt with. Popular approaches for cautious exploration include avoiding significant changes in the policy, or explicitly discouraging entering undesired regions in the state space (Deisenroth et al., 2013).

We also note that, so far, we have only been discussing the policy gradient for the stochasitic polices, which integrate over both the state and action spaces, and might not be efficient in high-dimentional action spaces. The deterministic policy gradient (Silver et al., 2014), on the other hand, only requires integrating over the state space, which makes it a much more sample-efficient algorithm. Below we list the *stochastic policy gradient* (for $\pi_\theta(\mathbf{a}|\mathbf{s})$, Eq. 22) and the *deterministic policy gradient* (for $\mu_\theta(\mathbf{s})$, Eq. 23) when using the Q-value function as their *score function*:

$$\nabla_\theta \mathcal{J}(\pi_\theta) = \mathbb{E}_{\mathbf{s},\mathbf{a}}\left[\nabla_\theta \log \pi_\theta(\mathbf{a}|\mathbf{s}) \cdot Q^\pi(\mathbf{s},\mathbf{a})\right], \tag{22}$$

$$\nabla_\theta \mathcal{J}(\mu_\theta) = \mathbb{E}_{\mathbf{s}}\left[\nabla_\theta \mu_\theta(\mathbf{s}) \cdot Q^\mu(\mathbf{s}, \mu_\theta(\mathbf{s}))\right]. \tag{23}$$

*3) **Actor-critic Methods**:* Following on from the discussions of the *policy-based* methods, *actor-critic* algorithms maintain an explicit representation of both the policy (the *actor*) and the value estimates (the *critic*). The most widely used *actor-critic* algorithms use the following *score function*:

$$f_{\pi_\theta}(\cdot) = Q^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) - V^{\pi_\theta}(\mathbf{s}_t). \tag{24}$$

Compared againest Eq. 21, Eq. 24 replaces the return G_t with its unbiased estimate $Q^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t)$, and uses $V^{\pi_\theta}(\mathbf{s}_t)$ as its *baseline* function to reduce variance. In fact,

$$A(\mathbf{s},\mathbf{a}) = Q(\mathbf{s},\mathbf{a}) - V(\mathbf{s}) \tag{25}$$

is called the *advantage* function, which estimates the advantage of taking a particular action $\mathbf{a}$ in state $\mathbf{s}$.

*4) **Integraing Planning and Learning**:* So far, we have been discussing *model-free* methods where the agent is not provided with the underlying transition model and simply learns optimal behaviors from experiences. There also exists another branch of *model-based* algorithms where a model is learned from experiences. with which the agent can interact and collect *imaginary rollouts* (Sutton, 1991), It has also been extended with DRL methods (Weber et al., 2017; Kalweit and Boedecker, 2017). However, the need for learning a model brings in another source of approximation error, and *model-based* RL can only perform as well as the estimated model. This problem might be partially dealt with by *Model Predicted Control* (MPC) methods, which are not a focus of this survey, so we will skip the details.

C. DRL Algorithms

Recent successes of DRL have extended the aforementioned algorithms to the high-dimensional domain, by deploying deep neural networks as powerful non-linear function approximators for the optimal value functions $V^*(\mathbf{s}), Q^*(\mathbf{s},\mathbf{a}), A^*(\mathbf{s},\mathbf{a})$, and the optimal policies $\pi^*(\mathbf{a}|\mathbf{s})$, $\mu^*(\mathbf{s})$. They usually take the observations as input (e.g, raw pixel images from Atari emulators (Mnih et al., 2015) or joint angles of robot arms ()), and output either the Q-values, from which greedy actions are selected, or policies that can be directly used to execute agents. In the following, we cover the most influential DRL algorithms.

*1) **DQN** (Mnih et al., 2015):* : As a *value-based* method, DQN approximates the optimal Q-value function with a deep convolutional neural network, called the deep *Q-network*, whose weights we denote as θ^Q: $Q(\mathbf{s},\mathbf{a};\theta^Q) \approx Q^*(\mathbf{s},\mathbf{a})$. In turn, the *td-error* (Eq. 12) and the *td-target* (Eq. 14) from the standard *Q-learning* are adopted into:

$$\delta_t^{\text{DQN}} = \mathbf{y}_t^{\text{DQN}} - Q(\mathbf{s}_t, \mathbf{a}_t; \theta_t^Q), \tag{26}$$

$$\mathbf{y}_t^{\text{DQN}} = R_{t+1} + \gamma \max_{\mathbf{a}'} Q(\mathbf{s}_{t+1}, \mathbf{a}'; \theta_t^-). \tag{27}$$

Then an update step is performed based on the following gradient calculation with a learning rate of α:

$$\theta_{t+1} \leftarrow \theta_t - \alpha \cdot \left(\partial \left(\delta_t^{\text{DQN}}(\theta_t^Q)\right)^2 / \partial \theta_t^Q\right). \tag{28}$$

Two main techniques have been proposed in DQN to stabilize learning: *target-network* and *experience replay*.

Target-network: In Eq. 27, the *td-target* is computed using the output from a *target-network* θ^-, instead of the *Q-network* θ^Q. The *target-network* and the *Q-network* share the same network architecture, but only the weights of the *Q-network* are learned and updated. The weights of the *Q-network* θ^Q are only periodically copied to the *target-network* θ^-. This reduces the correlations of the estimated Q-values with the target estimations. There is also *soft update* (Lillicrap et al., 2015), where a small portion of θ^Q are mixed into θ^- in every iteration, instead of the *hard update* used in the original DQN, where θ^Q are directly and completely copied to θ^- every several (e.g, $10,000$) iterations.

Experience replay: In this technique, instead of directly using the incoming frames from the online interactions, the collected experiences are firstly stored into a *replay memory*. During training, random samples are drawn from the *replay memory* (4 consecutive observations are stacked together to form a state, so as to deal with the *partial observability*) to be fed into the network as mini-batches. This way, gradient descent methods from the supervised learning literature can be safely used, to minimize the min-squared error between the predicted Q-values (output by the *Q-network*) and the target Q-values (output by the *target-network*). *Experience replay* thereby removes the temporal correlations in the consecutive observations, and smoothes over changes in the online data distribution.

Further techniques have been proposed on the basis of DQN to stabilize learning and improve efficiency: Double DQN (Van Hasselt et al., 2016) and Dueling DQN (Wang et al., 2016b).

For Double DQN (Van Hasselt et al., 2016), the greedy action is chosen based on the output from θ^Q (the original DQN uses θ^-), then the target Q-value of the chosen greedy action is computed using θ^- (Eq. 29). This prevents overoptimistic value estimates and avoids upward bias:

$$\mathbf{y}_t^{\text{Double}} = R_{t+1} + \gamma Q(\mathbf{s}_{t+1}, \underset{\mathbf{a}'}{\operatorname{argmax}}\, Q(\mathbf{s}_{t+1}, \mathbf{a}'; \theta_t^Q); \theta_t^-). \tag{29}$$

For Dueling DQN (Wang et al., 2016b), two output heads are used to estimate the *state-value* V and the *advantage* A respectively for each action. This helps the agent to efficiently learn which states are valuable, without having to learn the effect of each action for each state.

2) ***DDPG*** *(Lillicrap et al., 2015):* : DQN can deal with high-dimensional state spaces, but is only capable of handling discrete and low-dimensional action spaces. The deep deterministic policy gradient (DDPG) combines techniques from DQN with *actor-critic* methods, targeting solving continuous control tasks from raw pixels inputs.

If we write out the expectation in Eq. 9 for the *stochasitic policy* $\pi(\mathbf{a}|\mathbf{s})$ and *deterministic policy* $\mu(\mathbf{s})$, we get (E represents the environment that the agent is interacting with)

$$Q^\pi(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}_{R_{t+1}, \mathbf{s}_{t+1} \sim E}\left[R_{t+1} + \gamma \mathbb{E}_{\mathbf{a}_{t+1} \sim \pi}\left[Q^\pi(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})\right]\right], \tag{30}$$

$$Q^\mu(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}_{R_{t+1}, \mathbf{s}_{t+1} \sim E}\left[R_{t+1} + \gamma Q^\mu(\mathbf{s}_{t+1}, \mu(\mathbf{s}_{t+1}))\right]. \tag{31}$$

DDPG represents the Q-value estimates with θ^Q, and the deterministic policy with θ^μ. θ^μ is learned via the DPG given in Eq. 23, and θ^Q is learned following Eq. 31. (Note that different from DQN, where the dependence of the Q-value on $\mathbf{a}$ is represented by outputting one value for each action, the Q-network in DDPG deals with this dependence by taking the action as input for θ^Q.)

3) ***NAF*** *(Gu et al., 2016):* : Normalized advantage function offers another way to enable Q-learning in continuous action spaces with deep neural networks and is considerably simpler than DDPG. For continuous action problems, standard Q-learning is not easily directly applicable, as it requires maximizing a complex non-linear function for determining the greedy action. The key idea in NAF is to represent the Q-value function $Q(\mathbf{s}, \mathbf{a})$ in such a way that its maximum $\operatorname{argmax}_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a})$ can be easily analytically determined during the Q-learning update.

NAF uses the same techniques of *target network* and *experience replay* as DQN, but differs in the network outputs. Instead of directly outputting the Q-value estimates, its last hidden layer is connected to three output heads: θ^V, θ^μ and θ^L. θ^V represents the *state value* $V(\mathbf{s})$, while θ^μ and θ^L are used for estimating the *advantage* $A(\mathbf{s}, \mathbf{a})$; then $Q(\mathbf{s}, \mathbf{a})$ can be computed according to Eq. 25. To give a specific example, e.g, if both θ^μ and θ^L are represented with linear layers, with the number of outputs of θ^L being the square of that of θ^μ (equal to the action dimensions), then the output of θ^L is first reshaped into a matrix, from which $L(\mathbf{s}; \theta^L)$, being the lower-triangular of that matrix, is extracted, with the diagonal terms exponentiated. Then the *advantage* can be estimated by

$$A(\mathbf{s}, \mathbf{a}; \theta^\mu, \theta^L) = -\frac{1}{2}\left(\mathbf{a} - \mu(\mathbf{s}; \theta^\mu)\right)^T P(\mathbf{s}; \theta^L)\left(\mathbf{a} - \mu(\mathbf{s}; \theta^\mu)\right), \tag{32}$$

where

$$P(\mathbf{s}; \theta^L) = L(\mathbf{s}; \theta^L) L(\mathbf{s}; \theta^L)^T. \tag{33}$$

Although this representation is more restritive than a general network approximator, the greedy action for the Q-value is always directly given by $\mu(\mathbf{s}; \theta^\mu)$. An asynchronous version of NAF has also been proposed (Gu et al., 2017).

4) ***A3C*** *(Mnih et al., 2016):* : Minh et. al. proposed several asynchronous DRL algorithms. They deploy multiple actor-learners to collect experiences on multiple instances of the environment, while each actor-learner accumulates gradients calculated from its own collected samples w.r.t. its own set of network parameters θ; these gradients are used to update the weights of a shared model $\boldsymbol{\theta}$.

The most effective one, A3C (asynchronous advantage actor-critic), which has been very influential and become a standard baseline in recent DRL research, maintains a policy

representation $\pi(\mathbf{a}|\mathbf{s};\boldsymbol{\theta}^{\pi})$ and a value estimate $V(\mathbf{s};\boldsymbol{\theta}^{V})$. It uses the *advantage* function as the *score fucntion* in its policy gradient, which is estimated using a mixture of n-step returns by each actor-learner. To be more specific, each actor-learner thread spawns its own copy of the environment and collects rollouts of experiences up to $T_{\max}$ (e.g, 20) steps. After an actor-learner completes a segment of a rollout, it accumulates gradients from the experience of every time step contained in the rollout $\{0, 1, \cdots, t, \cdots, T\}$ by first estimating the *advantage* function (e.g., for time step t) according to the following formulation

$$A(\mathbf{s}_t, \mathbf{a}_t; \theta^{\pi}, \theta^{V}) = \left[\sum_{k=t}^{T-1}\left[\gamma^{k-t} R_{k+1}\right] + \gamma^{T-t} V(\mathbf{s}_T; \theta^{V}) - V(\mathbf{s}_t; \theta^{V}); \theta^{\pi}\right], \tag{34}$$

then calculating the corresponding gradients w.r.t. the its current set of network parameters θ^{π}, θ^{V}, which are then used to update the shared model $\boldsymbol{\theta}^{\pi}, \boldsymbol{\theta}^{V}$:

$$d\boldsymbol{\theta}^{\pi} \leftarrow d\boldsymbol{\theta}^{\pi} + \nabla_{\theta^{\pi}} \log \pi(\mathbf{a}_t|\mathbf{s}_t; \theta^{\pi}) A(\mathbf{s}_t, \mathbf{a}_t; \theta^{\pi}, \theta^{V}), \tag{35}$$

$$d\boldsymbol{\theta}^{V} \leftarrow d\boldsymbol{\theta}^{V} + \partial A(\mathbf{s}_t, \mathbf{a}_t; \theta^{\pi}, \theta^{V})^2 / \partial \theta^{V}. \tag{36}$$

The parallelization greatly stabilizes the update of the parameters as the samples collected by different actor-learners at the same time are much less correlated, which eliminates the requirement for keeping a *replay memory*. Also by running different exploration policies in different threads, the learners are very likely to explore different parts of the state space. Due to it being highly efficient, lightweight and conceptually simple, A3C is considered as a standard starting point in recent DRL research.

5) ***A2C*** *(Wang et al., 2016a; Wu et al., 2017):* : Some recent works found that the asynchrony in A3C does not necessarily lead to improved performance compared to the synchronous version: A2C. Different from A3C, A2C waits for each actor to finish its segment of experience before performing an update, which is averaged over all actors. This detail allows for effective GPU implementation.

6) ***GPS*** *(Levine and Koltun, 2013):* : As a model-based policy search algorithm, Guided Policy Search (GPS) is relatively sample efficient. GPS starts from guiding samples generated from some initial optimal control policies and augmented from samples generated from the current policy, from which at every iteration a set of training trajectories are sampled to optimize the current policy with supervised learning. The updated policy is then added as an additional cost term to bound the change in the policy, with which the trajectory optimization is performed again (e.g., with an LQR solver).

7) ***TRPO*** *(Schulman et al., 2015a):* : By making several approximations to the theoretically justified scheme, Schulman et al. (2015a) proposed a practical algorithm for optimizing large nonlinear policies, with guaranteed monotonic improvement.

To illustrate the algorithm, let us first define the *expected discounted cost* for an infinite horizon MDP, which replaces the reward function R in the *expected discounted return* with the cost function c:

$$\eta(\pi) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t c(\mathbf{s}_t) | \mathbf{s}_0 \sim \rho_0\right]. \tag{37}$$

In turn, we can rewrite the definitions for the state-value functions Eq. 4 and action-value functions Eq. 6 in terms of the cost function c:

$$V^{\pi}(\mathbf{s}) = \mathbb{E}_{\pi}\left[\sum_{k=t}^{\infty} \gamma^{k-t} c(\mathbf{s}_k) | \mathbf{s}_t = \mathbf{s}\right], \tag{38}$$

$$Q^{\pi}(\mathbf{s}, \mathbf{a}) = \mathbb{E}_{\pi}\left[\sum_{k=t}^{\infty} \gamma^{k-t} c(\mathbf{s}_k) | \mathbf{s}_t = \mathbf{s}, \mathbf{a}_t = \mathbf{a}\right], \tag{39}$$

and in turn, we have the advantage function:

$$A^{\pi}(\mathbf{s}, \mathbf{a}) = Q^{\pi}(\mathbf{s}, \mathbf{a}) - V^{\pi}(\mathbf{s}). \tag{40}$$

Since we are looking for a step size for the policy update that can guarantee a monotonic improvement from an old policy π_{old} to an updated policy π, it is beneficial to write the expected cost of π in terms of that of π_{old}, which leads to the following identity:

$$\eta(\pi) = \eta(\pi_{\text{old}}) + \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t A^{\pi_{\text{old}}}(\mathbf{s}_t, \mathbf{a}_t) | \mathbf{s}_0 \sim \rho_0\right]. \tag{41}$$

Before continueing, we denote the (unnormalized) discounted visitation frequencies for state $\mathbf{s}$ under policy π as $\rho^{\pi}(\mathbf{s})$, more formally,

$$\begin{aligned}\rho^{\pi}(\mathbf{s}) &= \big(P(\mathbf{s}_0 = \mathbf{s}) + \gamma P(\mathbf{s}_1 = \mathbf{s}) + \gamma^2 P(\mathbf{s}_2 = \mathbf{s}) + \cdots\big) \\ &= \sum_{t=0}^{\infty} \gamma^t P(\mathbf{s}_t = \mathbf{s}),\end{aligned} \tag{42}$$

where $\mathbf{s}_0 \sim \rho_0$, and the actions are selected following π.

Now, instead of summing over timesteps, if we sum over states, Eq. 41 can be rewritten as

$$\eta(\pi) = \eta(\pi_{\text{old}}) + \sum_{\mathbf{s}\sim\rho^{\pi}} \rho^{\pi}(\mathbf{s}) \sum_{\mathbf{a}\sim\pi} \pi(\mathbf{a}|\mathbf{s}) A^{\pi_{\text{old}}}(\mathbf{s}, \mathbf{a}). \tag{43}$$

This equation indicates that η is guaranteed to decrease or stay constant if the expected advantage at every state has a non-positive value. Since Eq. 43 is difficult to directly optimize, due to the complex dependency of ρ^{π} on π, a local approximation ignoring changes in the state visitation density induced by the changes in the policy, that matches η to the first order is introduced (the term $\eta(\pi)$ is left out here as it does not affect the solution to the underlying optimization problem):

$$L_{\pi_{\text{old}}}(\pi) = \sum_{\mathbf{s}\sim\rho^{\pi_{\text{old}}}} \rho^{\pi_{\text{old}}}(\mathbf{s}) \sum_{\mathbf{a}\sim\pi} \pi(\mathbf{a}|\mathbf{s}) A^{\pi_{\text{old}}}(\mathbf{s}, \mathbf{a}). \tag{44}$$

Standard policy gradient methods ascend on the $1st$ order gradient, where an increase on $L_{\theta_{\text{old}}}(\theta)$ does not guarantee an increase in $\eta(\pi_{\theta})$ with large step sizes, due to the approximations made above.

TRPO extends the policy improvement bound in the mixture policies setting given by Kakade and Langford (2002) to general stochastic policies, and shows that

$$\eta(\pi) \leq L_{\pi_{\text{old}}}(\pi) + CD_{\text{KL}}^{\max}(\pi_{\text{old}}, \pi), \tag{45}$$

$$\tag{46}$$

where

$$C = \frac{2\epsilon\gamma}{(1-\gamma)^2}, \tag{47}$$

$$\epsilon = \max_{\mathbf{s}} \left| \mathbb{E}_{\mathbf{a}\sim\pi} \left[A^{\pi_{\text{old}}}(\mathbf{s}, \mathbf{a}) \right] \right|. \tag{48}$$

This means that by performing the following optimization (here we denote $L_{\theta_{\text{old}}}(\theta) := L_{\pi_{\theta_{\text{old}}}}(\pi_\theta)$) with parameterized policies), we are guaranteed to improve the true objective η:

$$\underset{\theta}{\text{minimize}} \left[L_{\theta_{\text{old}}}(\theta) + CD_{\text{KL}}^{\max}(\pi_{\theta_{\text{old}}}, \pi_\theta) \right]. \tag{49}$$

However, if the penalty coefficient C, as calculated in Eq. 47, is used in practice, the step sizes will be very small.

To deal with this, TRPO first replaces the sum over actions in Eq. 44 by an importance sampling estimator (here, we only discuss the case for *single path* sampling, where π_{old} is used to generate trajectories) ($A^{\pi_{\theta_{\text{old}}}}$ is replaced by $Q^{\pi_{\theta_{\text{old}}}}$ which only changes the objective by a constant, and the Q-values are to be replaced by empirical estimates from sample averages, either *single path* or *vine*):

$$L_{\theta_{\text{old}}}(\theta) = \mathbb{E}_{\mathbf{s}\sim\rho^{\pi_{\theta_{\text{old}}}}, \mathbf{a}\sim\pi_{\theta_{\text{old}}}} \left[\frac{\pi_\theta(\mathbf{a}|\mathbf{s})}{\pi_{\theta\text{old}}(\mathbf{a}|\mathbf{s})} A^{\pi_{\theta_{\text{old}}}}(\mathbf{s}, \mathbf{a}) \right]. \tag{50}$$

Then it turns the soft constraint in Eq. 49 into the following hard constraint problem:

$$\underset{\theta}{\text{minimize}} \quad \mathbb{E}_{\mathbf{s}\sim\rho^{\pi_{\theta\text{old}}}, \mathbf{a}\sim\pi_{\theta\text{old}}} \left[\frac{\pi_\theta(\mathbf{a}|\mathbf{s})}{\pi_{\theta\text{old}}(\mathbf{a}|\mathbf{s})} Q^{\pi_{\theta\text{old}}}(\mathbf{s}, \mathbf{a}) \right], \tag{51}$$

$$\text{subject to} \quad \mathbb{E}_{\mathbf{s}\sim\rho^{\pi_{\theta\text{old}}}} \left[D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot|\mathbf{s}) || \pi_\theta(\cdot|\mathbf{s})) \right] \leq \delta. \tag{52}$$

where δ is a hyper parameter for the upper bound of the KL divergence between the old and the updated policy (e.g., $\delta = 0.01$). This constrained optimization problem is solved using a conjugate gradient followed by a line search; we refer readers to Schulman et al. (2015a) for a detailed description.

8) ***PPO*** *(Schulman et al., 2017):* : Instead of reformulating a hard constraint problem as in TRPO (Eq. 51 and 52), PPO solves the original soft constraint optimization (Eq. 49) with 1st-order SGD, adapting C according to the KL divergence. Since it is much simpler implementation-wise compared to TRPO and gives a googd performance, PPO has become the default DRL algorithm at OpenAI. A distributed version of PPO has also been proposed (Heess et al., 2017).

9) ***ACKTR*** *(Wu et al., 2017):* : The Actor Critic Kronecker-Factored Trust Region (ACKTR) is a scalable trust region natural gradient method for a actor-critic, with the Kronecker-factored approximation to the curvature. It is more computationally efficient than TRPO, and is more sample efficient than those methods taking steps in the gradient direction (e.g, A2C) rather than the natural gradient direction.

D. DRL Mechanisms

Many useful mechanisms have also been proposed that can be added on top of the aforementioned DRL algorithms. These mechanisms generally work orthogonally with the algorithms, and some can accelerate the DRL training by a large margin. Below we list several conceptually simple yet very effective ones.

- **Auxiliary Tasks** (Mirowski et al., 2016; Jaderberg et al., 2016; Levine et al., 2016; Yu et al., 2018; Riedmiller et al., 2018): Uses additional supervised or unsupervised tasks (e.g., regressing depth images from color images, detecting loop closures, predicting end-effector poses) alongside the main reinforcement learning task, to compensate for the sparse supervision signals usually provided to DRL agents.
- **Prioritized Experience Replay** (Schaul et al., 2015b): Prioritizes memory replay according to *td-error*; can be added to *off-policy* methods.
- **Hindsight Experience Replay** (Andrychowicz et al., 2017): Relabels the reward for collected experiences to make better use of failure trajectories, and effectively speed up *off-policy* methods with *binary* or *sparse* reward structures.
- **Curriculum Learning** (Bengio et al., 2009; Florensa et al., 2017; Zhang et al., 2017b): Presents the learning agent with progressively more complex task settings, such that it can grasp gradually more sophasticated skills.
- **Curiosity-driven Exploration** (Pathak et al., 2017): Augments the standard external reward with internal reward measured by intrinsic motivation.
- **Asymmetric Self-replay for Exploration** (Sukhbaatar et al., 2017): Drives exploration through an automatic curricula generated via the interplay of two versions of the same agent.
- **Noise in Parameter Space for Exploration** (Fortunato et al., 2018; Plappert et al., 2018): Pertubates network parameters to aid exploration.

E. DRL for Navigation

Autonomous navigation is one of the essential problems and challenges in mobile robotics. It can roughly be described as the ability of a robot to plan and follow a trajectory through the environment to reach a certain goal location without colliding with any obstacles in between. The recent literature has seen a growing number of proposed methods that tackle the task of autonomous navigation with DRL algorithms. Those works formulate the navigation problem as MDPs or POMDPs that first take in the sensor readings (color/depth images, laser scans, etc.) as observations and stack or augment them into states, and then search for the optimal policy that is capable of guiding the agent to navigate to goal locations in a timely and collision-free manner. Below we discuss several representative works in this category, that target the field of robotics.

Zhu et al. (2017b) input both the first-person view and the image of the target object to the A3C model, formulating a target-driven navigation problem based on the universal value function approximators (Schaul et al., 2015a). The training of

their model requires the features output from the pretrained *ResNet*-50 (He et al., 2016), and is performed in an indoor simulator (Kolve et al., 2017) where each new room is regarded as a new scene for which several scene-specific layers are added as another output head of the model. The success rate for generalizing the navigation policies to new targets one step away from the trained targets is 70%, and around is 42% for those that are two steps away. For navigation tasks with optimal solutions of 17.6 steps, Zhu et al. (2017b) achieved 210.7 average trajectory lengths after being trained on 100 million frames with an A3C agent. The trained policy was able to navigate a real robot inside an office environment after being fine-tuned on images collected from the real scene.

Zhang et al. (2017a) work on a deep *successor representation* formulation (Kulkarni et al., 2016; Barreto et al., 2017) for the Q-value function (Eqs. 15,16,17), targeting learning representations that are transferrable between related navigation tasks. Following the observation that most of the *value-based* DRL methods, such as DQN, usually learn a black-box function approximator for the optimal value functions, which makes how to transfer the knowledge gained from one task to a related task unclear, they extend on the *successor feature representation* that decouples the learning of the optimal value functions into two parts, learning task-specific reward functions, and learning task-specific features, and how those features evolve under the current task dynamics. While this representation has been shown to work well on transferring learned policies to differently scaled reward functions and changed goals in fixed environments, Zhang et al. (2017a) extend the formulations to cope with transferring policies to new environments. Both experiments in a simulated 3D maze with RGB inputs and real-world robotic experiments with depth image inputs are presented. The trained agents, either pre-trained or transferred, all achieved near-optimal performance, validating the ability of the proposed method to transfer DRL navigation policies into new environments.

The two methods mentioned above propose to learn navigation policies without a requirement for performing localization or mapping as in the traditional planning pipelines in robotics. They deal with navigating to different targets either by feeding the target image as input (Zhu et al., 2017b) or by treating it as a transfer problem (Zhang et al., 2017a). Tai et al. (2017), in contrast, propose a learning-based mapless motion planner, under the assumption that the relative position of the target w.r.t. the robot can be obtained via cheap solutions such as *wifi* or *visible light* localization, which are applicable to indoor robotic systems such as vacuum robots. The inputs for the model is 10-dimensional laser ranges, and the network outputs continuous steering commands after being trained via an asynchronous version of the DDPG. Since the simulated laser ranges and the real laser readings are quite similar, the trained model is directly generalizable to indoor office environments.

Mirowski et al. (2016) greatly improve the data efficiency and task performance of their variant of an A3C agent when learning to navigate in simulated 3D mazes, by using additional supervision signals from auxiliary tasks. In particular, the learning agent is additionally supervised by losses from depth prediction and loop closure classification. Extensive experiments are presented, validating the ability of the proposed agent to localize and to navigate between frequently changing start and goal locations.

The aforementioned methods all deal with navigation in static environments. Chen et al. (2017a) propose a DRL based systematic solution for socially aware navigation in dynamic environments with pedestrians. They extend a prior work (Chen et al., 2017b), and build a robotic system for their real-world experiment, where a differential-drive mobile robot is mounted with a Lidar for localization and three *Intel Realsense*'s for obstacle avoidance. From the sensor readings, the speed, velocity and radius of pedestrians are estimated, from which the reward (designed based on social norms) is calculated. Read robotic experiments show that the proposed method is capable of navigating agents at human walking speed in a dynamic environment with many pedestrians.

Long et al. (2017) deal with decentralized multi-agent collision avoidance with PPO. They supervised the agents with a well-shaped reward function, and test the algorithm under extensive simulated scenarios.

There is also a growing trend in the recent literature to incorporate traditional Simultaneous Localization and Mapping (SLAM) (Thrun et al., 2005) procedures, either partially or fully and embedded internally or externally, into DRL network architectures, with the intention to cultivate more sophisticated navigation capabilities in DRL agents (Stachenfeld et al., 2017; Kanitscheider and Fiete, 2017). Below we review the most representative robotics works in this promising direction.

Gupta et al. (2017a) train a Cognitive Mapping and Planning (CMP) model with *DAGGer*, which is an *imitation learning* algorithm that we will talk about in Sec. III-A. Although it dose not use DRL for training the navigation policies, we feel it fits best into this part of the discussion. CMP takes in the first-person view RGB images and applies egomotion to an internal mapper module, to encourage an egocentric multi-scale map representation to emerge out of the training process. Planning is done on this egocentric map utilizing Value Iteration Networks (VIN) (Tamar et al., 2016). Also trained with *DAGGer*, Gupta et al. (2017b) unify map-based spatial reasoning and path planning. Given the images and poses of several reference points and the starting point, as well as the pose for the goal, their proposed method is able to navigate toward the agent the desired goal location.

Zhang et al. (2017b) proposed *Neural SLAM*, based on the *Neural Map* proposed by Parisotto and Salakhutdinov (2017), where the *Neural Turing Machine* (NTM) is deployed for the DRL agent to interact with. More specifically, *Neural SLAM* embeds the motion prediction step and the measurement update step of traditional SLAM into the network architecture, biasing the write/read operations in the NTM towards SLAM operations, and treats the external memory as an internal representation of the environment for the learning agent. The whole architecture is then trained via A3C, and experiments show that the embedded structures are able to encourage the evolution of cognitive mapping capabilities of the agent, during the course of its exploration through the environment.

Khan et al. (2017) design a network architecture that

contains three components: a convolution network for feature extraction, a planner module to pre-plan in the environment, and a controller module that learns to selectively store past information that could be useful for planning.

Bruce et al. (2017) propose a method that enables zero-shot transfer for a mobile robot to learn to navigate to a fixed goal in an environment with variations unseen during the training. They introduce *interactive replay*, in which a rough world model is built from a single traversal of the environment. The agent is then able to interact with this world model to generate a large number of diverse trajectories, which can substantially reduce the number of real experiences needed.

Chaplot et al. (2018) introduce a network structure mimicking the Bayesian filtering process with a specially designed perception model. It takes as input the agent's observation, and outputs a likelihood map, inside which the belief is propagated through time following the classic filtering process used for localization in robotics (**?**).

Inspired by graph-based SLAM algorithms (Thrun et al., 2005; Kümmerle et al., 2011), Parisotto et al. (2018) embed the global pose optimization into the network architecture design for their *Neural Graph Optimizer*, which is composed of a local pose estimation model, a pose selection module and a graph optimization process. Savinov et al. (2018) introduce a memory architecture, *Semi-parametric Topological Memory*, for navigation in unseen environments. It contains a non-parametric graph with nodes representing locations in the environment, and a parametric deep network retrieving nodes from the graph based on observations.

F. DRL for Manipulation

In terms of manipulation, the tasks being considered for evaluating DRL algorithms are more standardized in the recent literature (Lillicrap et al., 2015; Schulman et al., 2015a; Mnih et al., 2016; Heess et al., 2017; Wu et al., 2017). Most of such works benchmark the proposed algorithms on standard tasks, including *reaching*, *pushing*, *pick-and-place*, etc., using the *MuJoCo* simulator (Todorov et al., 2012). Below we focus on the works that are presented with real robotic experiments.

Gu et al. (2017) propose an asynchronous version of NAF. Taking in the low dimensional states as inputs (joint angles, end effector poses, as well as their time derivatives, and the pose of the target), in addition to well-shaped reward signals, it allows the robot to learn a real-world door opening task in about 2.5 hours in a completely end-to-end manner, achieving a 100% success rate.

Levine et al. (2016) successfully train deep visuomotor policies with GPS, a model-based approach. Their proposed visuomotor policy network takes as input monocular RGB images and passes them through several convolutional layers and a spatial soft argmax layer, which are then concatenated with the robot configurations (joint angles, end effector poses). These representations are then passed through several fully connected layers and used to predict the corresponding motor torques. Various experiments on a *PR2* robot (with a 7-DOF arm) such as hanging a coat hanger on a clothes rack, inserting a block into a shape sorting cube, or screwing on a bottle cap have demonstrated to validate the effectiveness of the approach. This method, however, requires a known and fully observed state space, which could limit its potential use cases.

Model-based DRL methods are also utilized by Finn et al. (2016) and Tzeng et al. (2015), learning useful state representations for generating successful control policies. Fu et al. (2016) proposed one-shot learning of manipulation skills through model-based reinforcement learning by leveraging the neural network priors as a dynamic model. Learning dexterous manipulation skills with multi-fingered hands, for which model-based (Kumar et al., 2016; Gupta et al., 2016) and model-free (Popov et al., 2017) DRL algorithms have been proposed and demonstrated in real robotic experiments, is quite challenging.

While many works have carefully designed their reward structure to guide reinforcement learning, Riedmiller et al. (2018) propose a method to speed up learning from only binary or sparse rewards, under the observation that well-shaped rewards can often bias the learned control policy into potentially suboptimal directions. In contrast when only sparse reward signals are provided to the agent, the learner can discover novel and potentially preferable solutions. To achieve this, alongside the policy learning for the main task, Riedmiller et al. (2018) learn policies (which they refer to as intentions) for a set of semantically grounded auxiliary tasks, whose supervision signals can be easily obtained by the activation of certain sensors. Then a scheduling policy is learned to sequence the intention-policies. Their proposed algorithm is able to learn to solve challenging manipulation tasks from scratch, such as stacking two blocks into a tower or cleaning up a desk by putting objects desk into a box with a lid that can be opened, with a 9-DOF robot arm. Moreover, in their real-world experiments, a single robot arm learns a lifting task in about 10 hours.

G. The Reality Gap: From Simulation to the Real World

Although DRL offers a general framework for agents to learn high-dimensional control policies, it typically requires several millions of training samples. This makes it infeasible to train DRL agents directly in real-world scenarios, since real robotic control experiences are relatively expensive to obtain. As a consequence, DRL algorithms are generally trained in simulated environments, then transferred to the real world and deployed onto real robotic systems. This brings about the problem of the *reality gap*, which refers to the discrepancies in lighting conditions, noise patterns, textures, etc., between synthetic renderings and real-world sensory readings. The *reality gap* imposes major challenges for generalizing the DRL policies trained in simulation to real scenarios.

Since the problem of the *reality gap* is most severe in the visual domain, in that the aforementioned discrepancies are most significant between rendered color images and real color camera readings, some robotics works have proposed to circumvent this problem by using other input modalities whose domain variations are less distinct, such as depth images (Zhang et al., 2017a) or laser ranges (Tai et al., 2017). However, bridging the *reality gap* in the visual domain is of

great importance and remains one of the focuses of recent works. Below, we review methods that deal with the *reality gap* for visual control.

1) ***Domain Adaptation****:* In the visual domain, *domain adaptation* can also be referred to as *image-to-image translation*, which focuses on translating images from a source domain to a target domains, It can be considered as the method of choice in the recent literature to tackle the *reality gap* for visual control. The *domain confusion loss*, as proposed by Tzeng et al. (2014), is another solution that learns a semantically meaningful and domain invariant representation. However, minimizing the *domain confusion loss* requires that the data from both the source and the target domain are available from the beginning of the whole learning pipeline, which might not be as flexible in the robotics context.

In the following, we first formalize the *domain adaptation* problem, then continue to introduce several of the most general methods that require the least human intervention and are most directly applicable to robotics control tasks.

Consider visual data sources from two domains: $\mathbf{X}$ (e.g., synthetic images rendered by a simulator; $\mathbf{x} \sim p_{\text{sim}}$, where p_{sim} represents the simulated data distribution) and $\mathbf{Y}$ (e.g., real sensory readings from the onboard color camera of a mobile robot; $\mathbf{y} \sim p_{\text{real}}$, where p_{real} represents the distribution of the real color image readings). As we have just discussed, DRL agents are typically trained in the synthetic domain $\mathbf{X}$, then deployed onto real robotic platforms to perform control tasks in the real-world domain $\mathbf{Y}$. *Domain adaptation* methods aim to learn a mapping between these two domains.

GANs: Most of the *domain adaptation* works are based on Generative Adversarial Networks (GANs) (Goodfellow et al., 2014; Radford et al., 2015; Arjovsky et al., 2017). When learning a GAN model, a generator G and a discriminator D are trained in an adversarial manner. In the context of *domain adaptation* for visual inputs, the generator G takes images from the source domain, and tries to generate output images matching those from the target domain, while the discriminator D learns to tell the generated target images and the real target images apart.

CycleGAN (Zhu et al., 2017a): Zhu et al. propose one of the most popular unsupervised *domain adaptation* methods in the recent literature, Zhu et al. (2017a) proposed a simple yet very effective formulation that does not require paired data from the two domains of interest. Observing that the mapping from the source domain to the target domain, $G_{\mathbf{Y}} : \mathbf{X} \to \mathbf{Y}$, is highly under-constrained, *CycleGAN* proposes to add a cycle-consistent loss to enforce that a reverse mapping from the target domain back to the source domain exists: $G_{\mathbf{X}} : \mathbf{Y} \to \mathbf{X}$. More formally, *CycleGAN* learns two generative models to map between domains $\mathbf{X}$ and $\mathbf{Y}$: $G_{\mathbf{Y}}$ with its discriminator $D_{\mathbf{Y}}$, and $G_{\mathbf{X}}$ with its discriminator $D_{\mathbf{X}}$, by training two GANs simultaneously:

$$\mathcal{L}_{\text{GAN}_{\mathbf{Y}}}(G_{\mathbf{Y}}, D_{\mathbf{Y}}; \mathbf{X}, \mathbf{Y}) = \mathbb{E}_{\mathbf{y}}\left[\log D_{\mathbf{Y}}(\mathbf{y})\right] + \mathbb{E}_{\mathbf{x}}\left[\log(1 - D_{\mathbf{Y}}(G_{\mathbf{Y}}(\mathbf{x})))\right], \tag{53}$$

$$\mathcal{L}_{\text{GAN}_{\mathbf{X}}}(G_{\mathbf{X}}, D_{\mathbf{X}}; \mathbf{Y}, \mathbf{X}) = \mathbb{E}_{\mathbf{x}}\left[\log D_{\mathbf{X}}(\mathbf{x})\right] + \mathbb{E}_{\mathbf{y}}\left[\log(1 - D_{\mathbf{X}}(G_{\mathbf{X}}(\mathbf{y})))\right], \tag{54}$$

on top of which two sets of *cycle consistency loss* are added to constrain the two mappings:

$$\mathcal{L}_{\text{cyc}_{\mathbf{Y}}}(G_{\mathbf{X}}, G_{\mathbf{Y}}; \mathbf{Y}) = \mathbb{E}_{\mathbf{y}}\left[||G_{\mathbf{Y}}(G_{\mathbf{X}}(\mathbf{y})) - \mathbf{y}||_1\right], \tag{55}$$

$$\mathcal{L}_{\text{cyc}_{\mathbf{X}}}(G_{\mathbf{Y}}, G_{\mathbf{X}}; \mathbf{X}) = \mathbb{E}_{\mathbf{x}}\left[||G_{\mathbf{X}}(G_{\mathbf{Y}}(\mathbf{x})) - \mathbf{x}||_1\right]. \tag{56}$$

The full objective of *CycleGAN* then adds up to (λ denotes the weighting of the *cycle consistency loss*)

$$\begin{aligned}\mathcal{L}(G_{\mathbf{Y}}, G_{\mathbf{X}}, D_{\mathbf{Y}}, D_{\mathbf{X}}; \mathbf{X}, \mathbf{Y}) = \\ \mathcal{L}_{\text{GAN}_{\mathbf{Y}}}(G_{\mathbf{Y}}, D_{\mathbf{Y}}; \mathbf{X}, \mathbf{Y}) \\ + \mathcal{L}_{\text{GAN}_{\mathbf{X}}}(G_{\mathbf{X}}, D_{\mathbf{X}}; \mathbf{Y}, \mathbf{X}) \\ + \lambda_{\text{cyc}}\mathcal{L}_{\text{cyc}_{\mathbf{Y}}}(G_{\mathbf{X}}, G_{\mathbf{Y}}; \mathbf{Y}) \\ + \lambda_{\text{cyc}}\mathcal{L}_{\text{cyc}_{\mathbf{X}}}(G_{\mathbf{Y}}, G_{\mathbf{X}}; \mathbf{X}),\end{aligned} \tag{57}$$

which corresponds to the following optimization problem:

$$G^*_{\mathbf{Y}}, G^*_{\mathbf{X}} = \arg \min_{G_{\mathbf{Y}}, G_{\mathbf{X}}} \max_{D_{\mathbf{Y}}, D_{\mathbf{X}}} \mathcal{L}(G_{\mathbf{Y}}, G_{\mathbf{X}}, D_{\mathbf{Y}}, D_{\mathbf{X}}). \tag{58}$$

This conceptually simple method works surprisingly well in practice, especially in domains with relatively few semantic types (e.g., when the source domain images contain only horses and background, and the target domain images contain only zebras and background), where it is less challenging for the algorithm to find the matching semantics between the two domains (e.g., horse $\leftrightarrow$ zebra). However, the results of *CycleGAN* on translating between more complex data distributions containing many more semantic types, such as between urban street scenario images and their corresponding semantic labels, are not as satisfactory, in that the generators often permute the labels for some semantics.

CyCADA (Hoffman et al., 2017): The semantic consistency loss proposed in *CyCADA* offers a good solution to learning the mapping between more complex data distributions with relatively more semantic types. To be more specific, in *CyCADA*, a semantic segmentation network f is first trained in the domain where semantic labels are available (e.g., $f_{\mathbf{X}}$ for the synthetic domain $\mathbf{X}$). (This is applicable for the *domain adaptation* between the simulated domain $\mathbf{X}$ and the real-world domain $\mathbf{Y}$ in the context of robotics, since many recent robotics simulators provide the ground truth semantic maps of the rendered images, while the labels for the real images are expensive to obtain.) Then this semantic segmentation network is used to constrain the semantic consistency between the input and the translated output images of the generators:

$$\mathcal{L}_{\text{sem}_{\mathbf{Y}}}(G_{\mathbf{Y}}; \mathbf{X}, f_{\mathbf{X}}) = \text{CrossEnt}(f_{\mathbf{X}}(\mathbf{X}), f_{\mathbf{X}}(G_{\mathbf{Y}}(\mathbf{X}))), \tag{59}$$

$$\mathcal{L}_{\text{sem}_{\mathbf{X}}}(G_{\mathbf{X}}; \mathbf{Y}, f_{\mathbf{X}}) = \text{CrossEnt}(f_{\mathbf{X}}(\mathbf{Y}), f_{\mathbf{X}}(G_{\mathbf{X}}(\mathbf{Y}))), \tag{60}$$

where $\text{CrossEnt}(S_{\mathbf{X}}, f_{\mathbf{X}}(\mathbf{X}))$ represents the cross-entropy loss between the semantics of X predicted by the pretrained semantic segmentation network $f_{\mathbf{X}}$ and the ground truth label $S_{\mathbf{X}}$. The semantic consistency losses are then added to the *CycleGAN* objective (Eq. 58).

2) ***Domain Adaptation for Visual DRL Policies****:* While many extensions and variants have been proposed for *image-to-image translation* in the computer vision literature, here we focus on those *domain adaptation* methods that specifically

itarget transferring DRL control policies from simulation to real scenarios.

For manipulation tasks, Bousmalis et al. (2017) deal with the *reality gap* by adapting synthetic images to the realistic domain before feeding them into the DRL policy network during the training phase. However, the additional adaptation step required for every training iteration could significantly slow down the whole learning pipeline. Tobin et al. (2017) proposed to randomise the lighting conditions, viewing angles and textures of objects during the training phase of the DRL policies in simulation, in the hope that after being exposed to enough variations, the learned model can naturally generalize to real-world scenarios. However, this method can only be applied to simulators where such randomization can be easily achieved at a low cost, which is not the case for most of the popular robotic simulators. Moreover, there is no guarantee that for a random real-world scenario, its visual modality can be covered by the randomized simulations. Similarly, randomizing the dynamics of the simulator during training has also been proposed (Peng et al., 2017) to bridge the *reality gap*. Rusu et al. (2017) propose to progressively adapt the learned deep features and representations from the synthetic domain to the real-world domain. However, this method still requires going through an expensive DRL control policy training phase (although this procedure can be greatly sped up by initial training in the simulator) for each new visually different real-world scene.

The aforementioned methods realize *domain adaptation* via the *sim-to-real* direction, meaning that they either translate the synthetic images to the real-world domain during the training of DRL policies, or adapt the deep features of the simulated domain to those of the realistic domain. However, the DRL policy learning and the adaptation of the policies are entangled in this line of methods.

The recently proposed model of *VR Goggles* (Zhang et al., 2018) decouples the two components of policy learning and policy adaptation, by tackling the *reality gap* from the *real-to-sim* direction, which requires no extra transfer steps during the expensive training of DRL policies. Specifically, the *VR Goggles* deal with the *reality gap* only during the actual deployment phase, by translating real-world sensor reading streams back to the simulated domain, so as to adapt the unseen or unfamiliar characteristics of the real scenes to the synthetic features, which the agent has already learned well how to deal with, to *make the robot feel at home*. To constrain the consistency between the generated subsequent frames, a *shift loss* is added to the optimization objective, which is inspired by the *artistic style transfer for videos* literature (Ruder et al., 2017). This method is validated in transferring DRL navigation policies, which could be considered more challenging than manipulation tasks, since the environments the navigation agents operate in are typically at much larger scales than the confined workspace of manipulators.

Both results of outdoor and indoor scene adaptation have been presented. For the outdoor experiment, the synthetic data is collected from the *CARLA* simulator (Dosovitskiy et al., 2017) which provides the ground truth semantic labels, and the real world data is gathered from the *RobotCar* dataset (Maddern et al., 2017). The semantic consistency loss is added for the outdoor scenario, with a semantic segmentation network trained using the *DeepLab* model (Chen et al., 2016). The semantic consistency is critical for outdoor scenes containing various semantic types, without such a constraint, permutation of semantics occurs. It is also critical for situations where the model fails to generate a virtual car at the position at which there is a real car in the real image (This kind of performance is as reported by Yang et al. (2018) whithout constraining the semantic consistency), which could potentially lead to accidents in self-driving scenarios.

For indoor scenes, the semantic loss is not added, as the simulated domain *Gazebo* (Koenig et al., 2004) does not provide ground truth labels, and also the real scene, which is a real office environment, contains relatively fewer semantic types. A real robot (*Turtlebot3 Waffle*) is deployed in the office environment and feed its sensor readings (captured by a *RealSense R200 camera*) to the *VR Goggles* model. The translated *Gazebo* images are then fed to the DRL policy network to give control commands. The *VR Goggles* offer a lightweight and flexible solution for transferring DRL visual control policies from simulation to the real world, and should also be applicable to manipulation tasks.

H. Simulation Platforms

As mentioned before, DRL algorithms, at their current state, are in general not sample efficient enough to be directly trained on real robotic platforms. Thus robotics simulators are utilized for the initial training of DRL policies. Here we review several of the most widely used simulation platforms that are suitable for DRL training.

We summarize the most commonly used simulators in Table I, listing their available sensor observation types and their target use cases.

III. Imitation Learning

DRL offers a formulation for control skills acquisition. However, relying on learning from trial and error, DRL methods typically require a significant amount of system interaction time. Also, a carefully designed well-shaped reward structure is necessary to guide the search of optimal policies, which can often be non-trivial in complex scenarios.

Imitation learning, as an alternative to learning control policies, guides the policy search, not by hand-designed reward signals, but by providing the learning agent with experts' demonstrations (Bagnell, 2015). It offers a paradigm for agents to learn successful policies in fields where people can easily demonstrate the desired behavior but find it difficult to hand program or hardcode the correct cost or reward function. This is especially useful for humanoid robots or manipulators with high degrees of freedom.

Perhaps the most simple method of *imitation learning* is addressing it as a standard *supervised learning* problem. But as we have discussed, as a method for learning policies to make sequential control decisions, *imitation learning* cannot be conducted effectively by directly applying the classical *supervised learning* approaches. We emphasize the most critical distinctions below:

TABLE I
ROBOTIC SIMULATORS.

Simulator	Modalities	Framerate	Target Use Case
Gazebo (Koenig et al., 2004)	Sensor Plugins	10s+FPS	General Purposes
Vrep (Rohmer et al., 2013)	Sensor Plugins	10s+FPS	General Purposes
Airsim (Shah et al., 2017)	Depth/Color/Semantics	20s+FPS	Autonomous Driving
Carla (Dosovitskiy et al., 2017)	Depth/Color/Semantics	30s+FPS	Autonomous Driving
Torcs (You et al., 2017)	Color/Semantics	100s+FPS	Autonomous Driving
AI2-Thor (Kolve et al., 2017)	Color	100s+FPS	Indoor Navigation
Minos (Savva et al., 2017)	Depth/Color/Semantics	100s+FPS	Indoor Navigation
House3D (Wu et al., 2018)	Depth/Color/Semantics	600s+FPS	Indoor Navigation

Independent VS. Compounding Errors: Standard *supervised learning* assumes that the predictions made by the learning agents do not affect the environment in which they operate; hence the data distribution they are to encounter is assumed to be the same as what they have experienced. However, although the learning errors are independent for each sample in *supervised learning*, they are *compounded* in *imitation learning*. This is due to the fact that the standard *supervised learning* algorithms are only expected to do well over samples that are drawn from the same distribution as they have been trained on. This i.i.d. assumption, however,is badly violated in *imitaiton learning*, in which an early error could potentially *cascade* to a sequence of mistakes, carried out by the control decisions that are made sequentially by the learning agent.

Single-Timestep VS. Multi-Timestep Decisions: *Supervised learning* agents are only capable of learning *reactive policies*, since they completely ignore the temporal dependence between subsequent decisions, which leads to *myopic* strategies. In contrast, for making informative decisions, classical planning approaches in robotics reason far into the future (but often require sophisticatedly designed cost functions). Also, a naive imitation of the experts' demonstrations often misses the true learning objective: instead of copying the demonstrated behaviors given by the experts, the actual goal of *imitation learning* is in some cases quite different and inexplicitly optimized in the demonstrations, such as to increase the success rate of accomplishing a specific task, to minimize the probability of colliding with obstacles, or to minimize the total travel cost.

In the following, we proceed by going through the three most common approaches of *imitation learning*, which address the above issues from different perspectives, and introduce representative works in robotics for each.

A. *Behavior Cloning*

Behavior cloning tackles the problem of *imitation learning* in a supervised manner, by directly learning the mapping between the input observations and their corresponding actions, which are given by the expert policy. This simple formulation can give a satisfactory performance when there is enough training data, but will lead to *compounding errors*, as we have just discussed. One of the most well-known algorithms to compensate for this is *DAGGer* (in which *DAGG* stands for *Data AGGregation*) (Ross et al., 2011), which interleaves execution and learning. To be more specific, in the ith iteration of *DAGGer*, the current learned policy π_{i-1} will be executed to collect experiences. Then those newly recorded observations will be relabeled by the expert policy π_{E}. These corrected new experiences D_i will be added to the existing dataset D, on which a new policy π_i is trained. This interaction between execution and learning halts the error compounding and bounds the expected error to that in the standard *supervised learning* setting.

Due to its simple formulation, behavior cloning has been widely studied and applied in robotics control problems.

We start with the literature in the field of navigation and self-driving imitation. Bojarski et al. (2016) learn a direct mapping from raw first-person view color images to steering commands, on a training dataset collected by driving on a wide variety of roads and in diverse weather and lighting conditions, which in total adds up to 72 hours of driving data. Tai et al. (2016) drive an indoor mobile robot autonomously through a dataset based on joystick commands from human demonstrator. A depth visual image is taken as the only input in their implementation. Giusti et al. (2016) train a deep network to determine actions that can keep a quadrotor on a trail, by learning on single monocular images collected from the robot's perspective. Eight hours of video is captured using three GoPros mounted on the head of a hiker, with one pointing to the left, one to the right, and one straight ahead. The optimal actions for the collected images can then be easily labeled; e.g., the quadrotor should turn right when facing an image collected from the left-facing camera. Codevilla et al. (2017) observes that the pure *behavior cloning* assumption could break under certain situations, such as when a driver approaches an intersection. The driver's subsequent actions cannot be fully explained by the observations, since they are additionally affected by the driver's internal throughouts, such as the intended destination. To address this, a conditional *imitation learning* method is proposed to additionally constrain the *imitation learning* additionally on a representation

of the expert's intention, so as to resolve the ambiguity in the perceptuomotor mapping. Both simulated and real-world experiments are conducted, in which the synthetic dataset is collected in the simulated self-driving environment *Carla* (Dosovitskiy et al., 2017) and a real-world dataset from remote controlling a robotic truck in a residential area, each of which contains two hours of driving time.

In terms of *imitation learning* for manipulation, a recently proposed line three works presents and improves on *one-shot imitation learning*: from taking low dimensional states and expert action pairs as demonstrations (Duan et al., 2017), to learning from demonstrations of raw visual images paired with actions (Finn et al., 2017b), and finally arriving at the current state of learning from human demonstration videos without labeled actions (Yu et al., 2018). Below we discuss these methods in more detail.

Duan et al. (2017) present the imitation agent with pairs of demonstrations for each iteration during training, in which the network takes as input the firsth demonstration and a state sampled from the second demonstration. The network is then trained using behavior cloning losses to predict the corresponding action of that sampled state. The concrete example used in their problem setting is a distribution of block stacking tasks, in which the goal is to control a robot arm to stack various numbers of cubic blocks into configurations specified by the user. Each observation is a list of the relative positions of the blocks w.r.t. the gripper, and information indicating whether the gripper is open or closed. Several architectural designs, such as temporal dropout and convolutions, neighborhood attention, are incorporated into their training pipeline to cope with variable-dimensional and potentially long sequences of inputs. In their experiments, the performance of pure *behavior cloning* achieves the same level of performance as training with *DAGGer*, suggesting that at least for this specific block-stacking task, the interactive supervision in *DAGGer* might not necessarily lead to a performance gain.

Finn et al. (2017b) and Yu et al. (2018) both extend the *Model-Agnostic Meta-Learning* (MAML) method (Finn et al., 2017a), which we will briefly review here before proceeding. The objective of MAML is to learn a model, such that, after being trained on a variety of learning tasks, it is able to learn to solve new tasks with only a small number of training samples. Formally, this model of interest is denoted as f_θ with weights θ, and the meta-learning is considered over a distribution of tasks $p(\mathcal{T})$. The model parameters will be updated from θ to θ_i', when adapting to a new task $\mathcal{T}_i$. This update is performed using gradient descent on task $\mathcal{T}_i$:

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta), \tag{61}$$

where α denotes a step size, and $\mathcal{L}$ represents a behavior cloning loss function (e.g., mean squared error for continuous actions, cross-entropy loss for discrete actions). After the updated θ_i' is obtained, its performance is optimized w.r.t. θ across tasks sampled from $p(\mathcal{T})$, leading to the following meta-learning objective:

$$\min \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)}), \tag{62}$$

which is performed via SGD such that θ is updated as follows:

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}), \tag{63}$$

where β is the meta step size.

Here, the meta-optimization is performed over θ, while the loss is computed using the updated parameters θ'. This objective will help to find model parameters that are sensitive to changes in the task, such that small changes in the parameters could lead to large improvements in the performance on any task sampled from $p(\mathcal{T})$.

Based on the formulation of MAML, Finn et al. (2017b) learn policies that can be quickly adapted to new tasks using a single demonstration. Here each observation input into the model contains a color image from the robot's perspective, and the robot configurations (joint angles, end-effector poses). While both Duan et al. (2017) and Finn et al. (2017b) use only robot demonstrations throughout training and testing, Yu et al. (2018) is able to cope with domain shift by learning from both robot and human demonstrations, in which the human demonstrations are not labeled with expert actions. After meta-learning, the proposed method is capable of learning from human videos. To cope with the unlabelled human demonstrations, an adaptation loss function $\mathcal{L}_\psi$ is learned alongside the meta-learning objective. During training, human demonstrations are used to compute the updated policy parameters θ_i' with the gradients calculated using $\mathcal{L}_\psi$. Then the performance of θ_i' is evaluated using the behavior cloning loss to update both θ and ψ. Note that all the robot demonstrations are collected via teleoperation (Zhang et al., 2017c).

A recent work from Eitel et al. (2017) introduces a model that is able to propose push actions based on over-segmented RGB-D images, in order to separate unknown objects in cluttered environments.

B. *Inverse Reinforcement Learning*

Inverse reinforcement learning (IRL) frames *imitation learning* as solutions to MDPs, thus reducing the problem of learning to the problem of recovering a utility function that makes the demonstrated behavior (near-)optimal. After this utility function is obtained, reinforcement learning procedures can be performed on top to search for optimal policies. A representative IRL method, the Maximum Entropy IRL (Ziebart et al., 2008), fits a cost function from a family of functions $\mathcal{C}$ to optimize the following objective:

$$\underset{c \in \mathcal{C}}{\operatorname{argmax}} \left(\min_{\pi \in \prod} -H(\pi) + \mathbf{E}_\pi \left[c(\mathbf{s}, \mathbf{a}) \right] \right) - \mathbf{E}_{\pi^{\mathrm{E}}} \left[c(\mathbf{s}, \mathbf{a}) \right], \tag{64}$$

where $\prod$ denotes the family of policies.

In robotics, the formulation of IRL offers an efficient solution for learning policies for socially compliant navigation (Okal and Arras, 2016; Pfeiffer et al., 2016; Kretzschmar et al., 2016), where the agent needs to not only avoid collisions with static obstacles but also to behave in a socially compliant manner. Thus, the underlying cost function is non-trivial to hand-design, but the hebaviors are easy to demonstrate.

Wulfmeier et al. (2015) extends Maximum Entropy IRL under the deep learning context, utilizing fully convolutional neural networks as the approximator for learning the reward function. The proposed algorithm is successfully deployed for learning the cost map in urban environments, from a dataset of driving behaviors demonstrated by human experts.

C. *Generative Adversarial Imitation Learning*

The learning process of IRL can be indirect and slow. Inspired by Generative Adversarial Networks (GANs) (Goodfellow et al., 2014), Ho and Ermon (2016) propose Generative Adversarial Imitation Learning (GAIL), which surpasses the intermediate step of learning a reward function and is capable of directly learning a policy from expert demonstrations. To be more specific, in the GAIL model, a generator π_θ with parameters θ is trained to generate state-action ($\mathcal{S} \times \mathcal{A}$) pairs matching those of the expert demonstrations, while the discriminator D_ω learns to tell apart the generated policy π_θ from the expert (demonstrated) policy π^{E}. The GAIL optimization objective is defined as follows:

$$\mathbf{E}_{\pi_\theta}\left[\log(D(\mathbf{s},\mathbf{a}))\right] + \mathbf{E}_{\pi^{\mathrm{E}}}\left[\log(1 - D(\mathbf{s},\mathbf{a}))\right] - \lambda H(\pi_\theta), \tag{65}$$

where $H(\pi_\theta)$ denotes the causal entropy. The training of GAIL interleaves between updating parameters ω of the discriminator D_ω to maximize Eq. 65, and utilizing DRL techniques such as TRPO to minimize Eq. 65 w.r.t. the parameters θ of the generator π_θ. The scores given out by the discriminator for the generated experiences are regarded as costs for those state-action pairs for TRPO. Several extensions of GAIL have also been proposed (Baram et al., 2016; Wang et al., 2017).

In the field of navigation, Li et al. (2017) successfully apply GAIL in simulated autonomous vehicle navigation scenarios with raw visual input. Tai et al. (2018) learn a socially compliant navigation policy through GAIL, based on raw depth input, and demonstrate the learned behaviors in real robotics experiments.

For manipulation, Stadie et al. (2017) extend the GAIL formulation with ideas from domain confusion loss (Tzeng et al., 2014), and successfully utilize it to train agents to imitate third-person demonstrations, by learning a domain-agnostic representation of the agent's observations.

IV. Challenges and Open Research Questions

Utilizing deep learning techniques for learning control for robotics tasks has shown great potential. Yet, there still remain many challenges for scaling up and stabilizing the aforementioned algorithms to meet the requirements of operating robotics systems in real-world applications. We list the critical challenges and the corresponding future research directions.

- **Sample Efficiency**: Gathering experiences by interacting with the environment for *deep reinforcement learning*, or collecting expert demonstrations for *imitation learning*, are both expensive procedures, in terms of executing control commands on real robotics systems. Thus, designing sample efficient algorithms is of critical importance.
- **Strong Real-time Requirements**: A single forward pass of very deep networks with millions of parameters could be relatively slow if not equipped with special computation hardware and might not meet the real-time requirement for controlling real robotics systems. Learning compact representations for dexterous policies is preferable.
- **Safety Concerns**: Real robotics systems, such as mobile robots, quadrotors or self-driving cars, are expected to operate in environments that could be highly dynamic and potentially dangerous. Also, unlike a wrong prediction from a perception model, which would not cascade or affect the physical robotic systems or the environment, a single false output might lead to a serious accident. Thus, attention should be paid to include practical considerations to bound the uncertainty of the possible outcomes when deploying control policies on real autonomous systems.
- **Stability, Robustness and Interpretability**: DRL algorithms could be relatively unstable, and their performance might deviate a lot between configurations that only differ slightly from each other (Henderson et al., 2017). To overcome this problem, gaining more insight into the learned representations and the policies, could be beneficial for detecting adversarial scenarios to prevent robotic systems from safety threats.
- **Lifelong Learning**: The visual appearance of the environments that autonomous agents operate in cab vary dramatically during different seasons of the year, or even different times of day, which could hinder the performance of the learned control policies. Hence the ability of continuing to learn to adapt to environmental changes as well as preserving the solutions to the already experienced scenarios could be of critical value.
- **Generalization Between Tasks**: Most of the aforementioned algorithms are designed to excel in a particular task, which is not ideal, as intelligent robotic systems are expected to be capable of carrying out a set of tasks, with a minimal total training time for all considered tasks.

In contrast, with the rapid development of deep learning, several research directions are gaining much attention for robotics.

- **Unifying Reinforcement Learning and Imitation Learning**: Several recent works (Večerík et al., 2017; Nair et al., 2017; Gao et al., 2018; Zhu et al., 2018) have introduced algorithms that unify reinforcement learning and imitation learning such that the learning agent can benefit from both expert demonstrations and interactions with the environment. This setup can be beneficial for learning control, as pure DRL algorithms are, in general, relatively expensive to train, while learning purely by imitating demonstrated behaviors can restrict or bias the control policy in potentially suboptimal directions. Thus, the method of using demonstrations to kick-start the policy learning, then applying DRL methods to adjust the learned policy, can potentially lead to advanced performance.

- **Meta-learning**: Finn et al. (2017a) and Nichol and Schulman (2018) propose methods that can effectively lead the policy search to find parameters that can be adapted to give good performance on a new task with only a small number of training examples of the new task. Such formulations could be very beneficial, and have the potential to learn universal and robust policies.

V. Conclusion

In this paper, we give a brief overview of deep learning solutions for robotics control tasks, focusing mainly on *deep reinforcement learning* and *imitation learning* algorithms. We mainly introduce the formulations for each learning paradigm and the corresponding representative works in robotics. Finally, We discuss the challenges and potential future research directions.

References

Andrychowicz M, Crow D, Ray A, Schneider J, Fong R, Welinder P, McGrew B, Tobin J, Abbeel OP and Zaremba W (2017) Hindsight experience replay. In: *Advances in Neural Information Processing Systems*. pp. 5055–5065.

Arjovsky M, Chintala S and Bottou L (2017) Wasserstein gan. *arXiv preprint arXiv:1701.07875* .

Bagnell JA (2015) An invitation to imitation. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST.

Baram N, Anschel O and Mannor S (2016) Model-based adversarial imitation learning. *arXiv preprint arXiv:1612.02179* .

Barreto A, Dabney W, Munos R, Hunt JJ, Schaul T, Silver D and van Hasselt HP (2017) Successor features for transfer in reinforcement learning. In: *Advances in Neural Information Processing Systems*. pp. 4058–4068.

Bengio Y, Louradour J, Collobert R and Weston J (2009) Curriculum learning. In: *Proceedings of the 26th annual international conference on machine learning*. ACM, pp. 41–48.

Bojarski M, Del Testa D, Dworakowski D, Firner B, Flepp B, Goyal P, Jackel LD, Monfort M, Muller U, Zhang J et al. (2016) End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316* .

Bousmalis K, Irpan A, Wohlhart P, Bai Y, Kelcey M, Kalakrishnan M, Downs L, Ibarz J, Pastor P, Konolige K et al. (2017) Using simulation and domain adaptation to improve efficiency of deep robotic grasping. *arXiv preprint arXiv:1709.07857* .

Bruce J, Sünderhauf N, Mirowski P, Hadsell R and Milford M (2017) One-shot reinforcement learning for robot navigation with interactive replay. *arXiv preprint arXiv:1711.10137* .

Chaplot DS, Parisotto E and Salakhutdinov R (2018) Active neural localization. *arXiv preprint arXiv:1801.08214* .

Chen LC, Papandreou G, Kokkinos I, Murphy K and Yuille AL (2016) Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915* .

Chen YF, Everett M, Liu M and How JP (2017a) Socially aware motion planning with deep reinforcement learning. *arXiv preprint arXiv:1703.08862* .

Chen YF, Liu M, Everett M and How JP (2017b) Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In: *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, pp. 285–292.

Codevilla F, Müller M, Dosovitskiy A, López A and Koltun V (2017) End-to-end driving via conditional imitation learning. *arXiv preprint arXiv:1710.02410* .

Dayan P (1993) Improving generalization for temporal difference learning: The successor representation. *Neural Computation* 5(4): 613–624.

Deisenroth MP, Neumann G, Peters J et al. (2013) A survey on policy search for robotics. *Foundations and Trends® in Robotics* 2(1–2): 1–142.

Deng L (2014) A tutorial survey of architectures, algorithms, and applications for deep learning. *APSIPA Transactions on Signal and Information Processing* 3.

Dosovitskiy A, Ros G, Codevilla F, López A and Koltun V (2017) Carla: An open urban driving simulator. *arXiv preprint arXiv:1711.03938* .

Duan Y, Andrychowicz M, Stadie B, Ho OJ, Schneider J, Sutskever I, Abbeel P and Zaremba W (2017) One-shot imitation learning. In: *Advances in neural information processing systems*. pp. 1087–1098.

Eitel A, Hauff N and Burgard W (2017) Learning to singulate objects using a push proposal network. In: *Proc. of the International Symposium on Robotics Research (ISRR)*. Puerto Varas, Chile.

Finn C, Abbeel P and Levine S (2017a) Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400* .

Finn C, Tan XY, Duan Y, Darrell T, Levine S and Abbeel P (2016) Deep spatial autoencoders for visuomotor learning. In: *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, pp. 512–519.

Finn C, Yu T, Zhang T, Abbeel P and Levine S (2017b) One-shot visual imitation learning via meta-learning. *arXiv preprint arXiv:1709.04905* .

Florensa C, Held D, Wulfmeier M and Abbeel P (2017) Reverse curriculum generation for reinforcement learning. *arXiv preprint arXiv:1707.05300* .

Fortunato M, Azar MG, Piot B, Menick J, Osband I, Graves A, Mnih V, Munos R, Hassabis D, Pietquin O et al. (2018) Noisy networks for exploration. In: *International Conference on Learning Representations*. URL https://openreview.net/forum?id=rywHCPkAW.

Fu J, Levine S and Abbeel P (2016) One-shot learning of manipulation skills with online dynamics adaptation and neural network priors. In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, pp. 4019–4026.

Fu MC, Glover FW and April J (2005) Simulation optimization: a review, new developments, and applications. In: *Proceedings of the 37th conference on Winter simulation*. Winter Simulation Conference, pp. 83–95.

Gao Y, Xu HH, Lin J, Yu F, Levine S and Darrell T (2018) Reinforcement learning from imperfect demonstrations .

Giusti A, Guzzi J, Cireşan DC, He FL, Rodríguez JP, Fontana F, Faessler M, Forster C, Schmidhuber J, Di Caro G et al. (2016) A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters* 1(2): 661–667.

Goodfellow I, Bengio Y and Courville A (2016) *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A and Bengio Y (2014) Generative adversarial nets. In: *Advances in neural information processing systems*. pp. 2672–2680.

Gu S, Holly E, Lillicrap T and Levine S (2017) Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In: *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, pp. 3389–3396.

Gu S, Lillicrap T, Sutskever I and Levine S (2016) Continuous deep q-learning with model-based acceleration. In: *International Conference on Machine Learning (ICML-16)*.

Guo Y, Liu Y, Oerlemans A, Lao S, Wu S and Lew MS (2016) Deep learning for visual understanding: A review. *Neurocomputing* 187: 27–48.

Gupta A, Eppner C, Levine S and Abbeel P (2016) Learning dexterous manipulation for a soft robotic hand from human demonstrations. In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, pp. 3786–3793.

Gupta S, Davidson J, Levine S, Sukthankar R and Malik J (2017a) Cognitive mapping and planning for visual navigation. *arXiv preprint arXiv:1702.03920* .

Gupta S, Fouhey D, Levine S and Malik J (2017b) Unifying map and landmark based representations for visual navigation. *arXiv preprint arXiv:1712.08125* .

He K, Zhang X, Ren S and Sun J (2016) Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. pp. 770–778.

Heess N, Sriram S, Lemmon J, Merel J, Wayne G, Tassa Y, Erez T, Wang Z, Eslami A, Riedmiller M et al. (2017) Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286* .

Henderson P, Islam R, Bachman P, Pineau J, Precup D and Meger D (2017) Deep reinforcement learning that matters. *arXiv preprint arXiv:1709.06560* .

Ho J and Ermon S (2016) Generative adversarial imitation learning. In: *Advances in Neural Information Processing Systems*. pp. 4565–4573.

Hoffman J, Tzeng E, Park T, Zhu JY, Isola P, Saenko K, Efros AA and Darrell T (2017) Cycada: Cycle-consistent adversarial domain adaptation. *arXiv preprint arXiv:1711.03213* .

Huang G, Liu Z, Weinberger KQ and van der Maaten L (2017) Densely connected convolutional networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, volume 1. p. 3.

Jaderberg M, Mnih V, Czarnecki WM, Schaul T, Leibo JZ, Silver D and Kavukcuoglu K (2016) Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397* .

Kakade S and Langford J (2002) Approximately optimal approximate reinforcement learning. In: *ICML*, volume 2. pp. 267–274.

Kalweit G and Boedecker J (2017) Uncertainty-driven imagination for continuous deep reinforcement learning. In: Levine S, Vanhoucke V and Goldberg K (eds.) *Proceedings of the 1st Annual Conference on Robot Learning*, *Proceedings of Machine Learning Research*, volume 78. PMLR, pp. 195–206. URL http://proceedings.mlr.press/v78/kalweit17a.html.

Kanitscheider I and Fiete I (2017) Training recurrent networks to generate hypotheses about how the brain solves hard navigation problems. In: *Advances in Neural Information Processing Systems*. pp. 4532–4541.

Khan A, Zhang C, Atanasov N, Karydis K, Kumar V and Lee DD (2017) Memory augmented control networks. *arXiv preprint arXiv:1709.05706* .

Koenig N, A B and Howard A (2004) Design and use paradigms for gazebo, an open-source multi-robot simulator. In: *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3. IEEE, pp. 2149–2154.

Kolve E, Mottaghi R, Gordon D, Zhu Y, Gupta A and Farhadi A (2017) Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474* .

Kretzschmar H, Spies M, Sprunk C and Burgard W (2016) Socially compliant mobile robot navigation via inverse reinforcement learning. *The International Journal of Robotics Research* 35(11): 1289–1307.

Krizhevsky A, Sutskever I and Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*. pp. 1097–1105.

Kulkarni TD, Saeedi A, Gautam S and Gershman SJ (2016) Deep successor reinforcement learning. *arXiv preprint arXiv:1606.02396* .

Kumar V, Todorov E and Levine S (2016) Optimal control with learned local models: Application to dexterous manipulation. In: *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, pp. 378–383.

Kümmerle R, Grisetti G, Strasdat H, Konolige K and Burgard W (2011) g 2 o: A general framework for graph optimization. In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, pp. 3607–3613.

Levine S, Finn C, Darrell T and Abbeel P (2016) End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research* 17(1): 1334–1373.

Levine S and Koltun V (2013) Guided policy search. In: *ICML (3)*. pp. 1–9.

Li Y, Song J and Ermon S (2017) Inferring the latent structure of human decision-making from raw visual inputs. *arXiv preprint arXiv:1703.08840* .

Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D and Wierstra D (2015) Continuous control with deep reinforcement learning. In: *Proceedings of the International*

Conference on Learning Representations (ICLR).

Long J, Shelhamer E and Darrell T (2015) Fully convolutional networks for semantic segmentation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 3431–3440.

Long P, Fan T, Liao X, Liu W, Zhang H and Pan J (2017) Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. *arXiv preprint arXiv:1709.10082* .

Maddern W, Pascoe G, Linegar C and Newman P (2017) 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research (IJRR)* 36(1): 3–15. DOI:10.1177/0278364916679498. URL http://dx.doi.org/10.1177/0278364916679498.

Mirowski P, Pascanu R, Viola F, Soyer H, Ballard A, Banino A, Denil M, Goroshin R, Sifre L, Kavukcuoglu K et al. (2016) Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673* .

Mnih V, Badia AP, Mirza M, Graves A, Lillicrap TP, Harley T, Silver D and Kavukcuoglu K (2016) Asynchronous methods for deep reinforcement learning. *arXiv preprint arXiv:1602.01783* .

Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G et al. (2015) Human-level control through deep reinforcement learning. *Nature* 518(7540): 529–533.

Nair A, McGrew B, Andrychowicz M, Zaremba W and Abbeel P (2017) Overcoming exploration in reinforcement learning with demonstrations. *arXiv preprint arXiv:1709.10089* .

Nichol A and Schulman J (2018) Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999* .

Okal B and Arras KO (2016) Learning socially normative robot navigation behaviors with bayesian inverse reinforcement learning. In: *ICRA*. pp. 2889–2895.

Parisotto E, Chaplot DS, Zhang J and Salakhutdinov R (2018) Global pose estimation with an attention-based recurrent network. *arXiv preprint arXiv:1802.06857* .

Parisotto E and Salakhutdinov R (2017) Neural map: Structured memory for deep reinforcement learning. *arXiv preprint arXiv:1702.08360* .

Pathak D, Agrawal P, Efros AA and Darrell T (2017) Curiosity-driven exploration by self-supervised prediction. In: *International Conference on Machine Learning (ICML)*, volume 2017.

Peng XB, Andrychowicz M, Zaremba W and Abbeel P (2017) Sim-to-real transfer of robotic control with dynamics randomization. *arXiv preprint arXiv:1710.06537* .

Pfeiffer M, Schwesinger U, Sommer H, Galceran E and Siegwart R (2016) Predicting actions to act predictably: Cooperative partial motion planning with maximum entropy models. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. pp. 2096–2101.

Plappert M, Houthooft R, Dhariwal P, Sidor S, Chen RY, Chen X, Asfour T, Abbeel P and Andrychowicz M (2018) Parameter space noise for exploration. In: *International Conference on Learning Representations*. URL https://openreview.net/forum?id=ByBAl2eAZ.

Popov I, Heess N, Lillicrap T, Hafner R, Barth-Maron G, Vecerik M, Lampe T, Tassa Y, Erez T and Riedmiller M (2017) Data-efficient deep reinforcement learning for dexterous manipulation. *arXiv preprint arXiv:1704.03073* .

Radford A, Metz L and Chintala S (2015) Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434* .

Riedmiller M, Hafner R, Lampe T, Neunert M, Degrave J, Van de Wiele V Tom adn Mnih, Heess N and Springenberg JT (2018) Learning by playing - solving sparse reward tasks from scratch. *arXiv preprint arXiv:1802.10567* .

Rohmer E, Singh SP and Freese M (2013) V-rep: A versatile and scalable robot simulation framework. In: *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, pp. 1321–1326.

Ross S, Gordon G and Bagnell D (2011) A reduction of imitation learning and structured prediction to no-regret online learning. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. pp. 627–635.

Ruder M, Dosovitskiy A and Brox T (2017) Artistic style transfer for videos and spherical images. *arXiv preprint arXiv:1708.04538* .

Rusu AA, Večerík M, Rothörl T, Heess N, Pascanu R and Hadsell R (2017) Sim-to-real robot learning from pixels with progressive nets. In: *Conference on Robot Learning*. pp. 262–270.

Savinov N, Dosovitskiy A and Koltun V (2018) Semi-parametric topological memory for navigation. In: *International Conference on Learning Representations*. URL https://openreview.net/forum?id=SygwwGbRW.

Savva M, Chang AX, Dosovitskiy A, Funkhouser T and Koltun V (2017) Minos: Multimodal indoor simulator for navigation in complex environments. *arXiv preprint arXiv:1712.03931* .

Schaul T, Horgan D, Gregor K and Silver D (2015a) Universal value function approximators. In: *International Conference on Machine Learning*. pp. 1312–1320.

Schaul T, Quan J, Antonoglou I and Silver D (2015b) Prioritized experience replay. *arXiv preprint arXiv:1511.05952* .

Schmidhuber J (2015) Deep learning in neural networks: An overview. *Neural networks* 61: 85–117.

Schulman J, Levine S, Abbeel P, Jordan M and Moritz P (2015a) Trust region policy optimization. In: *International Conference on Machine Learning*. pp. 1889–1897.

Schulman J, Moritz P, Levine S, Jordan M and Abbeel P (2015b) High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438* .

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

Shah S, Dey D, Lovett C and Kapoor A (2017) Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In: *Field and Service Robotics*. URL https://arxiv.org/abs/1705.05065.

Silver D, Lever G, Heess N, Degris T, Wierstra D and Ried-

miller M (2014) Deterministic policy gradient algorithms. In: *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. pp. 387–395.

Stachenfeld KL, Botvinick MM and Gershman SJ (2017) The hippocampus as a predictive map. *Nature neuroscience* 20(11): 1643.

Stadie BC, Abbeel P and Sutskever I (2017) Third-person imitation learning. *arXiv preprint arXiv:1703.01703* .

Sukhbaatar S, Kostrikov I, Szlam A and Fergus R (2017) Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv preprint arXiv:1703.05407* .

Sutton RS (1991) Dyna, an integrated architecture for learning, planning, and reacting. *ACM SIGART Bulletin* 2(4): 160–163.

Sutton RS and Barto AG (1998) *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Szita I and Lörincz A (2006) Learning tetris using the noisy cross-entropy method. *Neural computation* 18(12): 2936–2941.

Tai L, Li S and Liu M (2016) A deep-network solution towards model-less obstacle avoidance. In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, pp. 2759–2764.

Tai L, Paolo G and Liu M (2017) Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. pp. 31–36.

Tai L, Zhang J, Liu M and Burgard W (2018) Socially-compliant navigation through raw depth inputs with generative adversarial imitation learning. In: *Robotics and Automation (ICRA), 2018 IEEE International Conference on*.

Tamar A, Wu Y, Thomas G, Levine S and Abbeel P (2016) Value iteration networks. In: *Advances in Neural Information Processing Systems*. pp. 2154–2162.

Thrun S, Burgard W and Fox D (2005) *Probabilistic robotics*.

Tobin J, Fong R, Ray A, Schneider J, Zaremba W and Abbeel P (2017) Domain randomization for transferring deep neural networks from simulation to the real world. In: *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, pp. 23–30.

Todorov E, Erez T and Tassa Y (2012) Mujoco: A physics engine for model-based control. In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, pp. 5026–5033.

Tzeng E, Devin C, Hoffman J, Finn C, Abbeel P, Levine S, Saenko K and Darrell T (2015) Towards adapting deep visuomotor representations from simulated to real environments. *arXiv preprint arXiv:1511.07111* .

Tzeng E, Hoffman J, Zhang N, Saenko K and Darrell T (2014) Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474* .

Van Hasselt H, Guez A and Silver D (2016) Deep reinforcement learning with double q-learning. In: *AAAI*, volume 16. pp. 2094–2100.

Večerík M, Hester T, Scholz J, Wang F, Pietquin O, Piot B, Heess N, Rothörl T, Lampe T and Riedmiller M (2017) Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817* .

Wang JX, Kurth-Nelson Z, Tirumala D, Soyer H, Leibo JZ, Munos R, Blundell C, Kumaran D and Botvinick M (2016a) Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763* .

Wang Z, de Freitas N and Lanctot M (2016b) Dueling network architectures for deep reinforcement learning. In: *Proceedings of The 33rd International Conference on Machine Learning*.

Wang Z, Merel JS, Reed SE, de Freitas N, Wayne G and Heess N (2017) Robust imitation of diverse behaviors. In: *Advances in Neural Information Processing Systems*. pp. 5326–5335.

Weber T, Racanière S, Reichert DP, Buesing L, Guez A, Rezende DJ, Badia AP, Vinyals O, Heess N, Li Y et al. (2017) Imagination-augmented agents for deep reinforcement learning. *arXiv preprint arXiv:1707.06203* .

Williams RJ (1992) Simple statistical gradient-following algorithms for connectionist reinforcement learning. In: *Reinforcement Learning*. Springer, pp. 5–32.

Wu Y, Mansimov E, Grosse RB, Liao S and Ba J (2017) Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In: *Advances in neural information processing systems*. pp. 5285–5294.

Wu Y, Wu Y, Gkioxari G and Tian Y (2018) Building generalizable agents with a realistic and rich 3d environment. *arXiv preprint arXiv:1801.02209* .

Wulfmeier M, Ondruska P and Posner I (2015) Maximum entropy deep inverse reinforcement learning. *arXiv preprint arXiv:1507.04888* .

Yang L, Liang X and Xing E (2018) Unsupervised real-to-virtual domain unification for end-to-end highway driving. *arXiv preprint arXiv:1801.03458* .

You Y, Pan X, Wang Z and Lu C (2017) Virtual to real reinforcement learning for autonomous driving. *arXiv preprint arXiv:1704.03952* .

Yu T, Finn C, Xie A, Dasari S, Zhang T, Abbeel P and Levine S (2018) One-shot imitation from observing humans via domain-adaptive meta-learning. *arXiv preprint arXiv:1802.01557* .

Zhang J, Springenberg JT, Boedecker J and Burgard W (2017a) Deep reinforcement learning with successor features for navigation across similar environments. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. pp. 2371–2378.

Zhang J, Tai L, Boedecker J, Burgard W and Liu M (2017b) Neural slam. *arXiv preprint arXiv:1706.09520* .

Zhang J, Tai L, Xiong Y, Liu M, Boedecker J and Burgard W (2018) Vr goggles for robots: Real-to-sim domain adaptation for visual control. *arXiv preprint arXiv:1802.00265* .

Zhang T, McCarthy Z, Jow O, Lee D, Goldberg K and Abbeel P (2017c) Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. *arXiv preprint arXiv:1710.04615* .

Zhu JY, Park T, Isola P and Efros AA (2017a) Unpaired image-

to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593* .

Zhu Y, Mottaghi R, Kolve E, Lim JJ, Gupta A, Fei-Fei L and Farhadi A (2017b) Target-driven visual navigation in indoor scenes using deep reinforcement learning. In: *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, pp. 3357–3364.

Zhu Y, Wang Z, Merel J, Rusu A, Erez T, Cabi S, Tunyasuvunakool S, Kramár J, Hadsell R, de Freitas N et al. (2018) Reinforcement and imitation learning for diverse visuomotor skills. *arXiv preprint arXiv:1802.09564* .

Ziebart BD, Maas AL, Bagnell JA and Dey AK (2008) Maximum entropy inverse reinforcement learning. In: *AAAI*, volume 8. Chicago, IL, USA, pp. 1433–1438.