Neural Networks 153 (2022) 13–36

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Review

A survey for deep reinforcement learning in markovian cyber–physical systems: Common problems and solutions☆

Timothy Rupprecht *, Yanzhi Wang

Northeastern University, 360 Huntington Ave, 02445 MA, Boston, United States

ARTICLE INFO

Article history:
Received 14 September 2021
Received in revised form 6 May 2022
Accepted 13 May 2022
Available online 21 May 2022

Keywords:
Deep reinforcement learning
Cyber–physical systems
Motor control
Resource allocation
HVAC

ABSTRACT

Deep Reinforcement Learning (DRL) is increasingly applied in cyber–physical systems for automation tasks. It is important to record the developing trends in DRL's applications to help researchers overcome common problems using common solutions. This survey investigates trends seen within two applied settings: motor control tasks, and resource allocation tasks. The common problems include intractability of the action space, or state space, as well as hurdles associated with the prohibitive cost of training systems from scratch in the real-world. Real-world training data is sparse and difficult to derive and training in real-world can damage real-world learning systems. Researchers have provided a set of common as well as unique solutions. Tackling the problem of intractability, researchers have succeeded in guiding network training with handcrafted reward functions, auxiliary learning, and by simplifying the state or action spaces before performing transfer learning to more complex systems. Many state-of-the-art algorithms reformulate problems to use multi-agent or hierarchical learning to reduce the intractability of the state or action spaces for a single agent. Common solutions to the prohibitive cost of training include using benchmarks and simulations. This requires a shared feature space common to both simulation and the real world; without that you introduce what is known as the reality gap problem. This is the first survey, to our knowledge, that studies DRL as it is applied in the real world at this scope. It is our hope that the common solutions surveyed become common practice.

Contents

☆ This work was supported by the National Science Foundation under grants CMMI-2013067 and CCD-1937500 received by Yanzhi Wang.
* Corresponding author.
E-mail addresses: rupprecht.t@northeastern.edu (T. Rupprecht), yanz.wang@northeastern.edu (Y. Wang).

https://doi.org/10.1016/j.neunet.2022.05.013
0893-6080/

1. Introduction

This survey sets out to explore how deep reinforcement learning is used within cyber–physical systems. We pay attention particularly to common problems and common solutions that researchers discover within this field. We will explore solutions to intractability of the state or action space, the prohibitive cost of training a learning system from scratch in the real world, and the reality gap problem that arises when training in a simulation and evaluating in the real world.

Deep reinforcement learning (DRL) is a type of deep learning (DL) meant to solve problems formulated as Markov Decision Processes (MDP). This is how reinforcement learning (RL) formulates all of the problems it is meant to solve. The seminal works in DRL owe much of their success to the advances made within the traditional reinforcement learning field. This symbiotic relationship goes both ways, too. Reinforcement learning lends concepts like the MDP problem formulation, concepts like value and policy iteration, as well as advances in Q-learning such as double Q-learning. But, going the reverse direction, DRL develops concepts like the experience replay buffer, and auxiliary learning that are then adopted into the field of traditional non-deep reinforcement learning.

We believe that there exists a similar symbiotic relationship between the development of theory, and the implementation of that theory in practice. For example, in DRL it is perhaps obvious that researchers borrow neural network architectures and training techniques for their own needs. But what is perhaps less observed is the understanding DRL researchers give back to the larger deep learning community: the handcrafted reward functions, the benchmarks, and the importance of multi-agent learning (MAL) and hierarchical learning (HL) problem formulation.

In the age of the internet-of-things there are many factors causing the adoption of automation. Companies like Boston Dynamics, Tesla and Uber, alongside the implementation of the smart-grid, and early adoption of self-driving vehicles are all contributing to an increased use of automation. DRL is fast becoming a favorite tool in the automation toolbox used by researchers to perform motor control tasks, and resource allocation tasks.

All of these technologies relying on DRL run into similar problems but while surveying the field a series of common solutions becomes apparent. Whether it is the intractability of the state or action space, or the prohibitive cost of training real-world autonomous vehicles, solutions abound in this field. It is our hope, that in reading this survey other researchers develop a sense of how to formulate problems encountered in their own research. We believe we are making two contributions to this field of research with this survey:

1. To our knowledge this is the first review covering DRL in both motor control, and resource allocation tasks within applied settings since 2019 (Liu, Xu, Liao, & Yu, 2019). It is the first review of its kind to cover DRL algorithms working with applications in HVAC control, automated surgical procedures, electric vehicle routing to charging stations, and simultaneous localization and mapping. In addition to these applications we discuss optimizations used to speed up algorithms for on device implementation.
2. We compare the common problems experienced in both motor control, and resource allocation tasks and curate the solutions common to both fields. This is the first review of its kind to curate examples using both architectural and training methods to overcome the reality gap between simulation and real-world evaluation.

2. Background

2.1. Technical advances in deep reinforcement learning

2.1.1. Origins of deep reinforcement learning

Deep reinforcement learning, perhaps obviously, is the marriage of the fields of deep learning and reinforcement learning. Deep learning and reinforcement learning provide springboards of success to DRL researchers at the architecture and system design levels of abstraction, respectively. Borrowed from deep learning are training techniques like batch normalization, and dropout, many familiar architectures such as progressive networks, guided policy search, convolutional layers, auto-encoders, and lastly recurrent layers such as long short-term memory (LSTM). Taking a look at the system design level of abstraction, reinforcement learning problems teach us to formulate problems as Markov Decision Processes (MDPs) which can be solved with value or policy iteration. These problems are governed by the Markov property which states that the future is conditionally independent of the past given the present state.

MDPs are five-tuple models of a system where determining the probability of state-traversal from state to state requires only the observation of the current state. The five model variables are $\{\mathbb{S}, \mathbb{A}, \mathbb{R}, \mathbb{T}, \gamma\}$. Each variable corresponds to the state space, the action space, the reward function, the state transition probabilities, and γ a hyper-parameter known as the discount factor used in the loss function during training. This problem formulation is shown in equation (4) as one of the many ways reinforcement learning influences DRL. Broadly speaking there are three types of MDP systems. Partially observable MDP (POMDP) systems are when the state space is not fully observed when a decision is made. There is also such a thing as an episodic MDP, where the state resets after some period of time. Finally, there also exist situations in which the action space is not a set of discrete actions but a continuous action space. These systems are called continuous MDPs. Similarly the state space may exist in continuous time, but in these circumstances it is still assumed that decisions are made in discrete intervals and the state is measured and observed at those intervals. When this assumption is not made

the system is said to be a semi-MDP. The foundational work on semi-MDPs (Sutton, Precup, & Singh, 1999) states that "Formally, a set of options defined over an MDP constitutes a semi-Markov decision process (SMDP), and the theory of SMDPs provides the foundation for the theory of options". Here options are sets of actions to be used as closed-loop policies for taking action over a period of time. The usage of options within a traditional MDP system allows learning, planning, and representing knowledge at multiple levels of temporal abstraction. For more valuable information please see the 1999 work (Sutton et al., 1999) by Sutton, et al. Taking the observation of the state space, and using a set of actions, the learning model seeks to maximize a reward function. Taking this problem formulation and then utilizing Bellmen's theory of optimality (Bellman, 1954) we arrive at the definitions seen in equations (1) through (5) that make up the foundation of reinforcement learning. These equations provide a grammar that all of RL and DRL build on. A policy that maximizes the expected reward given a policy is known as the optimal policy function:

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\ \mathbb{E}[R|\pi] \tag{1}$$

The expected reward given a policy and state is referred to as the value function:

$$V(s) = \mathbb{E}[R|s,\pi] \quad \forall s \in S \tag{2}$$

The expected reward given an optimal policy and state, and selected action is called the optimal quality function:

$$Q^*(s_t, a_t) = \underset{\pi}{\operatorname{argmax}}\ \mathbb{E}[R|s,a,\pi] \tag{3}$$

And finally $Q^*(s_t, a_t)$ rewritten as a Bellman equation:

$$Q^*(s_t, a_t) = \underset{\pi}{\operatorname{argmax}}\ \underset{s_{t+1}}{\mathbb{E}}\left[R_{t+1} + \gamma Q^*(s_{t+1}, a_{t+1})|s,a,\pi\right] \tag{4}$$

To learn more about traditional reinforcement learning, please see the 2009 survey from Gosavi (2009).

As this is a survey focusing on deep reinforcement learning in markovian environments, research focusing on non-markovian environments are outside the scope of what we will discuss. However the reader should still be aware of such environments as they still appear alongside cyber–physical systems like physiological systems, and environments that use human behavior modeling with long-range dependence in decision making. One such non-markovian process is the history dependent process (Majeed & Hutter, 2018). We also see non-Markovian rewards (Agarwal & Aggarwal, 2019; Gaon & Brafman, 2020). The recent development of fractional dynamics is another tool for modeling non-Markovianity using compact models (Gupta, Yin, Deshmukh, & Bogdan, 2021).

In DRL neural networks are trained as discriminate functions to approximate the quality function or policy function. When the network approximates a quality function this is called Deep Q-learning and is another example of how the traditional field of reinforcement learning provides springboards for success to DRL researchers. Q-learning is a form of value iteration where a learning model takes features observed from a state and produces an output that indicates the quality score of the corresponding actions from the possible action space. The action with the highest quality score is then selected.

As an alternative to Q-learning, models that use neural networks to approximate both a policy gradient as well as approximate the quality function are known as actor–critic models. The name actor–critic comes from conceptualizing the policy function as an actor, and the quality function as the critic. Models that seek to approximate only one of these functions are called actor or critic models respectively. Actor–critic models can be referred to as a hybrid of value and policy iteration methods because both functions are learned and approximated. In DRL this means having two networks, or one for each function. Oftentimes, actor–critic models seek to maximize the advantage function, which can be defined below:

$$A(s,a) = Q(s,a) - \overline{V}(s) \tag{5}$$

where $\overline{V}(s)$ is an average value for the state given any possible action. This of course is different from value iteration or Q-learning which seeks to only approximate and maximize the value or quality function.

2.1.2. Deep reinforcement learning networks

It is generally accepted that the advent of DRL was the 2013 publication and creation of the first iteration of the Deep Q Network, or DQN (Mnih et al., 2013). The DQN was a very simple neural network that achieved deeply exciting results. DQN uses an RGB pixel state space representation of an Atari video game. The researchers are said to maintain the Markov property in their problem formulation by defining the current state space as the preceding 30 frames of gameplay. When motion within a system must be extracted or analyzed then using a single frame RGB pixel state space representations often does not completely capture the motion or dynamics of a system. The 30 frame window allows the interpretation of those dynamics. The original DQN used two convolutional layers, followed by two fully connected linear layers to regress the quality function scores for each possible action the Atari game allows as an input to the Atari 2600 game emulator. Taking the action with the maximum quality score maximizes performance in the simple Atari games played by the DQN. The evolution of the DQN over the next several years from its creation and onward form the basis of a large corner stone of DRL theory. Over the years, DQN has been updated to use techniques seen in RL and DL such as, double Q-learning (Hasselt; van Hasselt, Guez, & Silver, 2015), dueling Q-learning (Wang et al., 2016), weighted Q-learning (Cini, D'Eramo, Peters, & Alippi, 2020), and with the swapping of the final fully connected linear layer with a Long Short-term Memory (LSTM) layer, a recurrent DQN for POMDPs (Hausknecht & Stone, 2017). Overestimation of the maximum action-value is an infamous problem in Q-learning that double Q-learning, and weighted Q-learning seek to overcome. Meanwhile, dueling Q-learning seeks to approximate the advantage function seen in equation (5) to better inform decisions made by the model.

This five year stretch culminated in the creation of Rainbow, a DQN that combines several of these improvements into one model (Hessel et al., 2017a). Rainbow includes, as previously mentioned, double Q-learning (van Hasselt et al., 2015), prioritized replay buffers (Schaul, Quan, Antonoglou, & Silver, 2016), dueling networks (Wang et al., 2016), multi-step learning (Asis, Hernandez-Garcia, Holland, & Sutton, 2018), distributed Q-learning (Bellemare, Dabney, & Munos, 2017), and the introduction of Gaussian noise into the fully connected layers to promote exploration of the action space (Fortunato et al., 2019). As stated before, double Q-learning helps overcome the overestimation bias seen in Q-learning, and dueling networks incorporates the additional information of the approximated advantage function seen in equation (5) to better inform decisions made by the model. Distributed Q-learning speeds training up by asynchronously updating the gradient updates from models training in parallel. Multi-step learning helps to propagate newly observed rewards faster to earlier visited states and also overcomes the bias–variance problem seen in reinforcement learning. And finally using both a prioritized replay buffer and the introduction of Gaussian noise into the final fully connected layer promote exploration of the action space. Fig. 1 reproduced with permission from Hessel et al. (2017a) shows how these different improvements synthesize into one model improving immensely over the

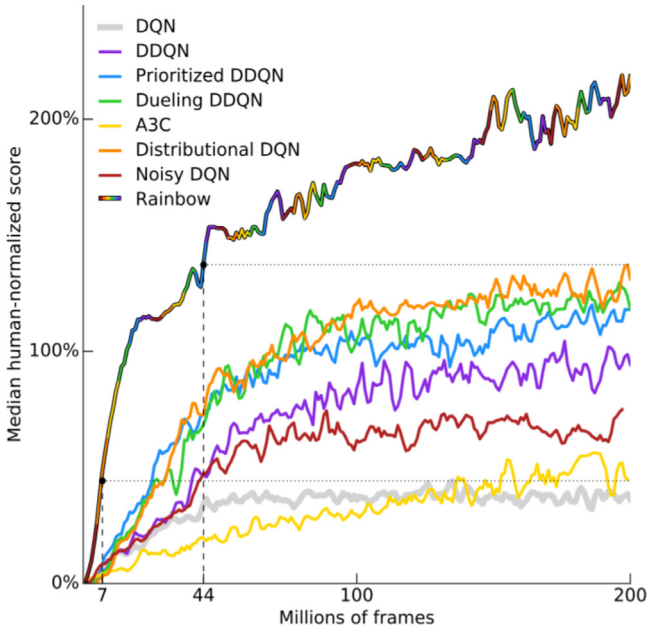


Fig. 1. This graph demonstrates how all of the proposed improvements to DQN synthesize into one network — the Rainbow DQN (Hessel et al., 2017a) (All rights reserved Hessel et al., 2017b). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

baseline model. Rainbow DQN not only outperforms the baseline implementations of both A3C, and DQN, but also out performs any one contribution to DQN's implementation. We direct readers to Obando-Ceron and Castro (2021) from 2021 for the newest examination of Rainbow developments. The authors of Obando-Ceron and Castro (2021) also explore how smaller training environments can yield significant insights as benchmarks for DQN experiments.

As impressive as early results with DQN were, DQN suffers from two major setbacks. First, the network cannot work with continuous action spaces (Mnih et al., 2013) which are ubiquitous in motor control tasks. And second, it cannot work with action spaces with a high dimensionality (Lillicrap et al., 2019). As the authors of Lillicrap et al. (2019) point out in their creation of the actor–critic network Deep Deterministic Policy Gradients (DDPG), even if one created discrete values for action states taken from a continuous action space, say a human arm with seven degrees of freedom, and limited the motor actions at each joint to $[-1, 0, 1]$, this still leaves DQN facing an action space with a dimensionality of $3^7 = 2187$ and the network will likely fail to converge during training. Neural networks can be used in an actor model to select a policy from across a continuous action space given a state space representation. The critic model is similar to a DQN but includes fusion with the actor network outputs to create a single quality score for the selected actions. As an example, using a motor control policy with continuous action space, the actor network of the actor–critic model regresses a value between $[-1, 1]$ for each motor under the model's supervision. These actions are handed off to the critic network which regresses a single quality function score reflecting the actions handed to it. The quality scores regressed by the critic network are what are used to indirectly tune the optimal policy gradients during back propagation by minimizing the loss function see in equation (6). The actor–critic network Deep Deterministic Policy Gradients (DDPG) (Lillicrap et al., 2019), the Asynchronous Advantage Actor Critic Network (A3C) (Mnih et al., 2016), and IMPALA (Espeholt et al., 2018) are all actor–critic networks that take advantage of the advantage function, and actor–critic framework.

2.1.3. Network training

Both DQNs, and actor–critic models like DDPG use some form of a target network for the purpose of calculating their loss functions instead of relying on ground truth. Researchers use a parallel network called a target network that is updated with the model weights of the learning network at constant intervals. DDPG (Lillicrap et al., 2019) and DQN (Mnih et al., 2013) try to minimize the following loss function:

$$L(\theta^Q) = E_{s_t, a_t, r_t}[(Q(s_t, a_t|\theta^Q) - y_t)^2] \tag{6}$$

where,

$$y_t = r(s_t, a_t) + \gamma Q(s_{t+1}, \mu(s_{t+1})|\theta^Q) \tag{7}$$

It is this y_t value that uses the quality function approximation of the target network, and not the learning network. The target network is thrown away after training. A lot of the previously mentioned research about DQN revolves around trying to improve the network training convergence. Since the quality function uses a maximization operator to determine the best action to take, the network is inherently biased towards overestimation. Double DQN (van Hasselt et al., 2015) seeks to alleviate this problem by replacing the target network of DQN with a network that learns alongside the primary network rather than sharing model weights with the learning network; this network is also discarded after training like a target network. The double DQN not only removes this overestimation bias, but also converges to much better performance in a variety of experiments.

Researchers also focus on improving reward functions by utilizing techniques such as auxiliary learning introduced in Jaderberg et al. (2016). When an extra term is added to the loss function that reflects the nature of the system this is referred to in traditional DL as multi-task learning. But auxiliary learning can also be used to create a better reward function that helps the network infer system properties not explicitly shown to the network. The original DQN rewards the agent when it makes progress towards an objective, like increasing the score of a video game (Mnih et al., 2013). In applied settings, to properly evaluate performance, reward functions are handcrafted with industry expertise as we see throughout Section 3 and IV. Additionally, network architecture can be altered to promote auxiliary learning. In one HVAC example we survey (Xu, Wang, Wang, O'Neill, & Zhu, 2020) the researchers use an architecture similar to an actor–critic framework, except between their two networks the researchers create a latent space that corresponds to desired temperature changes within a thermal zone of a HVAC system as opposed to regressing raw action values corresponding to the HVAC controller inputs. A second network uses these desired temperature changes as an input to create those action values. This alteration of the actor–critic framework provides lessons about the thermal dynamics of the thermal zones that regressing raw action values does not provide directly.

2.1.4. Hierarchical, multi-agent, and imitation learning

It is worth pointing out to the reader the importance of RL topics like hierarchical learning (HL), multi-agent learning (MAL), and imitation learning. These fields also have their DRL counterparts like we have seen before. Hierarchical learning is when problems are structured such that decision making is decentralized into a master policy and subpolicy organization. Multi-agent learning involves multiple agents choosing actions that each affect the state space. These agents can work cooperatively or competitively like in a game. Imitation learning involves showing agents exactly what to do given a state through expert demonstration. In the cyber–physical realm, it is rare one finds themselves working within a closed system in and around humans. Therefore we express caution in utilizing imitation learning in these environments discussed throughout this

survey. We will explore the methods of both hierarchical learning, and multi-agent learning that occur in the applied settings we choose to survey, but if the reader is interested they should explore the 2018 multi-agent DRL survey (Hernandez-Leal, Kartal, & Taylor, 2019) or the 2018 PhD dissertation from Sanjay Krishnan that explores hierarchical learning in robotics, and data science (Krishnan, 2018). Similarly, if the reader has outstanding questions about deep reinforcement learning they should consult with this (Arulkumaran, Deisenroth, Brundage, & Bharath, 2017) deep reinforcement learning survey.

2.1.5. Benchmarks

As a final note, it is important to highlight the importance benchmarks play in the progression of research across all deep learning fields — including deep reinforcement learning. As many remember, computer vision based deep learning exploded in use and popularity after the publishing of the ImageNet image classification dataset in 2010 (Russakovsky et al., 2015). Networks were being developed and needed a way of being compared together. Likewise, the first significant work of DRL by necessity included the use of a virtual benchmark where networks can now be compared in performance on a series of Atari video game experiments (Mnih et al., 2013). Similarly, in application tasks, breakthroughs are stimulated by the production of domain specific simulators and datasets that allow network benchmarking. These benchmarks, simulators, and datasets can serve traditional deep learning networks as well as DRL researchers. We will speak more about how benchmarks provide an important simulated playground where network training and performance evaluation can occur.

2.2. Cyber–physical systems

2.2.1. Common problems

It is important to remember that the usage of neural networks in cyber–physical systems predates the usage of deep reinforcement learning methods. In introducing different applications of deep reinforcement learning, some particular attention will be paid to the seminal works that involve vanilla neural networks. Generally speaking, these methods predate the 2013 rise of deep reinforcement learning as discussed in the previous section.

Generally speaking the research surveyed meets three criteria. They explicitly use deep reinforcement learning; they are cited by many works that come after them; and the algorithms whether at time of publishing or in the future are meant to be applied in the real world as a cyber–physical system. We decided to focus on two major applications of deep reinforcement learning: motor control tasks, and resource allocation tasks. These are umbrella terms to describe a plethora of problems, and they are umbrella terms because they share characteristics in motivation, problem formulation, problems encountered, and solutions used to overcome those problems. Beneath the two umbrella terms we survey, we will see DRL as applied to autonomous robots in general, and self-driving cars in particular, navigation tasks such as path planning and visual odometry, traffic control, vehicle routing, telecommunication resource allocation, HVAC control of buildings, and management of the power grid. The common denominator to all of these tasks is that their decisions interact with and affect the real-world.

Across all of these applications there exists a set of problems ubiquitous to them all. Researchers when working in the real world witness intractability of to the state space representations or action spaces. Seminal works in applied DRL often require simplifying either the state or action space. It is also a common problem in the development of cyber–physical systems that training is prohibitively costly. Real-world training data is sparse and difficult to derive and training in real-world can damage real-world learning systems. What happens if you only have one multi-million dollar robot that needs to learn how to walk on mars? Do you build a multi-million dollar facility with lower gravity to train the robot in a martian environment? And how many times can the robot be replaced after damage before the project is bankrupted? As it turns out, researchers have published a large amount of solutions about both of these matters and it is time the research community have access to these solutions all in one place.

2.2.2. Trusting cyber–physical systems

As AI algorithms grow in prominence and are deployed within cyber–physical systems there is a growing demand to quantify how much trust should be placed in these AI algorithms. 2020 we see the first attempt to quantify network trustworthiness from Cheng, Nazarian, and Bogdan (2020) where the authors propose DeepTrust. DeepTrust looks at pretrained neural network architectures to make a determination of how much trust to place in the network's predictions. DeepTrust can help warn a neural network user when a prediction is overconfident or under confident due to architecture constraints. Future work suggests a desire to explore methodologies for determining when the data used to train the network may be unreliable itself irrespective of architecture. If such a methodology could be used in tandem with DeepTrust researchers will have a robust way of determining just how much trust one should place in AI systems applied in the real world.

In an additional work (Cheng et al., 2021) from 2021 by the same researchers we see an effort to quantify the trustworthiness of individual agents within a multi-agent system. These learning agents were operating within traffic light control, and cooperative adaptive cruise control systems. These learning models successfully incorporate the measurement of other agents' trustworthiness to mitigate the effect of untrustworthy agents. In the cruise control environment the methodology can detect bad actors, and in the traffic control scenarios collisions were reduced compared to vanilla versions of the same algorithms.

2.2.3. Optimizing for hardware

As the demand grows for deep learning and deep reinforcement learning to be implemented within cyber–physical systems, so too does the demand grow for suitable methods of software hardware co-design. In general, neural networks are computationally burdensome and steps need to be taken to modify the neural networks before they are suitable for targeted hardware devices. There has been great success in the deep learning field applying the Alternative Direction Method of Multipliers algorithm to perform weight quantization and structured weight pruning to reduce the computational overhead of deep learning algorithms (Ren et al., 2018; Wang et al., 2019). Furthermore, lightweight frameworks for software compilation co-design have been implemented to further reduce computational overhead (Ma et al., 2019, 2020).

In some cases, deep reinforcement learning has been used in a design flow to increase network latency for the manycore systems of the future Xiao, Nazarian, and Bogdan (2021). In this work the researchers create a reinforcement learning graph convolutional network to select resources on hardware to assign to tasks. In a similar work from the same authors we see the usage of distributed Q-learning to schedule tasks to run on a GPU or a CPU at runtime (Xiao, Nazarian, & Bogdan, 2019). In both cases the authors see improvements in run-time compared to state of the art compilers and frameworks.

Table 1
Summary of papers surveyed for motor control tasks. Note that papers that use simulators in their training regimen may be set up for success in the real world, but unless evaluated in the real-world we hesitate to say the research has overcame the reality gap problem.

Paper	Network	Application	Virtual environment	HL	MAL	Problem overcame	Notes
Rusu et al. (2016a)	A3C	Robotic control	MuJoCo	N	N	Lack of training data + Reality gap	Uses Progressive Networks
Popov et al. (2017)	DDPG	Robotic control	MuJoCo	N	N	Lack of training data + Reality gap + Intractability of state/action space	Uses composite reward functions
Peng, Andrychowicz, Zaremba, and Abbeel (2018)	DDPG	Robotic control	MuJoCo	N	N	Lack of training data + Reality gap + Intractability of state/action space	Uses recurrent DDPG and randomized dynamics + Binary rewards and HER
Gu, Holly, Lillicrap, and Levine (2016)	DDPG + NAF	Robotic control	MuJoCo	N	N	Lack of training data	Asynchronous Training and random targets during training
Kalashnikov et al. (2018)	Qt-Opt	Robotic control	Bullet physics	N	N	Lack of training data	Uses Bellman Updater
Zhao, She, Zhu, Yang, and Xu (2021)	Unique Actor-Critic	Robotic control	Real-world	N	N	Intractability of state/action space	Effective auxiliary learning
Honerkamp, Welschehold, and Valada (2021)	TD3 + SAC	Robotic control	Custom	N	N	Intractability of state/action space	Uses feasible kinematics as sole reward signal
Liu and Jiang (2018)	Temporal CNN	Surgical control + Action segmentation	JIGSAWS	Y	N	Intractability of state/action space	Sets up future research to use HL
Omisore et al. (2018)	DQN	Surgical control	MatLab robotics toolbox	N	N	Lack of training data + reality gap + Intractability of state/action space	Develops deeply-learnt damped least-squares method
Tan, Chng, Su, Lim, and Chui (2019)	DDPG + GAIL	Surgical control	V-REP	N	N	Lack of training data + Reality gap	Uses Mask-RCNN on input video
Qureshi, Nakamura, Yoshikawa, and Ishiguro (2018)	Qnet + Pnet	Human robot interaction	Custom + Real-world	N	N	Intractability of state/action space	Uses intrinsic motivation
Sallab, Abdou, Perot, and Yogamani (2017)	DQN + DDAC	Autonomous driving	TORCS	N	N	Intractability of state/action space	Uses Recurrent DQN + DDAC
Pan, You, Wang, and Lu (2017)	A3C	Autonomous driving	TORCS	N	N	Lack of training data + Reality gap	Uses Mask-RCNN on input video
Duan, Eben Li, Guan, Sun and Cheng (2020)	APRL	Autonomous driving	Custom	Y	N	Intractability of state/action space	Compares/contrasts HL and non HL models

(*continued on next page*)

3. Motor control tasks

We have decided to hone in on three subsets of motor control tasks — autonomous robots in general, self-driving cars in particular, and navigation tasks. The autonomous robots in general include work on motor control for robotic joints, or contact manipulation problems, and even surgical applications. Navigation tasks include both path planning, as well as odometry/SLAM problems. While navigation tasks are not required to directly control the motors of their system, navigation is paramount in autonomous robots and vehicles, and will be surveyed alongside these other motor control tasks. These subsets of problems should see both similar action spaces, as well as similar state space representations. Table 1 summarizes the key take aways from the papers surveyed within this section.

Like a lot of the application settings we will discuss, autonomous control policy research predates the usage of DRL (Braganza, Dawson, Walker, & Nath, 2007; Hafner & Riedmiller, 2011; Omidvar & Elliott, 1997; Patre, MacKunis, Kaiser, & Dixon, 2008; Riedmiller, 2005). Perhaps the most famous example, Martin Riedmiller in 2005 developed a neural fitted Q Network (or NFQN) (Riedmiller, 2005). Six years later Martin Riedmiller realized the potential for applying this approximation of a quality function to the field of reinforcement learning by applying the NFQN to a motor control problem (Hafner & Riedmiller, 2011). However, surprisingly enough, one can go back all the way to 1997 to see researchers honing in on the importance of neural networks in control problems. Neural Systems for Control is an entire textbook dedicated to the early practices of applying neural systems to control problems (Omidvar & Elliott, 1997).

Sergey Levine and collaborators discovered Guided Policy Search in 2013 and they have applied it more recently to their robotics work (Levine & Koltun, 2013; Levine, Wagener, & Abbeel, 2015; Yahya, Li, Kalakrishnan, Chebotar, & Levine, 2017). Guided Policy Search (GPS) does not belong to a deep reinforcement learning algorithm and a deep explanation is outside the scope

Table 1 (*continued*).

Paper	Network	Application	Virtual environment	HL	MAL	Problem overcame	Notes
Das and Won (2021)	DQN	Autonomous driving	SUMO	N	N	Lack of training data + Intractability of state/action space	Effective auxiliary learning
Li, Sun, Chen, Tomizuka, and Zhan (2021)	Double DQN	Autonomous driving	Custom	Y	N	Lack of training data + Intractability of state/action space	Uses HL to perform different driving tasks with different motor controllers
Zhu, Gupta, Gupta, and Canova (2021)	PPO + LSTM	Autonomous driving	SUMO	N	N	Lack of training data + Intractability of state/action space	Uses LSTM to overcome POMDP nature of problem
Pfeiffer, Schaeuble, Nieto, Siegwart, and Cadena (2017)	CNN	Path planning	Custom	N	N	Lack of training data + Reality gap	Uses random start/goal states
Tai, Paolo, and Liu (2017)	DDPG	Path planning	V-REP	N	N	Lack of training data + Reality gap	Confirms findings in Pfeiffer et al. (2017)
Faust et al. (2017)	PRM-RL + DDPG	Path planning	Custom + MARHES	Y	N	Intractability of state/action space	Training generalizes to much larger real-world settings
Lv, Zhang, Ding, and Wang (2019)	DQN	Path planning	Custom	N	N	Intractability of state/action space	Extends DenseNet to DQN
Lei, Zhang, and Dong (2018)	DQN	Path planning	Gazebo	N	N	Lack of training data + Reality gap	Creates map from LiDAR inputs
Placed and Castellanos (2020)	DQN	Path planning	Custom	N	N	Intractability of state/action space	Uses handcrafted reward Function to promote Exploration for SLAM
Zhang, Zheng, Jia, and Li (2021)	PACNet	Path planning	Custom	Y	N	Intractability of state/action space	HL Visual Tracking algorithm for Path planning
Yan, Xiang, Wang, and Lan (2021)	PS-CACER	Path planning	Custom	N	N	Intractability of state/action space	Distributed algorithm for flocks of UAVs
Zhang, Tai, Liu, Boedecker, and Burgard (2020)	A3C	SLAM	Gazebo	N	N	Intractability of state/action space	First DRL method for SLAM and uses external memory

of this discussion, but it does incorporate neural networks and the reader should know about GPS as an alternative to DRL approaches so is explained superficially later on. A 2016 work from Levine and others (Levine, Pastor, Krizhevsky, & Quillen, 2016) saw them and their team work on training a real world robot to perform contact manipulation tasks using visual cues alone in two end-to-end trainable convolutional networks; one to predict grasping success and one to regress motor commands. Accomplishing this task required staging 800,000 grasping attempts using anywhere from six to fourteen robots active at a time, requiring in upwards of two months time to create this dataset to use in training. No doubt this represents a significant hurdle in time and resources for smaller teams of researchers. Further, by the authors' own account, while the algorithms proved invariant to changes in camera calibration and small variation in mechanical system hardware, the training does not provide an opportunity to generalize to new tasks not seen in the dataset nor to wholly different robots Levine et al. (2016). These constraints perhaps compound resource problems seen with creating this dataset as for every new task and robot a new dataset will need to be created. After this was published, the authors and associated team members when not using GPS, gravitate towards using DRL (Gu et al., 2016; Kalashnikov et al., 2018; Rajeswaran et al., 2018) as opposed to traditional deep learning approaches which require more ground truth (Nair et al., 2017).

The current state of the art for DRL algorithms use in motor control tasks all use simulations to train before evaluation in the real world. Therefore, close attention will be paid throughout this section on research meant to overcome the reality gap problem which can be introduced when simulations do not perfectly emulate real-world problems.

3.1. Motor control policies in general

Ruso, et al. in 2016 (Rusu et al., 2016, 2016a) proposed "using progressive networks to bridge the reality gap and transfer learned policies from simulation to the real world". Progressive networks are "immune to forgetting" through a process of sharing features between columns of layers trained to accomplish different tasks. When a column is finished training for a task, the weights are frozen, and a new column is added with connections to the first column and training on the new task begins. They ultimately found their "DRL algorithms are too slow to achieve performance on a real robot". Ruso et al. use A3C in this progressive network framework which may explain the high computational resources required to train and run the algorithms presenting a possible drawback for applications where memory and computational resources are not cheap. Fig. 2 reproduced with permission from Rusu et al. (2016a), shows this progressive network architecture.

Researchers at DeepMind, Popov, et al. in 2017 (Popov et al., 2017) showed that an asynchronous DDPG could be trained in simulation to stack objects with a robotic arm with 9 degrees of freedom. These researchers proposed using mini-batch training for their DDPG, created a composite reward function, and used apprenticeship start states. The future works suggest a goal

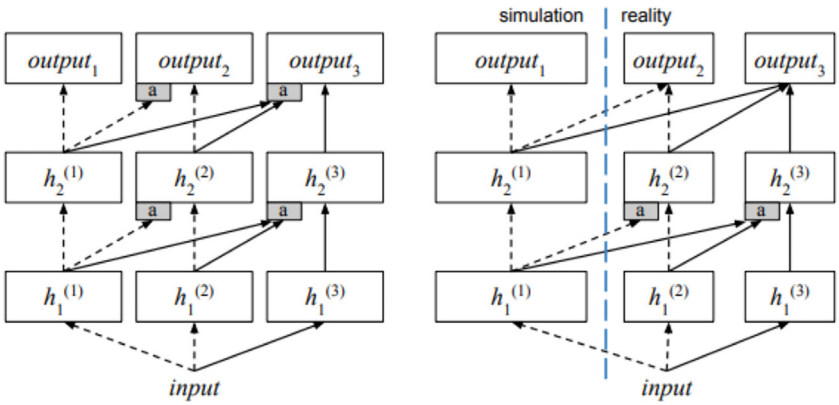


Fig. 2. Progressive networks in general work by having independently trained columns for separate tasks, that are then stacked in parallel and share features. For DRL motor control tasks there is one column for the simulated task, and one for the real world task (Rusu et al., 2016a, 2016b).

of developing an algorithm that is end-to-end trainable based on a visual input mirroring the work others have done within the field. A 2018 team of researchers from UC Berkeley and OpenAI (Peng et al., 2018) used a recurrent DDPG to train in simulation and evaluate performance in the real world. They achieved good generalization to the real-world and proved robust to calibration errors by leveraging randomization in the simulation's system dynamics, including but not limited to: table height, joint dampening, the mass of each arm joint, controller gains, and observation noise. Simulations were performed using the MuJoCo physics engine. A large contribution from this paper involves the use of a binary reward function and the Hindsight Experience Replay (or HER) described in Andrychowicz et al. (2017). As the authors of Peng et al. (2018) state, "by replaying past experiences with HER, the agent can be trained with more successful examples than is available in the original recorded trajectories" recorded in the replay buffer.

In 2015 Levine et al. (2015), showed that using Guided Policy Search to perform simple tasks with a robotic arm such as pushing a puck to a target position, screwing a bottle cap into a bottle, or stacking objects. Guided Policy Search is useful for creating a distribution of possible actions to draw from given a state that allows for better exploration in reinforcement learning problems. The guiding algorithm, such as iterative LQR, is used to model a distribution across policies and allows better training in the guided neural network motor controller as it chooses actions to learn the dynamics of a system through exploration and exploitation. Levine and their team have continued to apply GPS to reinforcement learning problems (Levine & Koltun, 2013; Yahya et al., 2017). In Gu, et al. from 2016 (Gu et al., 2016) working with Sergey Levine to build off previous work, these researchers proposed an asynchronous Normalized Advantage Function (NAF) that could achieve real-time performance in real-world robots performing door opening tasks. The team used NAF and DDPG trained with random task targets to make training more efficient. The algorithms were validated through training in the MuJoCo physics engine but due to the amount of real-world robots available to this team they had enough workers to train directly on real-world robots without simulated pretraining. By training directly on real-world robots without simulated pretraining, it cannot be said these researchers overcame the reality gap. DDPG could keep up with NAF on certain tasks, but where DDPG failed, NAF greatly outperformed DDPG. One of the most advanced approaches to use DRL to perform contact manipulation yet again comes a team associated with Sergey Levine at UC Berkeley. Kalashnikov, et al. in 2018 (Kalashnikov et al., 2018) proposed QT-Opt which uses episodic learning on a novel neural network architecture similar to DQN. The team trains their network in a distributed fashion, with parallel workers. They use a Bellman updater to draw training samples from their experience replay buffer for efficient training. Like in previous works (like Gu et al., 2016) the authors validated their work in simulation but were able to train directly on real-world robots and did not require simulation pretraining. Additionally, Qt-Opt continuously updates its grasping strategy up to the final moment of grasping. The framework proves moderately robust to occlusion and generalizes well. Qt-Opt is able to grasp objects that are never seen before at evaluation at a success rate of 96%.

In 2017 Hayes and Shah (2017) working out of MIT proposed a unique DRL paper that seeks to "intuitively" explain control policies learned by a DRL learning system. The future works section is exciting in that they hope that researchers can someday perform the reverse operation, and be able to explain a policy to a DRL agent that regresses the described policies. This would save DRL researchers in the future from the requirement to develop training procedures to reach the described policy through training and this work should be closely followed by the community.

A paper from 2021 (Zhao et al., 2021) attempts to solve a more difficult version of the bin packing problem with increased uncertainty in the items to be placed into bins. The algorithm uses an actor–critic framework with an RGB input sensor. The researchers add a multi-layer perceptron to create a feasibility mask of possible actions based on the visual input after it is sent through a feature extractor. This mask is then combined with the output the actor network. These architectural decisions act as a form of auxiliary learning which facilitates the agent to learn policies very efficiently. This approach when compared to human performance achieves almost equal performance.

In another paper from 2021 (Honerkamp et al., 2021), the researchers have a robotic arm that needs to perform a task. They use an actor–critic like with an end-effector generation network and a base agent network (based on soft-actor–critic (SAC) and a Twin Delayed Deep Deterministic (TD3)). The end-effector generation network proposes velocities for the end-effector. These proposals are fed into a base agent that proposes corresponding kinematics of the trajectory based on the velocities. The kinematics are then sent to an inverse-kinematics evaluator to generate the reward signal based on whether or not the proposed kinematics are feasible for the robot. When the robot learns in this way, kinematic feasibility of the trajectories are the sole learning signal. This prevents infeasible kinematics from being proposed by the system and makes training more efficient. The researchers showed in a variety of simulated and real-world experiments that their approach generalizes well to other environments and robots.

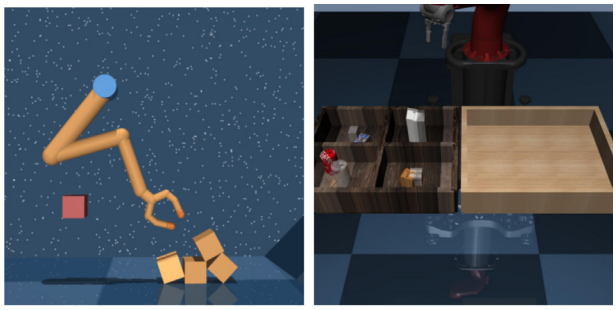
DRL based cyber–physical systems have the potential to revolutionize the way surgery is performed and work has already progressed in this application (Liu & Jiang, 2018; Omisore et al., 2018; Tan et al., 2019; Yu, Yu, Chen, & Zhang, 2019). One of the more recent works in 2019 (Tan et al., 2019), authored by a team from the University of Singapore, showed that DRL can be used to learn the motor control policies used in the mechanical surgeries. The authors compared and contrasted DDPG to GAIL by first training in simulation and performing finetuning on real-world robots. What is important to note here is the usage of Mask-RCNN to perform semantic segmentation on the visual video feed of the area undergoing surgery. This is one way of bridging the reality gap between medical simulations for training and evaluation in the real world when using RGB cameras as an input. This method of bridging the reality gap comes from researchers involved in autonomous driving cars, and will be discussed more later.

An earlier 2018 paper (Liu & Jiang, 2018) from Peking University is very important in reformulating the surgery setting into a hierarchical learning system. This research HL specifically, but researchers interested in DRL and surgery applications need to be aware of how this work can help formulate DRL surgery applications as HL problems. The researchers train on the JIGSAWS surgical dataset and perform action segmentation across these different surgeries and learn different steps of typical surgeries (e.g. suturing/stitching wounds, applying gauze, and cleaning the wound, etc.). After this initial segmentation task, different motor control policies can be learned by different agents to perform the specific action required during a surgery. These researchers did not take this next step to train a network to perform these tasks as subpolicies, leaving it to future work and other researchers.

Another paper from 2018 (Omisore et al., 2018) develops the deeply-learnt damped least-squares method to learn the kinematics of snake-like robot using DQN within the MatLab Toolbox. Here DQN is used to predict a dampening factor for the 8-joint arm of a snake robot meant to interact with a human subject. This paper is a good example of how the reality gap is smaller in robots that only use joint positions as an input to the network as opposed to rely on a RGB camera input. The network is trained in simulation and implemented directly on a real-world robot because the inputs are so consistent across both virtual and real settings.

We would like to draw the reader's attention to an interesting paper from 2018 on human robot interaction. As humans and robots interact more and more, researchers will seek to maximize "the human-like behavior" of their robots. This paper (Qureshi et al., 2018) uses two networks an action-conditional prediction network (Pnet) and a policy network (Qnet) or essentially a novel actor–critic algorithm. This algorithm uses intrinsic motivation while interacting with humans to learn a "more human" greeting pattern relying on the set of actions: wait, look towards human, wave hand, and handshake. The models trained with their proposed intrinsic motivation techniques learned qualitatively "more human" interaction skills than the baseline robots.

We would like to conclude our remarks on autonomous robots by briefly highlighting four of the emerging popular benchmarks for control tasks. Not all of the benchmarks used in the surveyed papers will be discussed here and not all of these benchmarks have been used in the surveyed papers. Four of these emerging benchmarks and simulation tools are the 2018 DeepMind Control Suite (Tassa et al., 2018), the 2012 MuJoCo Physics Simulator (Todorov, Erez, & Tassa, 2012), the 2018 Surreal open source RL and Robotics Manipulation benchmark (Fan et al., 2018), and the opensource multi-robot simulator known as Gazebo (Koenig & Howard, 2004) which was published the earliest in 2004. Fig. 3 shows snapshots of these simulators in action using snap shots of each reproduced with the permission from Fan et al. (2018) and Tassa et al. (2018). Traditionally, MuJoCo and Gazebo are most used in applications concerning navigation and walking. Surreal and the DeepMind Control Suite are both more oriented towards contact manipulation and tasks requiring the modeling physics. There is a lot of overlap in capabilities and researchers should investigate each of these more closely to determine what emulator is the best for their needs.



(a) DM Control Suite (b) SURREAL

Fig. 3. Two popular physics engines and simulators for control tasks (Fan et al., 2018; Tassa et al., 2018).

3.2. Self-driving cars in particular

Autonomous driving systems have a lot of overlap with classic motor control systems. For example, steering can be interpreted as a continuous action space, and the same goes for braking and accelerating. Additionally, the autonomous vehicles are controlled in the real-world through motors that perform the steering and the pressing of pedals. Therefore it is fair to say Autonomous driving is a subclass of motor control problems with slight variations in action space, and state space representation. One big difference between the two is in how problem solvers need to model the environment of the DRL agent. In driving tasks, there is more uncertainty in the state space representation than in classic motor control problems. Is that a motorcycle in the blind spot? Is that car stopped to let a pedestrian pass in front of it? Humans as well as agents need to adapt to the uncertainty in the driving environment. DRL agents generally do this by adopting RNN architectures, and model their MDP as a POMDP.

One of the first major works in DRL (Sallab et al., 2017) that saw an application in autonomous driving was the seminal work from the European Valeo Automotive company by researchers Sallab et al. in 2017. They performed a benchmark on DQN, and a Deep Deterministic Actor-Critic network known as DDAC, and in both cases, the networks were updated to include recurrent LSTM layers to better handle the POMDP nature of the problem. Further, convolutional layers are used to promote an attention mechanism to better encode a state space representation of the simulation. The researchers used the Open-source Racing Car Simulator known as Torcs with the Simulated Car Racing add-on to allow precise control of the agent. The researchers used the simulator frames as an input, which includes an overhead shot of the car, and in the upper-left hand corner, there is a small square with the driver's point of view. These features are fused with details about the car's current orientation, velocity, and relative distance to the race track border. This work is a little different from the rest of the works we will survey as the input to the DRL networks are a little different, and only available to virtual problems, and additionally, the work was not intended to be translated into the real-world. Nevertheless, they have laid out a viable framework for researchers who wish to bring this technology to the open road.

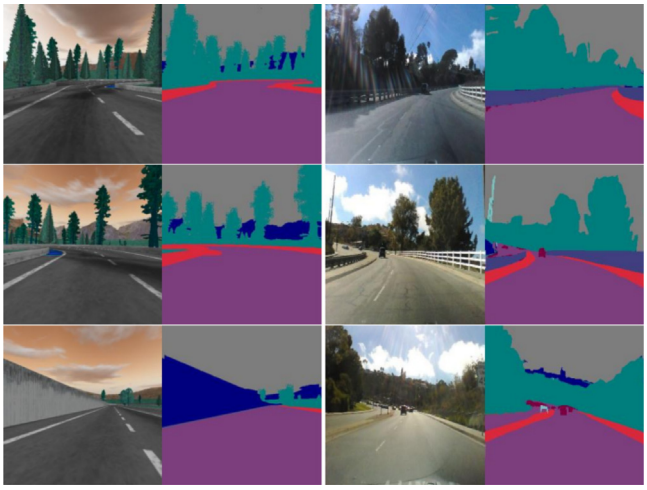


Fig. 4. This figure shows the common state space representation between virtual and real-world environments when Mask-RCNN is used on the opposing RGB inputs (Pan et al., 2017). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The same year, a team of researchers Pan et al. (2017) developed the first attempt to train an autonomous vehicle in simulation and then evaluate in the real-world. The researchers used A3C. They were motivated by the prohibitive cost of training an autonomous vehicle in the real-world. One of the largest contributions the paper makes is in the way the researchers bridge the reality gap from virtual simulation to real-world. The authors of this paper have a driver's point-of-view provided by a camera in both the real-world and virtual environments. Instead of passing the raw RGB pixel data to the neural network, the researchers perform image segmentation using Mask-RCNN. As shown in Fig. 4 reproduced with permission from Pan et al. (2017), although the RGB state space representations look different, the addition of Mask-RCNN into the pipeline provides a similar state space representation within both domains. The authors showed that their model could train in simulation and then adapt to real-world data successfully bridging the reality gap.

Following this logic we see one of the first domain specific emulators created for autonomous driving tasks. The CARLA driving simulator (Dosovitskiy, Ros, Codevilla, Lopez, & Koltun, 2017) was created in late 2017, and provides three state space representations for a driver POV in simulation. The three sensing modalities are the RGB camera, a depth map, and finally, image segmentation of the RGB camera frames. Already cited more than a thousand times, the CARLA autonomous driving simulator has provided a springboard of success to researchers everywhere.

A 2020 paper (Duan, Eben Li et al., 2020) shows the importance of framing a reinforcement learning task as a hierarchical reinforcement learning task. They learn sub-policies for motor control tasks to perform a left lane change, a right lane change, and holding in a lane. There is a master policy that learns when to utilize these sub-policies. Each sub-policy is implemented by a different neural network trained with unique state space representations and reward functions. The authors noted that compared to a model that does not use a hierarchical learning framework, the HL method provides higher rewards, and requires less training. The preliminary work is promising and future work is supposed to evolve to more complex driving environments like city driving where pedestrians become a factor.

As the world gets ready for more and more autonomous cars to be on the road, the research community has turned their attention to issues like designing an autonomous cruise control to minimize the effects on traffic (Das & Won, 2021), algorithms to control autonomous cars operating in complex driving scenarios (Li et al., 2021), and controlling a hybrid electric car to minimize the effects on the environment (Zhu et al., 2021). In Das and Won (2021) the researchers seek to control the inter-vehicle gap and other safety parameters based on micro and macro traffic conditions. By using a deep Q network to select a safe "time-to-collision" or TTC value and a second network takes this selected TTC as an input and outputs a speed for the adaptive cruise control. Using the architecture this way to create a latent space that refers to a real-world value is reminiscent of the work we will see in Xu et al. (2020). These architectural changes promote auxiliary learning and makes training more efficient for complex and intractable state and action spaces. The proposed algorithm SAINT-ACC performs in simulation as effectively as other state-of-the-art approaches, but appears better than state-of-the-art algorithms in more complex driving scenarios (e.g. increased car density).

We also see researchers trying to control for complex driving situations in Li et al. (2021) from 2021. This team uses a hierarchical framework where a double DQN acts as a high level coordinator of lower motor controllers. The lower level controllers are governed by a series of Constrained Iterative Linear Quadratic Regulators. Each regulator corresponds to a task that is determined by the higher level double DQN. While navigating through roundabouts or larger intersections these tasks are intermediate steps to driving, such as: entering a busy intersection, exiting to an off ramp or merging from an on ramp, waiting until it is clear to turn, or following a vehicle. The proposed algorithm achieved high completion rates and low collision rates despite needing to brake hard in some cases to avoid collision. The usage of the hierarchical framework alleviates the action and state space intractability caused by having different motor behaviors depending on the specific task similar to Duan, Eben Li et al. (2020).

In our final paper on autonomous driving we see in Zhu et al. (2021) an attempt to control a hybrid electrical autonomous car with a DRL algorithm to preserve fuel and maintain optimal speed. The DRL in question is a Proximal Policy Optimization (PPO) trained with LSTM to better handle the POMDP nature of the problem. There is also significant work presented on a custom reward function. In simulations the proposed DRL algorithm decreases fuel consumption by 17% compared to more computationally demanding algorithms.

3.3. Navigation tasks

Navigation problems present themselves in both the arena of general motor control policies, and the arena self driving vehicles. We will discuss two different types of navigation here, path planning in general, and then path planning for odometry and SLAM problems in particular. Path planning is an important aspect of autonomous robots. Unlike a SLAM or odometry problem which requires creating and storing a map of the environment from sensor data, path planning algorithms can be as simple as avoiding obstacles while completely forgetting previously seen environments as agents navigate to a target. There are several important works that incorporate DRL into path planning tasks (Faust et al., 2017; Lei et al., 2018; Lv et al., 2019; Pfeiffer et al., 2017; Tai et al., 2017; Yan et al., 2021; Zhang et al., 2021). And a handful that (directly or indirectly) involve SLAM (Placed & Castellanos, 2020; Wang, Clark, Wen and Trigoni, 2017; Wen et al., 2020; Zhang et al., 2020).

The seminal work of Pfeiffer et al. (2017) in 2016 claims to be the first end-to-end framework for an autonomous robot to

perform mapless obstacle avoidance on the way to a known target destination. Using a Kobuki based turtlebot the researchers train ResNet to act as their DQN. One year later in 2017, a research team published (Tai et al., 2017) which develops a similar framework that built around an asynchronous DDPG. Both teams train their model in simulation and successfully evaluate their model in the real-world. Perhaps the most impressive feat of all was a transfer to real-world environments that had not been seen in simulation, suggesting great generalizability. Depth maps can be used in simulation and real-world providing a state space representation common to both the simulation and real-world. This shared feature space helps eliminate the reality gap. Both papers successfully and maplessly navigate their turtlebots through real-world environments.

In 2017 we also see the first attempt from Faust et al. (2017) to reframe the path planning task as a hierarchical learning problem and provide enormous generalization benefits. The framework presented is known as PRM-RL. Probabilistic Road Maps are used for choosing a route to a destination. A subagent then controls the motors to reach the chosen target. The ground experiments use DDPG, while the aerial drone experiments use Continuous Action Fitted Value Iteration. Having a non-DRL agent create the path, and subsequently giving the DDPG agent in the ground experiment a known target destination, the problem becomes very similar to the path planning algorithms discussed in the previous paragraph. Both ground experiments and aerial experiments saw great generalization when transferred to the real-world.

Path planning researchers have made some unique contributions to the architectures of traditional DRL agents. As seen in this 2019 paper (Lv et al., 2019), researchers have actually extended dense residual connections to the DQN architecture on almost 4 years after residual connections were suggested in the creation of ResNet. Overall, their dense model is more stable, faster, and accurate than traditional DQN.

The final path planning papers we will explore before moving onto SLAM, are all newer and may be of interest to path planning engineers (Yan et al., 2021; Zhang et al., 2021). In Zhang et al. (2021) the researchers do not design a path planning algorithm; they use hierarchical learning to create a tracking algorithm. Many path planning applications involve following a moving target, and this algorithm provides a state-of-the-art real-time approach to doing so. They call this algorithm PAC-Net because it constituted by a policy network for switching modes and an actor–critic network with an LSTM layer to perform searching and bounding box regression within the frames. Both hierarchical decision networks share a ResNet18 feature extractor for observation. In the paper (Yan et al., 2021) the researchers handle the unique problem of path planning for flocks of autonomous drones. The researchers propose a distributed DRL actor critic network to overcome the flocking control and collision avoidance problem for a squad of autonomous aerial vehicles. A leader robot controlled by a human is followed by the flock who are trained with parameter sharing to avoid collisions and follow the leader within a desired threshold. The researchers showed they could transfer the model to new experiments without the need of fine-tuning showing great promise for real-world implementation.

As robotics researchers will tell you, odometry is perhaps one of the more important aspects of any autonomous vehicle framework. It is great that autonomous vehicles can learn these control policies discussed here in this survey, but what use are these policies if the autonomous robot looses track of where it is within the environment? Odometry is the science (some say art Poddar, Kottath, & Karar, 2018) of using sensor data to verify the current position of a vehicle. In autonomous vehicles this is often extended to a simultaneous localization and mapping problem — or a SLAM problem. The vehicles need to confirm its location within the environment but also need to map and store the sensor data for navigation tasks. Deep reinforcement learning has already shown to provide some answers to this complex problem. As we have seen before, traditional deep learning was used to develop DeepVO to perform visual odometry (Wang, Clark et al., 2017). In a 2018 paper from Nanjing University (Lei et al., 2018), we see the researchers use episodic learning with a Double-DQN to learn to navigate a robot and create a world map as it progresses to a target while avoiding obstacles. The only draw back to the map created is that obstacles occlude parts of the environment and the learning agent does not adopt a route to fully explore the environment. By relying on depth maps for input, the model overcomes the reality gap when trained in simulation and evaluated in real-world environments.

A paper published in 2020 in Applied Sciences (Placed & Castellanos, 2020) makes use of a Dueling Double DQN to successfully explore an environment gathering enough features to perform SLAM using gmapping in ROS. The researchers get these results primarily by developing a handcrafted reward function that promotes full exploration. The reward function is below:

$$R_{aug} = \begin{cases} -100 & \text{if collision} \\ 1 + tanh(\frac{\eta}{f(\Sigma)}) & \text{if } \omega = 0 \\ .05 + tanh(\frac{\eta}{f(\Sigma)}) & \text{if } \omega \neq 0 \end{cases} \tag{8}$$

where, $f(\Sigma)$ is the D-optimality criterion, dependent on the eigenvalues of the covariance matrix of the observed features. This equation is given below.

$$f(\Sigma) = \text{D-opt} = exp(\frac{1}{l}\sum_{k=1}^{l} log(\lambda_k)) \tag{9}$$

The variable η is a scaling factor that for these researcher's experiments is set to .01. And l is the dimension of the observed features. In essence, this means when the observed data has higher variance, or uneven variance, the reward is lower. When the observed features have a lower variance then the reward is higher. The researchers also experiment with using Shannon's entropy in place of the D-optimality criterion. This reward function successfully promoted enough exploration to gather enough features so that gmapping could perform SLAM and detect loop closure. This work is preceded by another 2020 paper (Wen et al., 2020) from researchers who used FastSLAM instead of gmapping to perform SLAM using the LiDAR range findings from their robot that successfully navigates through a dynamic environment while avoiding obstacles.

A research team assembled from the University of Freiburg and the Hong Kong University of Science and Technology published a paper (Zhang et al., 2020) in late 2020. This paper, unlike the previous mentioned papers, seeks to use DRL to perform SLAM directly. They utilize A3C and train in simple gridworld like environments successfully evaluating their work in more realistic Gazebo environments. The researchers take advantage of external memory access mechanism to account for the increased memory and computational demands associated with creating a map of the environment.

3.4. Lessons learned

Across a lot of these tasks we see continuous action spaces and similar state space representations. When the action space is continuous we see the usage of actor–critic algorithms like DDPG. In the case the autonomous driving, the unique feature is that the MDP is redefined as a POMDP system and therefore we are more likely to see usage of recurrent neural network architectures like LSTM (Sallab et al., 2017; Zhu et al., 2021).

Researchers utilize many of the famous DRL algorithms in these applied settings; DQN (Das & Won, 2021; Lei et al., 2018; Li et al., 2021; Lv et al., 2019; Omisore et al., 2018; Sallab et al., 2017), and actor–critic models like DDPG (Faust et al., 2017; Gu et al., 2016; Peng et al., 2018; Popov et al., 2017; Tai et al., 2017; Tan et al., 2019), and A3C (Pan et al., 2017; Rusu et al., 2016a) are the most popular algorithms used by Researchers for motor control tasks. State space representations generally rely on visual features (Pan et al., 2017; Rusu et al., 2016a; Sallab et al., 2017) or depth maps from systems like LiDAR (Dosovitskiy et al., 2017; Lei et al., 2018; Pfeiffer et al., 2017; Tai et al., 2017). In at least one case we explored they use both RGB and Depth as an input (Zhao et al., 2021). In general, techniques from traditional deep learning are slow to be introduced into the field of DRL. As referenced before, the 2019 paper (Lv et al., 2019) is one of the first papers in the DRL field that use a residual feedforward connection in their network architecture. Please note that residual connections were introduced in 2015 (He, Zhang, Ren, & Sun, 2015) as the most critical part in the ResNet feature extractor and classifier, but remain not widely adopted by the DRL community.

The field of robotics and autonomous vehicles is the first place we see the development of simulation methods that bridge the "reality gap". In motor control tasks, we saw the usage of progressive networks as an early attempt to bridge the reality gap, although this led to an increased demand in computational resources (Rusu et al., 2016a). Autonomous vehicle research involves using visual features as an input, so the usage of Mask-RCNN and semantic or instance segmentation is a very clever way of overcoming the reality gap in the autonomous driving papers we have surveyed (Dosovitskiy et al., 2017; Pan et al., 2017). In autonomous vehicle tasks we also saw the use of an emulator where depth maps can be extracted, or where an image segmentation algorithm like Mask-RCNN can be applied to create a similar state space from virtual and real-world camera inputs (Dosovitskiy et al., 2017).

The development of reward functions to serve as a method of auxiliary learning is more prevalent in the resource allocation task section. However, there are still a few prominent examples of reward function manipulation guiding networks to better solutions. Of course this is alluding to Honerkamp et al. (2021), Placed and Castellanos (2020), Popov et al. (2017) and Yan et al. (2021). The usage of intrinsic motivation alongside reward function manipulation has also lead to promising results (Qureshi et al., 2018).

Throughout this section we also see the emergence of the use of hierarchical learning to reform complex problems with intractable state and/or action spaces into simpler systems (Duan, Eben Li et al., 2020; Faust et al., 2017; Li et al., 2021; Liu & Jiang, 2018; Zhang et al., 2021). Where HL is used we see a reduction in the time of training (Duan, Eben Li et al., 2020), and in some cases algorithms created with a HL framework generalize better to environments that are never seen before in the real-world (Faust et al., 2017).

4. Resource allocation

Resource Allocation can refer to a lot of different tasks. In this section we will discuss traffic control, vehicle routing, and the control of telecommunication resources, before finishing our discussion with automating HVAC control, and management of the power grid. Traffic engineering is not traditionally classified as a resource allocation task, but it shares similarities. At the heart of the problem is the flow of resources or "traffic". Researchers seek to maximize the flow of traffic in order to decrease traffic congestion. Vehicle routing is more obviously a resource allocation task, where the fleet of taxis need to be distributed according to customer demand. The rise of companies like Lyft and Uber have caused increased focus on this problem domain in recent years. Generally speaking, telecommunication resource tasks revolve around decreasing power consumption or increasing bandwidth utilization. Table 2 summarizes the key take aways from the papers surveyed within this section.

Similar to the other application settings for cyber–physical systems, the usage of neural networks in human aided decision making for resource management predates the usage of deep reinforcement learning. Researchers have been shown to be able to predict demand within a power grid (Luh, Michel, Friedland, Guan, & Yuting Wang, 2010; Marinescu, Harris, Dusparic, Clarke, & Cahill, 2013) and predict energy consumption in a specific building with a HVAC system (Mocanu et al., 2016). Traditional reinforcement learning has also been used alongside neural networks to try better respond to market conditions (Lu & Hong, 2019). Similarly, there is important work (Ke, Zheng, Yang, & Chen, 2017a) using traditional deep learning in the study of vehicle routing tasks.

These tasks are very similar in general, and rely on hand-crafted reward functions and auxiliary learning for some of the major breakthroughs. HVAC control, while similar to power grid management, differs from other resource allocation tasks in the sense that the credit assignment problem appears due to the long lengths of time between an agent's action and the result. Generally when the credit assignment problem appears in other resource allocation task it is due to the rapidly changing nature of the state space. A warehouse may require hours of heating or cooling before reaching a desired temperature. HVAC control is further complicated by a need for enormous amounts of data. A paper we will talk about later (Xu et al., 2020) mentions two additional works (Wei, Ren, & Zhu, 2019; Wei, Yanzhi Wang, & Zhu, 2017) that require in upwards of 50 to 100 months of data in order to have their models trained properly. More complex systems require in upwards of more than 4000 months of data.

4.1. Traffic control

Traffic control got a lot of attention starting in 2016 with the publication of two seminal works (Genders & Razavi, 2016; Li et al., 2016). The first of these papers (Li et al., 2016) was published in July by a research team assembled from the Chinese Academy of Sciences and Tsinghua University. Using a modified DQN with four auto-encoding layers, the team compared the performance of DRL and RL and observed that DRL outperforms traditional RL algorithms. As one of the first works in this domain, the team limited the state and action spaces by not allowing turns at four-way intersections, nor did they allow yellow lights to present, and red lights occur for a fixed amount of time. The action space was limited to picking which streams have the green light. In the following work (Genders & Razavi, 2016) published in the same year, we see the proposal of a state space representation referred to as the DTSE — discrete traffic state encoding. This state space is shown in Fig. 5 reproduced with permission from Genders and Razavi (2016). The two channels represent one of the four streams of traffic in the intersection that act as the input to the convolution based DQN. This work advances over the previous, by allowing cars to make left and right turns at the intersection. However, similar to the previous work, this agent can only decide which light would be green at a given moment, with fixed intervals of yellow and red lights to follow.

Both Casas (2017) and Van der Pol and Oliehoek (2016) improve on the above results. However, there are still outstanding questions and problems that need to be solved. Elise van der Pol and Frans A. Oliehoek from the University of Amsterdam made some success with a multi-agent DQN implementation with three

Table 2
Summary of papers surveyed for resource allocation tasks. Note that papers that use simulators in their training regimen may be set up for success in the real world, but unless evaluated in the real-world we hesitate to say the research has overcame the reality gap problem.

Paper	Network	Application	Virtual environment	HL	MAL	Problem overcame	Notes
Genders and Razavi (2016)	DQN	Traffic control	SUMO	N	N	Intractability of state/action space	Proposes DTSE
Van der Pol and Oliehoek (2016)	DQN	Traffic control	SUMO	N	Y	Intractability of state/action space	First to use multi-agent learning
Li, Lv, and Wang (2016)	DQN	Traffic control	PARAMICS	N	N	Intractability of state/action space	Uses autoencoders instead of conv. layers
Casas (2017)	DDPG	Traffic control	Aimsun	N	N	Intractability of state/action space	Limits state space the least
Xu, Wang, Wang, Jia, and Lu (2021)	HiLight	Traffic control	CityFlow	Y	N	Lack of training data + Intractability of state/action space	Distributed HL system using neighborhood statistics
Lin, Zhao, Xu, and Zhou (2018)	DQN + A2C	Vehicle routing	Custom	N	Y	Intractability of state/action space	First to use multi-agent learning
Liu et al. (2020)	DDPG	Vehicle routing	Custom	N	Y	Intractability of state/action space	Confirms usefulness of multi-agent learning
Al-Abbasi, Ghosh, and Aggarwal (2019)	DeepPool + DQN	Vehicle routing	Custom	N	N	Intractability of state/action space	One of the first to tackle ride pooling
Xu, Wang, Tang, Wang, and Gursoy (2017)	DQN	Telecommunication	Custom	N	N	Intractability of state/action space	Handcrafted reward function to minimize power usage
Liu et al. (2017)	DQN + DDPG	Telecommunication	Custom	N	N	Intractability of state/action space	Applied work from Li, Xu, Tang, and Wang (2018)
Sun, Peng, and Mao (2019)	DQN	Telecommunication	Custom	N	N	Intractability of state/action space	Handcrafted reward function to minimize power usage
Chen, Lingys, Chen, and Liu (2018)	DDPG	Telecommunication	Custom	Y	N	Intractability of state/action space	Distributes agents to hosts
Ruffy, Przystupa, and Beschastnikh (2018)	Benchmarks multiple agents	Telecommunication	Iroko	N	N	Lack of training data	Proposes Iroko data center emulator
Chinchali et al. (2018)	DDPG	Telecommunication	Custom	N	N	Intractability of state/action space	Handcrafted reward function to maximize traffic flow
Xu, Tang, Yin, Wang, and Xue (2019)	DQN	Telecommunication	Real-world	N	N	Intractability of state/action space	Handcrafted reward function to maximize traffic flow
Kaviani et al. (2021)	DeepCQ+	Telecommunication	Custom	N	Y	Intractability of state/action space	Used MARL in Vehicle Platooning communication
Cao and Yin (2021)	DQN	Telecommunication	MATLAB	N	N	Lack of training data	Decreased packet collisions compared to baseline
Wang, Velswamy and Huang (2017)	Recurrent actor–critic network	HVAC	EnergyPlus	N	N	Lack of training data + Intractability of state/action space	Uses LSTM to overcome POMDP nature of problem
Yu et al. (2020)	MAAC	HVAC	Custom	N	Y	Lack of training data + Intractability of state/action space	Multi-agent learning reduces intractability

(*continued on next page*)

sequential traffic lights (Van der Pol & Oliehoek, 2016). Taking lessons from van der Pol's master's thesis (van der Pol, 2016), they limited the state space by disallowing turns or by experimenting in environments where a turn is not possible, like the three sequential traffic light scenario in the paper. A later paper in 2017 from Noe Casas (Casas, 2017) showed that DDPG could outperform a traditional non-DRL multi-agent Q-learning algorithm in realistic scenarios, allowing turns, and allowing control of yellow and red light sequences. Casas shows that DDPG scales very well, up to a three by two grid of four-way intersections. However, DDPG falls short in providing stable traffic in the most realistic scenario involving a section of a model city. In this sense, DDPG better handles the problem's innate intractability compared to traditional RL but has a little ways to go before we can call this problem solved.

The most recent state-of-the-art traffic light control research is Xu et al. (2021), a 2021 paper from Peng Cheng Labs and Peking University. The researchers note their predecessor's work

Table 2 (*continued*).

Paper	Network	Application	Virtual environment	HL	MAL	Problem overcame	Notes
Xu et al. (2020)	2 Novel DNNs	HVAC	EnergyPlus	N	N	Lack of training data + Intractability of state/action space	Transfer learning reduces amount of data needed
Diao et al. (2019)	DQN	Power grid management	GridPACK + IEEE 14-bus System	N	N	Lack of training data	Develops grid mind framework
Duan et al. (2020)	DQN + DDPG	Power grid management	Custom	N	N	Lack of training data	Refines grid mind and experiments with DDPG
Chen et al. (2020)	PowerNet (from IA2C)	Power grid management	Custom	N	Y	Lack of training data + Intractability of state/action space	Proposes using MAL to manage power grid
Yang et al. (2018)	DQN	Power grid management	Custom	N	Y	Lack of training data + Intractability of state/action space	Proposes using MAL to establish pricing
Wei, Wan, and He (2020)	DDPG + DQN	Power grid management	Custom	N	N	Lack of training data	Proposes cyber attack recovery algorithm

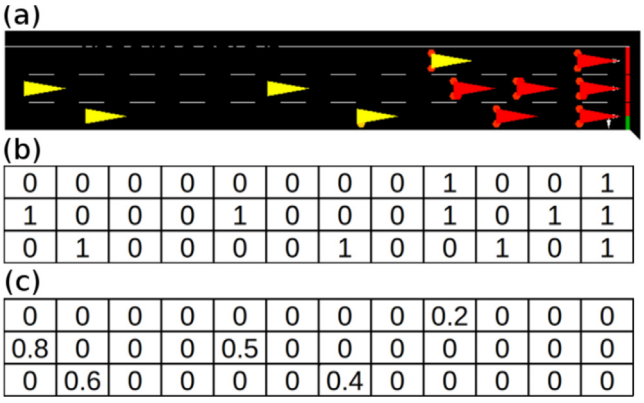


Fig. 5. This figure shows the Discrete Traffic State Encoding or DTSE. (a) refers to a lane within the simulation. (b) refers to a channel holding boolean values indicating the presence of a car in the corresponding cell within the simulation. And (c) refers to the normalized velocities of the cars (Genders & Razavi, 2016). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

of identifying suitable short term rewards such as minimizing queue lengths, waiting times, and delays. It would be preferable to minimize a long term goal of total time in car for every individual driver however due to the time frames involved this becomes difficult for local traffic light operators to measure. To achieve something like this the authors make use of hierarchical learning to great effect in an algorithm they call HiLight. The researchers use an actor critic method to select a sub-policy that is implemented in an intersection for training. Each sub-policy tries to optimize for one of the short term goals iterated before. However, what is unique here is their usage of a multi critic algorithm, one critic looks solely at the local intersections short-term goals, and the second one criticizes the proposed policy according to overall neighborhood traffic statistics. The sub-policies are updated with their short term rewards at every timestep at every intersection, where the higher level controller is rewarded at the time interval of T timesteps. The system adaptively weighs these two critics in order to balance achieving short term traffic goals along with longer term neighborhood goals when selecting sub-policies for local agents. In simulations, HiLight outperforms other cooperative traffic lighting algorithms. The other traffic controllers we have explored are states-of-the-art in their domain of non-cooperative local agents. However HiLight and its usage of hierarchical learning that incorporates neighborhood traffic statistics to learn longer term rewards is much closer to solving the real-world traffic controller problem than any local non-cooperative agent.

We will end our discussion of the traffic control problem, by referring the reader to valuable benchmarks and similar research that may be interesting to the traffic control engineer. Li et al. (2016) uses the PARAMICS traffic simulator (Smith, Duncan, & Druitt, 1995) while Genders and Razavi (2016) and Van der Pol and Oliehoek (2016) use the SUMO traffic simulator (Krajzewicz, Hertkorn, Feld, & Wagner, 2002). Meanwhile, Noe Casas, author of Casas (2017), used AIMSUN (Barcelo, Codina, Casas, Ferrer, & García, 2005). In Xu et al. (2021) they use CityFlow (Zhang et al., 2019). In Noe Casas's work they explain that there are microscopic and macroscopic traffic simulators. SUMO and PARAMICS are microscopic, and AIMSUN can work as either microscopic, or macroscopic, or even as hybrid of the other two known as mesoscopic (Casas, 2017). SUMO has wider applications than solely traffic, and allows users to simulate CO2 emissions, and contains datasets corresponding to traffic conditions at real-world events (Krajzewicz et al., 2002). Lastly, we would like to point out to the reader the research from the University of Tokyo. In 2016, DeepTransport was created (Song, Kanasugi, & Shibasaki, 2016), and while this work does not incorporate DRL, we think this could potentially prove useful to engineers working on traffic control problems. DeepTransport is a deep learning model that can predict and generate realistic routes individuals within Tokyo may travel along. Using a recurrent network, and data gathered from GPS data in Tokyo, this network could prove useful in creating realistic datasets for simulations that use Tokyo as a testing or training environment. Similarly, this work can be replicated for any city that offers samples of GPS route data to researchers.

4.2. Vehicle routing

A very important paper in the discussion of vehicle routing is the 2017 work (Ke et al., 2017a) from Zhejiang University and Hong Kong University of Science and Technology. These researchers created a novel deep learning network that does not incorporate deep reinforcement learning, in order to predict customer demand for vehicle routing services. The work is interesting on its own despite not involving DRL, but it remains important to DRL researchers because the authors of this paper scrutinized the state space for a vehicle routing platform. These researchers used a spatial aggregated random forest algorithm to determine which state space variables have the greatest impact on predicting user demand. Fig. 6 reproduced with permission from Ke et al. (2017a) describes this state space.

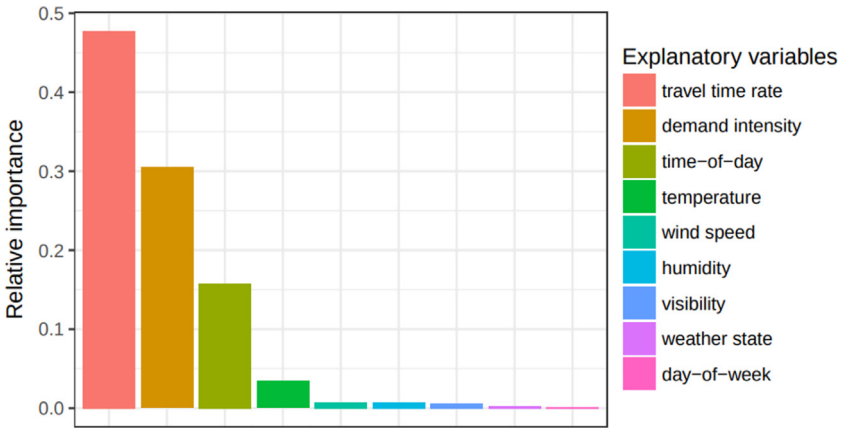


Fig. 6. This graph presents which state space variables have the greatest effect on predicting future user demand in vehicle routing platforms. Intuitively the rate of the ride is the largest factor in determining user demand, but that is closely followed by current demand intensity (or the amount of current ride requests). Time of day and temperature are also important (Ke et al., 2017a). Copyright ownership is maintained by Elsevier (Ke, Zheng, Yang, & Chen, 2017b). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Turning our attention back to deep reinforcement learning, we must now look at work presented in 2018 (Lin et al., 2018) by a research team from Michigan State University in conjunction with Didi Chuxing — a Chinese Vehicle Routing platform. These researchers presented two multi-agent DRL algorithms that they refer to as contextualized DQN, and contextualized A2C — a non-asynchronous version of the Asynchronous Advantage Actor–Critic (or A3C) algorithm. The state space included a geographic map of the city, and a contextualized map showing the locations of other agents/vehicles. The map of the city was overlaid with a grid, and the action space directs agents to these grids. They performed their experiments in a handcrafted simulator using actual passenger data provided by Didi Chuxing. The researchers declared their work a success.

A large ensemble of researchers from various Chinese Universities in 2020 updated their 2019 research on using DRL to route an electric vehicles to charging stations (Liu et al., 2020). Using a SDN-enhanced vehicular edge computing network to monitor the battery of the electric vehicle locally. The DRL agent seeks to minimize overhead in both travel time to charging stations when the electric vehicle battery is low, and the charging cost at a particular charging station. The researchers use a deep neural network to approximate the quality function similar to DQN, and have a state space including vehicle position and the state of the battery. Selecting from a grid overlaid a city, the agent directs cars to available fast charging stations. Interested readers should explore these researcher's hand crafted reward function that takes advantage of domain knowledge to model vehicle overhead in this problem. Comparing themselves to an algorithm using game theory to make reservations at the fast charging stations available, the DRL agent both minimizes overhead more effectively and also requires less computational resources to compute.

A subset of problems within the vehicle routing domain revolve around a newer idea referred to as ride-pooling. Uber and Lyft for cheaper rates allow multiple passengers to be picked up at a time, and like a bus, the vehicle delivers passengers to their respective locations. In 2018, researchers in Purdue University developed DeepPool (Al-Abbasi et al., 2019). Using a distributed DQN to route vehicles to passengers, these researchers claim to be the first DRL solution to vehicle dispatch problem to incorporate user demand statistics into their state space. By distributing a DQN for each agent in the ride sharing program, the individual DQN's handle the problem's intractability compared to having a single DQN making decisions for the entire fleet. This echos the research of Ke et al. (2017a). And while the researchers showed success using real-world taxi data from NYC in simulations, they did feel required to limit the state space to disallow separate exits from the vehicle. The vehicles will travel around picking up multiple passengers, but the simulation forces these passengers to be dropped off at the same area and at the same time. Despite this minor set back, these researchers developed a framework that adapts well to new dynamic environments, and successfully avoids dispatching vehicles to low-demand areas.

4.3. Telecommunication resources

Radio Access Networks (RANs) have become increasingly prevalent especially as we get closer to ubiquitous 5G wireless communication. RANs provide access to some core network using radio technology. There are a variety of RANs including Cloud based RANs, or Fog RANs. While relying on different technology for their implementation, they all serve the same purpose, and DRL can be used to help mitigate power consumption. However, this reduction in power usage cannot come at the expense of user experience, or bandwidth utilization. As such, reward functions will be of a high importance when investigating this technology domain.

An important contribution to the domain came in 2017 (Xu et al., 2017) when a team of researchers at Syracuse University made an effort to decrease the power consumption without sacrificing user demand satisfaction. Users and groups of Remote Radio Heads (RRHs) are divided into cells based of proximity. Performance and power consumption are measured at each node, and a centralized agent makes decisions to place RRHs into a sleep mode, or to turn them on, and when turned on applying a weight to the RRH's beamforming to reflect current demand. Using a self-implemented simulation designed in Tensorflow, the researchers compare their work to two other state-of-the-art algorithms and show improvement on power consumption without sacrificing user experience; in one case they use 18% less power than one of their state-of-the-art alternatives. The agent also adapts to highly dynamic environments. The researchers used industry knowledge to handcraft a reward function that evaluates how the agent minimizes power consumption, and balances user demand by finding an optimal beamforming weights. The reward is given by:

$$R_t = P_{max} - P(S, A, G) \tag{10}$$

where P_{max} is the maximum possible power consumption, and

$$P(S, A, G) = \sum_{r\in A}\sum_{u\in U}\frac{1}{m}|w_{r,u}|^2 + \sum_{r\in A}P_{r,active} + \sum_{r\in S}P_{r,sleep} + \sum_{r\in G}P_{r,transition} \tag{11}$$

Here, $P(S, A, G)$ is the total power consumption for all active RRHs in set A, for all sleeping RRHs in set S, and for all transition RRHs in set G. $P_{r,active}$, $P_{r,sleep}$, and $P_{r,transition}$ are the respective power for each RRH r in each of three sets. Finally, $w_{r,u}$ are the beamforming weights for each user connected to a RRH. While the agent tries to minimize this $P(S, A, G)$, the agent simultaneously attempts to solve this optimization problem meant to find the optimal beamforming weights for each user connected to the active set of radio heads:

$$\underset{w_{r,u}}{\operatorname{argmin}} \sum_{r\in A}\sum_{u\in U}|w_{r,u}|^2 \tag{12}$$

Subject to:

$$SINR_u \geq \gamma_u,\ u \in U; \qquad \sum_{u\in U}|w_{r,u}|^2 \leq P_r,\ r \in A; \tag{13}$$

where,

$$\gamma_u = \Gamma_m(2^{R_u/B} - 1) \tag{14}$$

Here $SINR_u$ is the signal to noise ratio for each user's connection to a RRH.

This previous work was built upon in both Li et al. (2018) and Liu et al. (2017) by members of the same team as Xu et al. (2017). In Li et al. (2018), the team focused on the general-purpose Distributed Stream Data Processing System (DSDPS), which processes unbounded streams of continuous data at scale distributed in real-time. They focused on the adaptive and systematic solution to the fundamental scheduling problem (i.e., assigning workload to workers/machines) with the objective of minimizing the average end-to-end tuple processing time. They developed a novel model-free approach using DRL that can learn to well control a DSDPS from its experience rather than accurate and mathematically solvable system models, just as a human learns a skill. Compared to Storm's default scheduler and the state-of-the-art model-based method, the proposed framework reduces average tuple processing by 33.5% and 14.0%, respectively. Moreover, in another work (Liu et al., 2017), this team applied the DRL technique from Li et al. (2018) to solve the cloud computing request dispatching and processing problem with high state and action dimensions, consistently outperforming prior work thanks in part to the system's distributed nature. These preliminary works lay a solid foundation on state-dependent, model-free ML/AI with high-dimensional state and action spaces, as a perfect fit to the domains of this project.

A more recent paper from 2019 (Sun et al., 2019) tries to decrease the latency seen in Fog-based Radio Access Networks. Taking advantage of the "edge computing" paradigm, which states that latency in a networking tasks can be decreased by moving the physical location of the processing and data storage closer to where the data ends up. By distributing individual decision making agents to each node the researchers are better positioned to handle the problem's intractability. Using an actor–critic model they employ a network similar to A3C to lower the latency for users. The learning agent chooses which user requests to process, where the request will be processed, and how many resources to dedicate to the task. These decisions are decided by the state space: the number of data delivery requests, the number of data processing requests, the number of Fog RAN nodes, the size of the input data for the task, the computational requirements for the task, the popularity of the data being requested, a flag indicating the presence of the data within storage, and a vector of SINR values indicating signal strength between the user an each Fog Radio Head. Ultimately these researchers were able to compare to state-of-the-art algorithms and alternative frameworks using numerical simulations and showed that the DRL method decreased latency the most.

Similar to controlling resources for a RAN, DRL has been applied to automating the resource allocation of more typical datacenters. Datacenters currently rely on TCP congestion control algorithms to fully utilize bandwidth and satisfy user demand with minimal latency. Researchers have tried applying DRL to this problem domain. In 2018, a team of researchers benchmarked current DRL approaches to this problem while also offering their own approach (Chen et al., 2018). They found that previous efforts have fallen short, as most DRL frameworks struggle to make decisions on the millisecond time scale. Proposing a novel DRL algorithm called AuTO (Automatic Traffic Optimization) they mimic the nervous system in animals which have both a central and peripheral nervous system. Similarly to this construct, the researchers have part of the algorithm deployed on peripheral local host systems which make choices to optimize traffic involving shorter data requests, while a central system at the datacenter or server aggregates global information to better inform peripheral systems what traffic to prioritize. The success of this work hinges on the observation that most datacenter traffic can be modeled as a long-tail distribution; which means the majority of datacenter traffic is from long-flow requests, while the majority of requests are in fact for short-flow requests of data. The peripheral system on the local hosts makes decisions about when to process the shorter requests when the central system makes scheduling decisions (for under 1 s) concerning longer-flow requests for data. Another important development is the hand-crafted reward function the researchers developed seen below:

$$r_t = \frac{\sum_{f^t\in F_t^D} Tput_f^t}{\sum_{f^{t-1}\in F_{t-1}^D} Tput_f^{t-1}} \tag{15}$$

Here F_t^D is the set of completed flows at time step t and, $Tput_f^t$ is the average throughput of each completed flow at time step t. This reward function is a ratio of the completed flows average throughput for the current time step over the previous time step. This ratio signifies an increase or decrease in throughput based on the agent's actions. The authors choose not to use simulation for training or evaluation, they use their own thirty-two server testbed to train and develop the end-to-end algorithm. The researchers concluded that AuTO works better than previous traffic optimization approaches and they credit the decoupling of their central and peripheral systems for this success.

As always, benchmarks developed within an application setting can prove to be important springboards for future research and success. Iroko in 2018 (Ruffy et al., 2018) is an emulator developed by a team of researchers from the University of British Columbia. Iroko aims to provide a benchmark for DRL algorithms meant to automate congestion control in data center settings. The paper benchmarks two TCP congestion control algorithms alongside DDPG, as well as other RL algorithms such as RIENFORCE, and Proximal Policy Gradients. DDPG alongside PPO outperform TCP New Vegas which is ubiquitous in networking solutions.

A research team at Stanford University has studied a similar problem in 2018 (Chinchali et al., 2018), but applied to cellular telephone networks instead of datacenters. They too wish to increase traffic throughput over modern alternatives. Using actual

network data recorded in Melbourne Australia, these researchers show that pairing human operators with hand-crafted reward functions, their agent can increase throughput up to 14.7%. This team built their own simulator to perform training and evaluation. They determined that features relevant to their task included: average user throughput, cell congestion, average cell efficiency, number of connections, and traffic volume. The team uses a recurrent DDPG with LSTM layers to train in their simulation end-to-end, where the agent learns to regress a control policy between [0, 1] for each user's traffic flow. This number corresponds to a rate which the user's traffic can be served on top of conventional traffic rates which generally split the user's bandwidth equally. The researchers come up with this tunable reward function seen below:

$$r(s_t, a_t) = \alpha V_t^{IoT} + \beta V_t^{loss} + k V_t^{belowlimit} \tag{16}$$

Here V_t^{IoT} is a weighted sum of IoT traffic served at time t. V_t^{loss} is the bytes lost due to congestion caused by the IoT traffic flow. $V_t^{belowlimit}$ is the amount of traffic served under a desired threshold. The hyper-parameters α, β, and k are tunable parameters that are controlled by human operators wishing to prioritize one of the parameters as they respond to real-world observations about traffic flow and congestion.

A special case of TCP congestion control algorithms handle the multi-path TCP scenario where traffic can be directed to its target along different paths to increase redundancy, and better handle congestion. Xu et al. (2019) seek to employ DRL to work in this multi-path TCP setting. The experiments show that their model DRL-CC outperforms traditional MPTCP-CC algorithms, can adapt to many dynamic environments, and can be used in traditional TCP environments where multi-path routing is not an option. The DRL-CC algorithm involves a single recurrent DDPG operating on a real-world server communicating with two hosts. The framework presented in this paper allows for easy swapping out of different reward functions to allow end users to prioritize their own network statistics. The researchers used a utility function as the reward function that helps the network learn to increase *goodput* within a network. Goodput is just the utilized bandwidth, because technically speaking throughput refers to a theoretical upperbound on utilization. When people talk about increasing throughput they often mean "goodput". Below is the reward function used by researchers to increase goodput:

$$r_t = \sum_i^N log\ g_t^i \tag{17}$$

In this specific implementation, g_t^i is the average goodput for flow i at timestep t. A unique aspect of the DRL-CC algorithm is the LSTM based feature extractor which can be trained end-to-end with the recurrent actor–critic algorithm DDPG. The feature extractor takes in state variables for each data flow being monitored: the current rate of transmission, current goodput, round trip time, the mean deviation of round trip time, and current congestion window size (a TCP hyperparameter). The LSTM network extracts features from these state variables and passes it to the recurrent DDPG that regresses a proposed change to the TCP parameter congestion window for each flow and subflow (when multi-path TCP routing is an option).

We turn our attention now to mobile ad-hoc networks. In 2021, scientists published Kaviani et al. (2021) which demonstrates the importance of multi-agent DRL in CQ-routing for mobile networks. They have created an algorithm called DeepCQ+ which looks at a mobile network node and determines where to route incoming packets and what mode to send them in (broadcast v. unicast). DeepCQ+ out performs peers in minimizing the amount of transmissions per successfully delivered packets. They do this by taking advantage of the decentralized nature of the problem and having different cooperative agents work at each node. As we have seen in other experiments the usage of one learning agent across all these different nodes can lead to intractability caused by both the state space and action space. Despite the decentralized implementation MARL training can be centralized in simulations through policy parameter sharing. DeepCQ+ outperforms the original CQ+ routing algorithm.

In another paper from 2021, Cao and Yin (2021) deal with automated vehicle platooning for networked vehicles. With the arrival of 5G networks it will become possible for networked cars to communicate over different network types while driving to try and mitigate traffic and accidents. One way of decreasing traffic is by platooning vehicles, which means letting cars stay close together despite maintaining high speeds. The cars then act in concert regarding braking and accelerating by communicating with a platoon leader. In current implementations of this concept experiments suffer from high rates of packet collision because the communication networks used by the cars are selected at random. This team proposes a DRL algorithm that allows the platoon leader to learn the environment's available ad-hoc communication networks to communicate with the platoon cars in order to decrease packet collision probability. The proposed DRL algorithm outperforms current implementations of platoon communication which rely on random selection of communication options. In simulation, they decrease packet collisions by 73% and 45% in low and high density vehicle experiments respectively.

4.4. HVAC control

One of the seminal works in DRL as applied to HVAC control of a building, was a 2017 paper (Wang, Velswamy et al., 2017) published by a research team at the University of Alberta. They used a novel Monte-Carlo recurrent actor–critic network using a REINFORCE policy gradient where the critic network models the value function (differing from other actor–critic networks that model the advantage function, or the quality function). Their largest contribution involves overcoming a lack of real-world data. As mentioned before in HVAC control there is a large credit assignment problem, due to the slow change in state resulting from agent actions. This compounds into a very difficult problem, because this also implies needing large amounts of simulated or real data to train the networks. This 2017 paper is one of the first to use DRL and attains results using only 2 days of training data for an evaluation period of 5 days. The reason for using less data is threefold. (1) The system being controlled is much smaller than the systems referenced before that require months worth of data. (2) The researchers employ different weather schemes in training that give training a sense of a "random initial condition" that we have seen in other works improve generalization. (3) It is not explicit, but the usage of LSTM in their actor–critic network probably helps the agent inform on the new weather patterns as well overcome the POMDP nature of the problem itself. The networks were trained and evaluated in the open source building modeling platform OpenStudio and a Building Control Virtual Test Bed (with EnergyPlus) to allow the agent to control the thermostat. Ultimately this system attains 15% more thermal comfort compared to the alternatives tested against, and utilizes power more efficiently by about 2.5%. The one downside to this framework is that their work is only regulating the HVAC for a single thermal zone, an office suite within a larger building; and additionally, the agent after training cannot easily adapt to new unseen building dynamics. To apply this work in a new building or thermal zone requires retraining the networks.

A very impressive work in 2020 (Yu et al., 2020) from Nanjing, Xi'an Jiaotong, and Huazhong Universities created a novel

multi-agent actor–critic model to control the thermal zones of a building. This work sees a separate cooperative agent responsible for each of the zones. Interestingly enough the system is trained without any explicit indication of the building's thermal dynamics. One key contribution to this paper has got to be the researcher's hand-crafted reward function. The paper dedicates several paragraphs deriving several functions that factor into a single reward function. Their reward function accounts for four separate optimization tasks: (1) a penalty for the energy consumption caused by the air supply fan within the vents. (2) the energy consumption caused by the system's cooling coil. (3) a penalty for the zone's temperature. And finally (4) a penalty for when the concentration of CO_2 exceeds limits. Similar to the previous work, these researchers use the EnergyPlus simulator to model the dynamics of the thermal zones. These researchers utilized real-world weather data and pricing data alongside months worth of training data. These researchers successfully minimize power utilization, while maintaining thermal comfortability, and maintaining CO_2 concentration levels.

The last paper that should be discussed is another recent 2020 paper from researchers at Northwestern, Northeastern, and Texas A&M Universities (Xu et al., 2020). Together they have provided to the research community a novel training regimen that takes advantage of transfer learning to provide high generalization benefits. Training on a simpler version of a problem, with simpler actions, and simpler state spaces, they transfer learn into more complicated systems without requiring larger quantities of additional data. These researchers create a DQN-like architecture with two subnetworks. The front-end network Q is referred to as the building agnostic network that takes the input state I and maps to a latent space ΔT which corresponds to a desired change in temperature in each zone, essentially a form of auxiliary learning. This latent space is then fed forward with the input state I into the back-end network referred to as F^{-1} or the inverse building network which captures the building specific behavior. Fig. 7 reproduced with permission from Xu et al. (2020) details this process more closely. The authors start with a simple building, and initially simplify the actions of the thermostat to be simple on–off controllers and begin training their two networks for as little as two weeks time. After the initial training has occurred, the first subnetwork Q can be brought into a new system with more complicated building dynamics or construction materials, weather conditions, and more complicated thermostat commands. With finetuning the target building is ready in as little as three weeks time. These authors successfully transfer the DRL HVAC controller from a simpler source building to a target building that can have a different number of thermal zones, different building materials and layouts, different HVAC equipment, and even under different weather conditions in certain cases (from more varying weather to less varying weather cases).

4.5. Power-grid management

New technologies are creating demand for new methodologies of power management. Two early examples from 2014 and 2015 in the field of electric vehicles saw efforts to use non-deep reinforcement learning to moderate the power consumption and battery usage of a hybrid electric vehicle (Lin, Bogdan, Chang, & Pedram, 2015; Lin, Wang, Bogdan, Chang, & Pedram, 2014). Both efforts saw a decrease in operational costs of at least 40%. The rise of renewable energy is taking its toll on modern power grids created long ago. For example, houses connected to the power grid that give back solar energy raised on that property cause traditional assumptions about the power grid model to fail, i.e. modern power grids now need to account for a house being a source of power and not solely a power drain. Further, this reversal in demand and supply also leads to non-constant loads on the grid as in the case of solar they are only giving power back to the power grid during sunny days. In 2019, researchers from China and the United States have teamed up to use DRL to try and regulate voltage controls across a smart power grid (Diao et al., 2019). Their proposed DRL agent is called Grid Mind and uses DQN to take measurements of active and reactive power flows on transmission lines. Humans today decide whether to adjust the generator terminal voltage set point, switching shunt elements, or changing the transformer tap ratios to help moderate the voltage across a power grid. The action space for Grid Mind is limited to just altering voltage set points, but future work plans on expanding the scope of actions. The reward function rewards the agent for keeping voltages within operational levels, and penalizes the agent for divergences from operational ideals, and heavily penalizes the agent for violations. Grid Mind can work both online and offline, to continue to learn in real-time. These authors build on their work in the 2020 paper (Duan, Shi et al., 2020) that expands the scope of their experiments to work with DDPG. From these experiments it seems that DDPG handles the problem's intractability better than DQN. DDPG also trains more efficiently. Additionally, the size of their evaluation scenario is increased offering a promising framework for real world systems.

Perhaps unsurprisingly, the state-of-the-art work done on automating the control of a power grid incorporates a multi-agent learning problem formulation. Researchers in late 2020 from Michigan State University, Stanford University, and the Argonne National Laboratory propose a network PowerNet (Chen et al., 2020). PowerNet uses LSTM to overcome the partially observed nature of the state space since agents only have measurements about their local area. Communication between the agents is used so they can share their understanding of their local state space with direct neighbors. The researchers trained their network in a custom simulation called PGSim and their future works section states their desire to add real-world data into the PGSim emulator as opposed to simulated data. The power grid in this paper is decentralized which better mirrors the realities of the modern smart grid. Like the previous work explored, PowerNet uses a state space representation for the active and reactive power, and the currents and voltages at their distributed generators. Using this state as an input, PowerNet decides the voltage setpoint to use at a distributed generator. To handle the credit assignment problem, a spatial discount factor is utilized in the reward function to prevent bad decisions from benefiting from the good decisions of neighbors.

So far we have seen papers solely with regulating that power grid. However, a team of researchers in 2018 from Tianjin University, Imperial College London, and NetEase, Inc. use DRL to model pricing and act as a broker for energy in a smart power grid (Yang et al., 2018). Researchers cluster costumers into groups based on electricity consumption patterns using a K-Means algorithm with Dynamic Time Warping distance criterion. Rather than having a single agent negotiate pricing for each cluster of customers, a recurrent multi-agent cooperative game is created where an agent handles the brokering for one specific cluster. Prices are set based on looking at the past history of customer power usage, and previous electric contracts. The researchers also propose a reward shaping mechanism to prevent bad pricing decisions made by a single agent from being rewarded solely due to good decisions from neighboring agent's decisions. This would potentially confuse the bad actor like in credit assignment problems. The researchers performed an ablation study on their model to determine the importance of the reward shaping mechanism they proposed, and found that without it, the agents and broker failed to make a profit. The researchers also experiment with using a single recurrent DQN instead of their proposed multi-agent

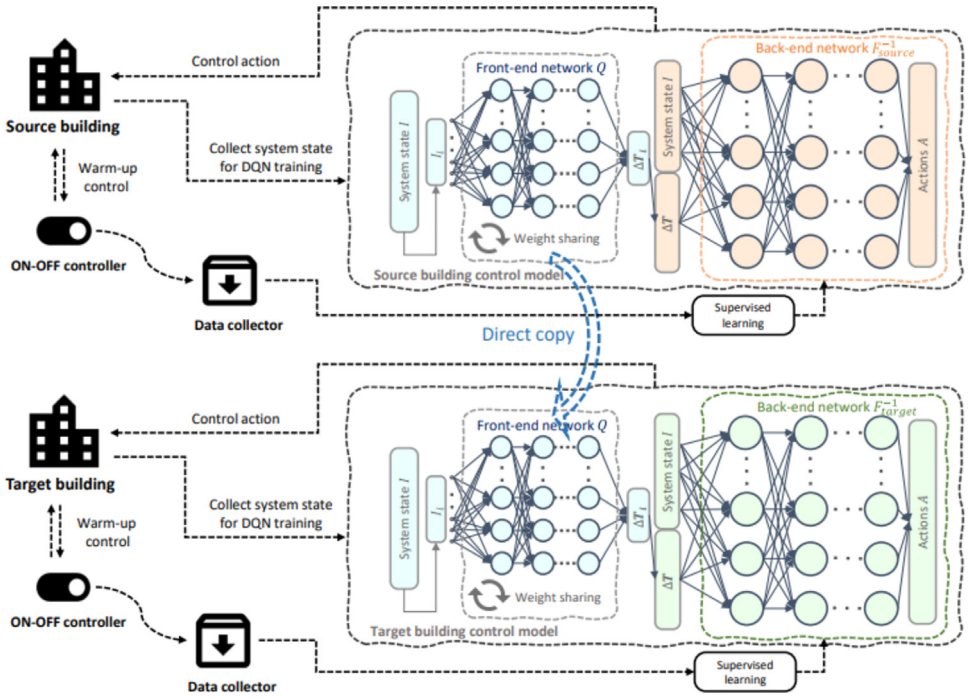


Fig. 7. A front-end network Q captures the building-agnostic control scheme, while a back-end network (inverse building network) F^{-1} captures the building-specific control scheme. The weights are shared directly from the first Q network with the target building's Q network, allowing the target building's F^{-1} network to learn the more complicated building thermal dynamics or more complicated HVAC controller (Xu et al., 2020).

recurrent DQN to show that the multi-agent problem formulation is necessary. Without the multi-agent problem formulation the agent does not converge to a policy that generates profit for the broker.

Another paper that deserves attention is Wei et al. (2020) from 2020. With the rise of smart grids, and automated control within the power industry, researchers have directed their attention to automating the security of these systems, too. A cyber attack on a power grid often takes the form of tripping the transmissions lines to prevent the transfer of power in or out of a region. This team from the University of Rhode Island compares two DRL agents, DQN and DDPG, to determine in the event of an attack which agent better handles the automated reopening of these transmission lines while preventing current inrushes, and power swings. In at least one scenario experimented with DDPG performs better than DQN, but both converge to policies that can mitigate the harms of reclosing transmission lines during a cyber attack on a smart power grid.

4.6. Lessons learned

In the previous Lessons Learned subsection on motor control tasks we were dealing with three different applications that all had a lot of overlap in terms in problem formulation. So it made sense to analyze the similar trends seen across all of the applications together. We believe the tasks revolving around resource allocation are more diverse and actually the different applications seem to teach us different things about DRL. This is why we will go application by application in this section.

As referenced before, traffic control is almost a solved problem, although the initial research is very encouraging (Casas, 2017; Xu et al., 2021). We suggest combining the best aspects of the discussed work. Casas (2017) shows the importance of using DDPG which is better suited for intractable state spaces, Genders and Razavi (2016) shows the importance of using the DTSE state space representation, Van der Pol and Oliehoek (2016) and Xu et al. (2021) show how the usage of multi-agent learning or distributed learning can in some situations improve the results when dealing with such an intractable system such as city wide traffic control, and finally Xu et al. (2021) show the importance of using hierarchical learning to capture neighborhood level statistics for all of their distributed agents.

Vehicle routing, where the resources being allocated are the drivers/vehicles of the taxi fleet, has had a similar evolution as traffic control. Pioneers in the field were not afraid to limit the action or state space, especially in more complicated problems like vehicle routing with ride pooling (Al-Abbasi et al., 2019; Lin et al., 2018; Liu et al., 2020). The complexity of the action space in these cases can be controlled by increasing or decreasing the size of the grids that cars are directed to and user demand is predicted for. Also similar to traffic control, reframing the problem as a multi-agent learning problem amplified the success of previous researchers (Lin et al., 2018).

For resource allocation tasks centered in the telecommunication arena, success is rooted not in how problems are simplified and then built upon in subsequent works, but in how reward functions are crafted with industry expertise. These handcrafted reward functions provide an insight into how the systems work. Equations (8) through (17) and their respective papers all shed light on how state-of-the-art algorithms use industry insights to guide DRL networks through training. Also prevalent in the telecommunication domain is the usage of distributed algorithms (Chen et al., 2018; Li et al., 2018; Liu et al., 2017; Sun et al., 2019) which helps keep the state and action spaces less intractable. We also see the deployment of multi-agent learning in Kaviani et al. (2021), a communication system that successfully decreases packet collision.

Let us recall these lessons from the telecommunication domain and consider the success seen in traffic control and vehicle routing. While traffic in some scenarios can be optimized, there remains a question of fairness. Some researchers (Casas, 2017; Genders & Razavi, 2016) admit that to achieve the results they did required sacrificing fairness. Some lanes of traffic would stay

Table 3
Summary of papers surveyed organized by problems encountered and method of solution.

Problem encountered	Solution	Relevant papers
Lack of training data	Architectural	Das and Won (2021), Rusu et al. (2016a) and Xu et al. (2020)
	Training regimen	Diao et al. (2019), Duan, Shi et al. (2020), Gu et al. (2016), Kalashnikov et al. (2018), Lei et al. (2018), Li et al. (2021), Omisore et al. (2018), Pan et al. (2017), Peng et al. (2018), Pfeiffer et al. (2017), Popov et al. (2017), Ruffy et al. (2018), Rusu et al. (2016a), Tai et al. (2017), Tan et al. (2019), Wang, Velswamy et al. (2017), Wei et al. (2020), Xu et al. (2021, 2020) and Yu et al. (2020)
Reality gap	Architectural	Pan et al. (2017), Rusu et al. (2016a) and Tan et al. (2019)
	Training regimen	Lei et al. (2018), Omisore et al. (2018), Pfeiffer et al. (2017), Rusu et al. (2016a) and Tai et al. (2017)
Intractability of	Architectural	Casas (2017), Das and Won (2021), Duan, Shi et al. (2020, 2020), Honerkamp et al. (2021), Kalashnikov et al. (2018), Li et al. (2016), Lv et al. (2019), Wang, Velswamy et al. (2017), Xu et al. (2021, 2020) and Zhao et al. (2021)
State or action space	Custom loss function	Al-Abbasi et al. (2019), Chinchali et al. (2018), Diao et al. (2019), Peng et al. (2018), Pfeiffer et al. (2017), Placed and Castellanos (2020), Popov et al. (2017), Qureshi et al. (2018), Sun et al. (2019), Xu et al. (2019, 2017), Xu et al. (2021), Yan et al. (2021) and Zhu et al. (2021)
	Training regimen	Al-Abbasi et al. (2019), Cao and Yin (2021), Genders and Razavi (2016), Liu et al. (2017), Omisore et al. (2018), Sallab et al. (2017), Sun et al. (2019), Xu et al. (2020), Zhang et al. (2020) and Zhao et al. (2021)
	Problem simplification	Al-Abbasi et al. (2019), Li et al. (2016) and Sallab et al. (2017)
	Multi-Agent RL	Chen et al. (2020), Kaviani et al. (2021), Lin et al. (2018), Liu et al. (2020), Van der Pol and Oliehoek (2016), Yang et al. (2018) and Yu et al. (2020)
	Hierarchical RL	Chen et al. (2018), Duan, Eben Li et al. (2020), Faust et al. (2017), Li et al. (2021), Liu and Jiang (2018), Xu et al. (2021) and Zhang et al. (2021)

at a red light far longer than would be tolerated by actual human drivers. We urge the readers and DRL researchers working in traffic control and vehicle routing to explore how to balance traditional reward functions with fairness, or customer satisfaction as seen in Xu et al. (2021) which balances these different rewards by using hierarchical learning. This is exemplified by the multi-agent learning vehicle routing paper, that does in fact utilize a custom reward function based on the averaged revenues of drivers directed into a cell to satisfy customer demand (Lin et al., 2018). No ablation study was performed to determine whether the multi-agent learning problem formulation, or the custom reward function proved to be most important, but it is probable that both factor into the successes witnessed in that paper. DeepPool also has a custom reward function (Al-Abbasi et al., 2019).

Power grid management and HVAC control, perhaps non-surprisingly, have been the most successful in applying DRL algorithms within their application domain. It is not surprising because simulations are more effective during training because the reality gap is less prevalent in systems when voltage measurements or temperature sensors are the predominant input to the learning system. As noted before, the typical amount of data required to train RL algorithms has decreased substantially from at least to 50 months in optimal cases, down to a staggering 5 weeks of data (Xu et al., 2020). The novel network architecture proposed in Xu et al. (2020) that uses two networks connected by a latent space is a form of auxiliary learning that should be explored in additional settings. Additionally, the proposed transfer learning of the "target agnostic" network directly into a more complex scenario from a simpler scenario is going to prove paramount in the years to come in alleviating the problems in other domains where data is scarce (Xu et al., 2020). Multi-agent learning has also shown to be successful when applied in these settings (Chen et al., 2020; Yang et al., 2018; Yu et al., 2020). In the future, we hope to see a comparison of these multi-agent and single-agent approaches.

5. Overcoming common problems

5.1. Intractability in state and action space

It is a little ambiguous whether or not a system's intractability stems from interpreting the state space, or stems from choosing an action. Sometimes it is obvious, like when dealing with continuous action spaces (action space intractability) or interpreting road conditions in front of an autonomous vehicle (state space intractability). It is usually not that obvious and in most cases there is cause to say the intractability stems from both the action and state spaces. Regardless of where this intractability stems from we have witnessed that the solutions required to overcome action and state space intractability have a lot in common.

Seminal works in most of these applications start by reducing the state or action space so the problems become less intractable. Once a common framework is found, these constraints are lifted. Both auxiliary learning and handcrafted reward functions, as seen in equations(8) through equations(17), can help guide networks training within more complex systems. In some instances edge computing and distributed algorithms can help shrink the state and action space for learning models. Formulating complex problems into multi-agent learning problems or hierarchical learning problems has lead to success in otherwise intractable problems, and has also lead to increased performance during training and evaluation.

5.2. Lack of real-world training data and the reality gap

Especially in regard to robotics, real-world training and experiments can be costly to create and prohibitively destructive when failures arise. Real-world training data is sparse and difficult to derive. That is why simulators are often used to simplify the training process. Using transfer learning to go from a simpler simulated environment to a more complex environment is one promising technique when data is scarce.

Once researchers make the jump from simulation to real-world they encounter the reality gap problem. Using simulations that share a common feature space with the real-world is a promising technique for overcoming the reality gap problem. We have also discussed architectural design decisions, and training regimens that overcome instances of the reality gap problem.

5.3. Discussion

We have sorted the papers we surveyed based on the problems encountered and solutions used; the results are seen in Table 3. This way researchers who already have an idea of how they want to overcome a problem they are encountering can easily look at a large selection of papers across a variety of domains that all employ similar approaches to solving the common problem encountered. Some papers may appear twice (or even three times) if they are encountering different problems or if they are using multiple solutions.

The future of these applications hinge on real-world adoption and overcoming the reality gap. Real-world adoption will lead to new constraints that do not necessarily come up in simulated experiments. Adoption on any edge device in the field will force algorithms to be more efficient, both in terms of needing faster run-times and needing to be implemented on lower quality hardware compared to dedicated research computers. Some of the works we highlighted in the related reading section deal with optimizations in deep learning for edge device implementation. A decent amount of papers we surveyed have already made the jump to real-world implementation with great success. Many more of the papers surveyed have not made the jump but are well positioned to.

6. Conclusion

This survey explores the work researchers have done in applying deep reinforcement learning algorithms to applied settings, like in motor control tasks, and resource allocation tasks. To our knowledge this is the first survey that seeks to explore these advances in the field at this scope. The state-of-the-art algorithms overwhelmingly rely on hierarchical learning or multi-agent learning for more complicated tasks. Valuable insight can be learned by exploring the auxiliary learning techniques and hand crafted reward functions that come from intimate knowledge of the application domains explored herein. It is our hope that researchers use this survey to become aware of common problems and learn to overcome them with the common solutions found within this survey. It is then and only then can researchers team up together to take on more complex real-world problems.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Agarwal, Mridul, & Aggarwal, Vaneet (2019). Reinforcement learning with non-Markovian rewards. ArXiv: Learning.

Al-Abbasi, Abubakr O., Ghosh, Arnob, & Aggarwal, Vaneet (2019). DeepPool: Distributed model-free algorithm for ride-sharing using deep reinforcement learning. CoRR abs/1903.03882. arXiv:1903.03882. URL: http://arxiv.org/abs/1903.03882.

Andrychowicz, Marcin, Wolski, Filip, Ray, Alex, Schneider, Jonas, Fong, Rachel, Welinder, Peter, et al. (2017). Hindsight experience replay. CoRR abs/1707.01495. arXiv:1707.01495. URL: http://arxiv.org/abs/1707.01495.

Arulkumaran, Kai, Deisenroth, Marc Peter, Brundage, Miles, & Bharath, Anil Anthony (2017). A brief survey of deep reinforcement learning. CoRR abs/1708.05866. arXiv:1708.05866. URL: http://arxiv.org/abs/1708.05866.

Asis, Kristopher De, Hernandez-Garcia, J. Fernando, Holland, G. Zacharias, & Sutton, Richard S. (2018). Multi-step reinforcement learning: A unifying algorithm. arXiv:1703.01327 [cs.AI].

Barcelo, Jaume, Codina, Esteve, Casas, Jordi, Ferrer, J., & García, David (2005). Microscopic traffic simulation: A tool for the design, analysis and evaluation of intelligent transport systems. *Journal of Intelligent and Robotic Systems*, *41*, 173–203. http://dx.doi.org/10.1007/s10846-005-3808-2.

Bellemare, Marc G., Dabney, Will, & Munos, Rémi (2017). A distributional perspective on reinforcement learning. arXiv:1707.06887 [cs.LG].

Bellman, Richard Ernest (1954). The theory of dynamic programming. Santa Monica, CA: RAND Corporation.

Braganza, D., Dawson, D. M., Walker, I. D., & Nath, N. (2007). A neural network controller for continuum robots. *IEEE Transactions on Robotics*, *23*(6), 1270–1277. http://dx.doi.org/10.1109/TRO.2007.906248.

Cao, Liu, & Yin, Hao (2021). Resource allocation for vehicle platooning in 5G NR-V2X via deep reinforcement learning. In *2021 IEEE international black sea conference on communications and networking (BlackSeaCom)* (pp. 1–7). http://dx.doi.org/10.1109/BlackSeaCom52164.2021.9527765.

Casas, Noe (2017). Deep deterministic policy gradient for urban traffic light control. CoRR abs/1703.09035. arXiv:1703.09035. URL: http://arxiv.org/abs/1703.09035.

Chen, Dong, Li, Zhaojian, Chu, Tianshu, Yao, Rui, Qiu, Feng, & Lin, Kaixiang (2020). PowerNet: Multi-agent deep reinforcement learning for scalable powergrid control. arXiv:2011.12354 [eess.SY].

Chen, Li, Lingys, Justinas, Chen, Kai, & Liu, Feng (2018). AuTO: Scaling deep reinforcement learning for datacenter-scale automatic traffic optimization. In *SIGCOMM '18, Proceedings of the 2018 conference of the ACM special interest group on data communication* (pp. 191–205). New York, NY, USA: Association for Computing Machinery, ISBN: 9781450355674, http://dx.doi.org/10.1145/3230543.3230551.

Cheng, Mingxi, Nazarian, Shahin, & Bogdan, Paul (2020). There is hope after all: Quantifying opinion and trustworthiness in neural networks. *Frontiers in Artificial Intelligence*, [ISSN: 2624-8212] *3*, 54. http://dx.doi.org/10.3389/frai.2020.00054, URL: https://www.frontiersin.org/article/10.3389/frai.2020.00054.

Cheng, Mingxi, Yin, Chenzhong, Zhang, Junyao, Nazarian, Shahin, Deshmukh, Jyotirmoy, & Bogdan, Paul (2021). A general trust framework for multi-agent systems. In *AAMAS '21. Virtual event, United Kingdom* (pp. 332–340). Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, ISBN: 9781450383073.

Chinchali, S., Hu, P., Chu, T., Sharma, M., Bansal, M., Misra, R., et al. (2018). Cellular network traffic scheduling with deep reinforcement learning. In *Proc. AAAI conf. on artificial intelligence. New Orleans, Louisiana*. URL:.

Cini, Andrea, D'Eramo, Carlo, Peters, Jan, & Alippi, Cesare (2020). Deep reinforcement learning with weighted Q-learning. arXiv:2003.09280 [cs.LG].

Das, Lokesh, & Won, Myounggyu (2021). SAINT-ACC: Safety-aware intelligent adaptive cruise control for autonomous vehicles using deep reinforcement learning. In *ICML*.

Diao, Ruisheng, Wang, Zhiwei, Shi, Di, Chang, Qianyun, Duan, Jiajun, & Zhang, Xiaohu (2019). Autonomous voltage control for grid operation using deep reinforcement learning. arXiv:1904.10597 [cs.SY].

Dosovitskiy, Alexey, Ros, German, Codevilla, Felipe, Lopez, Antonio, & Koltun, Vladlen (2017). CARLA: An open urban driving simulator. arXiv:1711.03938 [cs.LG].

Duan, Jingliang, Eben Li, Shengbo, Guan, Yang, Sun, Qi, & Cheng, Bo (2020). Hierarchical reinforcement learning for self-driving decision-making without reliance on labelled driving data. *IET Intelligent Transport Systems*, [ISSN: 1751-9578] *14*(5), 297–305. http://dx.doi.org/10.1049/iet-its.2019.0317.

Duan, J., Shi, D., Diao, R., Li, H., Wang, Z., Zhang, B., et al. (2020). Deep-reinforcement-learning-based autonomous voltage control for power grid operations. *IEEE Transactions on Power Systems*, *35*(1), 814–817. http://dx.doi.org/10.1109/TPWRS.2019.2941134.

Espeholt, Lasse, Soyer, Hubert, Munos, Remi, Simonyan, Karen, Mnih, Volodymir, Ward, Tom, et al. (2018). IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. arXiv:1802.01561 [cs.LG].

Fan, Linxi, Zhu, Yuke, Zhu, Jiren, Liu, Zihua, Zeng, Orien, Gupta, Anchit, et al. (2018). SURREAL: Open-source reinforcement learning framework and robot manipulation benchmark. In *Conference on robot learning*.

Faust, Aleksandra, Ramirez, Oscar, Fiser, Marek, Oslund, Kenneth, Francis, Anthony G., Davidson, James, et al. (2017). PRM-RL: long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning. CoRR abs/1710.03937. arXiv:1710.03937. URL: http://arxiv.org/abs/1710.03937.

Fortunato, Meire, Azar, Mohammad Gheshlaghi, Piot, Bilal, Menick, Jacob, Osband, Ian, Graves, Alex, et al. (2019). Noisy networks for exploration. arXiv:1706.10295 [cs.LG].

Gaon, Maor, & Brafman, Ronen (2020). Reinforcement learning with non-Markovian rewards. In *Proceedings of the AAAI conference on artificial intelligence, Vol. 34.04* (pp. 3980–3987). http://dx.doi.org/10.1609/aaai.v34i04.5814, URL: https://ojs.aaai.org/index.php/AAAI/article/view/5814.

Genders, Wade, & Razavi, Saiedeh (2016). Using a deep reinforcement learning agent for traffic signal control. arXiv:1611.01142 [cs.LG].

Gosavi, Abhijit (2009). Reinforcement learning: A tutorial survey and recent advances. *INFORMS Journal on Computing*, *21*, 178–192. http://dx.doi.org/10.1287/ijoc.1080.0305.

Gu, Shixiang, Holly, Ethan, Lillicrap, Timothy, & Levine, Sergey (2016). Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. arXiv:1610.00633 [cs.RO].

Gupta, Gaurav, Yin, Chenzhong, Deshmukh, Jyotirmoy, & Bogdan, Pau (2021). Non-Markovian reinforcement learning using fractional dynamics. In *DeepAI*.

Hafner, Roland, & Riedmiller, Martin (2011). Reinforcement learning in feedback control: Challenges and benchmarks from technical process control. *Machine Learning*, *84*, 137–169. http://dx.doi.org/10.1007/s10994-011-5235-x.

Hasselt, Hado Double Q-learning. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, & A. Culotta (Eds.), *Advances in neural information processing systems*. Curran Associates, Inc..

Hausknecht, Matthew, & Stone, Peter (2017). Deep recurrent Q-learning for partially observable MDPs. arXiv:1507.06527 [cs.LG].

Hayes, Bradley, & Shah, Julie A. (2017). Improving robot controller transparency through autonomous policy explanation. In *HRI '17*, *Proceedings of the 2017 ACM/IEEE international conference on human-robot interaction* (pp. 303–312). New York, NY, USA: Association for Computing Machinery, ISBN: 9781450343367, http://dx.doi.org/10.1145/2909824.3020233.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, & Sun, Jian (2015). Deep residual learning for image recognition. arXiv:1512.03385 [cs.CV].

Hernandez-Leal, Pablo, Kartal, Bilal, & Taylor, Matthew E. (2019). A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, [ISSN: 1573-7454] *33*(6), 750–797. http://dx.doi.org/10.1007/s10458-019-09421-1.

Hessel, Matteo, Modayil, Joseph, van Hasselt, Hado, Schaul, Tom, Ostrovski, Georg, Dabney, Will, et al. (2017a). Rainbow: Combining improvements in deep reinforcement learning. CoRR abs/1710.02298. arXiv:1710.02298. URL: http://arxiv.org/abs/1710.02298.

Hessel, Matteo, Modayil, Joseph, van Hasselt, Hado, Schaul, Tom, Ostrovski, Georg, Dabney, Will, et al. (2017b). Rainbow: Combining improvements in deep reinforcement learning. (p. 1). CoRR abs/1710.02298. Copyright 2018 Association for the Advancement of Artificial Intelligence. arXiv:1710.02298. URL: http://arxiv.org/abs/1710.02298.

Honerkamp, Daniel, Welschehold, Tim, & Valada, Abhinav (2021). Learning kinematic feasibility for mobile manipulation through deep reinforcement learning. http://dx.doi.org/10.48550/ARXIV.2101.05325, URL: https://arxiv.org/abs/2101.05325.

Jaderberg, Max, Mnih, Volodymyr, Czarnecki, Wojciech Marian, Schaul, Tom, Leibo, Joel Z., Silver, David, et al. (2016). Reinforcement learning with unsupervised auxiliary tasks. CoRR abs/1611.05397. arXiv:1611.05397. URL: http://arxiv.org/abs/1611.05397.

Kalashnikov, Dmitry, Irpan, Alex, Pastor, Peter, Ibarz, Julian, Herzog, Alexander, Jang, Eric, et al. (2018). QT-Opt: Scalable deep reinforcement learning for vision-based robotic manipulation. arXiv:1806.10293 [cs.LG].

Kaviani, Saeed, Ryu, Bo, Ahmed, Ejaz, Larson, Kevin A., Le, Anh, Yahja, Alex, et al. (2021). Robust and scalable routing with multi-agent deep reinforcement learning for MANETs. http://dx.doi.org/10.48550/ARXIV.2101.03273, URL: https://arxiv.org/abs/2101.03273.

Ke, Jintao, Zheng, Hongyu, Yang, Hai, & Chen, Xiqun (Michael) (2017a). Short-term forecasting of passenger demand under on-demand ride services: A spatio-temporal deep learning approach. *Transportation Research Part C (Emerging Technologies)*, [ISSN: 0968-090X] *85*, 591–608. http://dx.doi.org/10.1016/j.trc.2017.10.016, URL: https://www.sciencedirect.com/science/article/pii/S0968090X17302899.

Ke, Jintao, Zheng, Hongyu, Yang, Hai, & Chen, Xiqun (Michael) (2017b). Short-term forecasting of passenger demand under on-demand ride services: A spatio-temporal deep learning approach. *Transportation Research Part C (Emerging Technologies)*, [ISSN: 0968-090X] *85*, 602. http://dx.doi.org/10.1016/j.trc.2017.10.016, Copyright 2017 Elsevier. URL: https://www.sciencedirect.com/science/article/pii/S0968090X17302899.

Koenig, N., & Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS) (IEEE cat. no.04CH37566), Vol. 3* (pp. 2149–2154). http://dx.doi.org/10.1109/IROS.2004.1389727, vol.3.

Krajzewicz, Daniel, Hertkorn, Georg, Feld, Christian, & Wagner, Peter (2002). SUMO (simulation of urban mobility); an open-source traffic simulation. (pp. 183–187). ISBN: 90-77039-09-0.

Krishnan, Sanjay (2018). *Hierarchical deep reinforcement learning for robotics and data science* (Ph.D. thesis).

Lei, Xiaoyun, Zhang, Zhian, & Dong, Peifang (2018). Dynamic path planning of unknown environment based on deep reinforcement learning. *Journal of Robotics*, *2018*, 1–10. http://dx.doi.org/10.1155/2018/5781591.

Levine, Sergey, & Koltun, Vladlen (2013). Guided policy search. In *ICML'13*: *Vol. 28*, *Proceedings of the 30th international conference on international conference on machine learning* (pp. III–1–III–9). JMLR.org.

Levine, Sergey, Pastor, Peter, Krizhevsky, Alex, & Quillen, Deirdre (2016). Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. CoRR abs/1603.02199. arXiv:1603.02199. URL: http://arxiv.org/abs/1603.02199.

Levine, Sergey, Wagener, Nolan, & Abbeel, Pieter (2015). Learning contact-rich manipulation skills with guided policy search. arXiv:1501.05611 [cs.RO].

Li, L., Lv, Y., & Wang, F. (2016). Traffic signal timing via deep reinforcement learning. *IEEE/CAA Journal of Automatica Sinica*, *3*(3), 247–254. http://dx.doi.org/10.1109/JAS.2016.7508798.

Li, Jinning, Sun, Liting, Chen, Jianyu, Tomizuka, Masayoshi, & Zhan, Wei (2021). A safe hierarchical planning framework for complex driving scenarios based on reinforcement learning. http://dx.doi.org/10.48550/ARXIV.2101.06778, URL: https://arxiv.org/abs/2101.06778.

Li, Teng, Xu, Zhiyuan, Tang, Jian, & Wang, Yanzhi (2018). Model-free control for distributed stream data processing using deep reinforcement learning. *Proceedings of the VLDB Endowment*, [ISSN: 2150-8097] *11*(6), 705–718. http://dx.doi.org/10.14778/3184470.3184474.

Lillicrap, Timothy P., Hunt, Jonathan J., Pritzel, Alexander, Heess, Nicolas, Erez, Tom, Tassa, Yuval, et al. (2019). Continuous control with deep reinforcement learning. arXiv:1509.02971 [cs.LG].

Lin, Xue, Bogdan, Paul, Chang, Naehyuck, & Pedram, Massoud (2015). Machine learning-based energy management in a hybrid electric vehicle to minimize total operating cost. In *2015 IEEE/ACM international conference on computer-aided design (ICCAD)* (pp. 627–634). http://dx.doi.org/10.1109/ICCAD.2015.7372628.

Lin, Xue, Wang, Yanzhi, Bogdan, Paul, Chang, Naehyuck, & Pedram, Massoud (2014). Reinforcement learning based power management for hybrid electric vehicles. In *2014 IEEE/ACM international conference on computer-aided design (ICCAD)* (pp. 33–38). http://dx.doi.org/10.1109/ICCAD.2014.7001326.

Lin, Kaixiang, Zhao, Renyu, Xu, Zhe, & Zhou, Jiayu (2018). Efficient large-scale fleet management via multi-agent deep reinforcement learning. CoRR abs/1802.06444. arXiv:1802.06444. URL: http://arxiv.org/abs/1802.06444.

Liu, J., Guo, H., Xiong, J., Kato, N., Zhang, J., & Zhang, Y. (2020). Smart and resilient EV charging in SDN-enhanced vehicular edge computing networks. *IEEE Journal on Selected Areas in Communications*, *38*(1), 217–228. http://dx.doi.org/10.1109/JSAC.2019.2951966.

Liu, Daochang, & Jiang, Tingting (2018). Deep reinforcement learning for surgical gesture segmentation and classification. arXiv:1806.08089 [cs.CV].

Liu, Ning, Li, Zhe, Xu, Zhiyuan, Xu, Jielong, Lin, Sheng, Qiu, Qinru, et al. (2017). A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning. arXiv:1703.04221 [cs.DC].

Liu, Xing, Xu, Hansong, Liao, Weixian, & Yu, Wei (2019). Reinforcement learning for cyber-physical systems. In *2019 IEEE international conference on industrial internet (ICII)* (pp. 318–327). http://dx.doi.org/10.1109/ICII.2019.00063.

Lu, Renzhi, & Hong, Seung Ho (2019). Incentive-based demand response for smart grid with reinforcement learning and deep neural network. *Applied Energy*, [ISSN: 0306-2619] *236*, 937–949. http://dx.doi.org/10.1016/j.apenergy.2018.12.061, URL: https://www.sciencedirect.com/science/article/pii/S0306261918318798.

Luh, P. B., Michel, L. D., Friedland, P., Guan, C., & Yuting Wang (2010). Load forecasting and demand response. In *IEEE PES general meeting* (pp. 1–3). http://dx.doi.org/10.1109/PES.2010.5590062.

Lv, L., Zhang, S., Ding, D., & Wang, Y. (2019). Path planning via an improved DQN-based learning policy. *IEEE Access*, *7*, 67319–67330. http://dx.doi.org/10.1109/ACCESS.2019.2918703.

Ma, Xiaolong, Guo, Fu-Ming, Niu, Wei, Lin, Xue, Tang, Jian, Ma, Kaisheng, et al. (2019). PCONV: the missing but desirable sparsity in DNN weight pruning for real-time execution on mobile devices. CoRR abs/1909.05073. arXiv:1909.05073. URL: http://arxiv.org/abs/1909.05073.

Ma, Xiaolong, Niu, Wei, Zhang, Tianyun, Liu, Sijia, Guo, Fu-Ming, Lin, Sheng, et al. (2020). An image enhancing pattern-based sparsity for real-time inference on mobile devices. CoRR abs/2001.07710. arXiv:2001.07710. URL: https://arxiv.org/abs/2001.07710.

Majeed, Sultan Javed, & Hutter, Marcus (2018). On Q-learning convergence for non-Markov decision processes. In *Proceedings of the twenty-seventh international joint conference on artificial intelligence, IJCAI-18* (pp. 2546–2552). International Joint Conferences on Artificial Intelligence Organization, http://dx.doi.org/10.24963/ijcai.2018/353.

Marinescu, A., Harris, C., Dusparic, I., Clarke, S., & Cahill, V. (2013). Residential electrical demand forecasting in very small scale: An evaluation of forecasting methods. In *2013 2nd international workshop on software engineering challenges for the smart grid (SE4SG)* (pp. 25–32). http://dx.doi.org/10.1109/SE4SG.2013.6596108.

Mnih, Volodymyr, Badia, Adrià Puigdomènech, Mirza, Mehdi, Graves, Alex, Lillicrap, Timothy P., Harley, Tim, et al. (2016). Asynchronous methods for deep reinforcement learning. CoRR arXiv:1602.01783. arXiv:1602.01783. URL: http://arxiv.org/abs/1602.01783.

Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Graves, Alex, Antonoglou, Ioannis, Wierstra, Daan, et al. (2013). Playing atari with deep reinforcement learning. CoRR abs/1312.5602. arxiv:1312.5602. URL: http://arxiv.org/abs/1312.5602.

Mocanu, Elena, et al. (2016). Unsupervised energy prediction in a smart grid context using reinforcement cross-building transfer learning. *Energy and Buildings*, [ISSN: 0378-7788] *116*, 646–655. http://dx.doi.org/10.1016/j.enbuild.2016.01.030.

Nair, Ashvin, Chen, Dian, Agrawal, Pulkit, Isola, Phillip, Abbeel, Pieter, Malik, Jitendra, et al. (2017). Combining self-supervised learning and imitation for vision-based rope manipulation. CoRR abs/1703.02018. arXiv:1703.02018. URL: http://arxiv.org/abs/1703.02018.

Obando-Ceron, Johan S., & Castro, Pablo Samuel (2021). Revisiting rainbow: Promoting more insightful and inclusive deep reinforcement learning research. CoRR abs/2011.14826. arXiv:2011.14826. URL: https://arxiv.org/abs/2011.14826.

Omidvar, Omid, & Elliott, David (1997). Neural systems for control.

Omisore, Olatunji Mumini, Han, Shipeng, Ren, Lingxue, Elazab, Ahmed, Hui, Li, Abdelhamid, Talaat, et al. (2018). Deeply-learnt damped least-squares (DL-DLS) method for inverse kinematics of snake-like robots. *Neural Networks*, [ISSN: 0893-6080] *107*, 34–47. http://dx.doi.org/10.1016/j.neunet.2018.06.018, Special issue on deep reinforcement learning. URL: https://www.sciencedirect.com/science/article/pii/S0893608018302181.

Pan, Xinlei, You, Yurong, Wang, Ziyan, & Lu, Cewu (2017). Virtual to real reinforcement learning for autonomous driving. arXiv:1704.03952 [cs.AI].

Patre, P. M., MacKunis, W., Kaiser, K., & Dixon, W. E. (2008). Asymptotic tracking for uncertain dynamic systems via a multilayer neural network feedforward and RISE feedback control structure. *IEEE Transactions on Automatic Control*, *53*(9), 2180–2185. http://dx.doi.org/10.1109/TAC.2008.930200.

Peng, Xue Bin, Andrychowicz, Marcin, Zaremba, Wojciech, & Abbeel, Pieter (2018). Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, ISBN: 9781538630815, http://dx.doi.org/10.1109/icra.2018.8460528.

Pfeiffer, Mark, Schaeuble, Michael, Nieto, Juan I., Siegwart, Roland, & Cadena, Cesar (2017). From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots. CoRR abs/1609.07910. arXiv:1609.07910. URL: http://arxiv.org/abs/1609.07910.

Placed, Julio A., & Castellanos, José A. (2020). A deep reinforcement learning approach for active SLAM. *Applied Sciences*, [ISSN: 2076-3417] *10*(23), http://dx.doi.org/10.3390/app10238386, URL: https://www.mdpi.com/2076-3417/10/23/8386.

Poddar, Shashi, Kottath, Rahul, & Karar, Vinod (2018). Evolution of visual odometry techniques. arXiv:1804.11142 [cs.CV].

Popov, Ivaylo, Heess, Nicolas, Lillicrap, Timothy, Hafner, Roland, Barth-Maron, Gabriel, Vecerik, Matej, et al. (2017). Data-efficient deep reinforcement learning for dexterous manipulation. arXiv:1704.03073 [cs.LG].

Qureshi, Ahmed Hussain, Nakamura, Yutaka, Yoshikawa, Yuichiro, & Ishiguro, Hiroshi (2018). Intrinsically motivated reinforcement learning for human–robot interaction in the real-world. *Neural Networks*, [ISSN: 0893-6080] *107*, 23–33. http://dx.doi.org/10.1016/j.neunet.2018.03.014, Special issue on deep reinforcement learning. URL: https://www.sciencedirect.com/science/article/pii/S0893608018301072.

Rajeswaran, Aravind, Kumar, Vikash, Gupta, Abhishek, Vezzani, Giulia, Schulman, John, Todorov, Emanuel, et al. (2018). Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. arXiv:1709.10087 [cs.LG].

Ren, Ao, Zhang, Tianyun, Ye, Shaokai, Li, Jiayu, Xu, Wenyao, Qian, Xuehai, et al. (2018). ADMM-NN: an algorithm-hardware co-design framework of DNNs using alternating direction method of multipliers. CoRR abs/1812.11677. arXiv:1812.11677. URL: http://arxiv.org/abs/1812.11677.

Riedmiller, Martin (2005). Neural fitted q iteration – first experiences with a data efficient neural reinforcement learning method. In João Gama, Rui Camacho, Pavel B. Brazdil, Alípio Mário Jorge, & Luís Torgo (Eds.), *Machine learning: ECML 2005* (pp. 317–328). Berlin, Heidelberg: Springer Berlin Heidelberg.

Ruffy, Fabian, Przystupa, Michael, & Beschastnikh, Ivan (2018). Iroko: A framework to prototype reinforcement learning for data center traffic control. arXiv:1812.09975 [cs.NI].

Russakovsky, Olga, Deng, Jia, Su, Hao, Krause, Jonathan, Satheesh, Sanjeev, Ma, Sean, et al. (2015). ImageNet large scale visual recognition challenge. arXiv:1409.0575 [cs.CV].

Rusu, Andrei A., Rabinowitz, Neil C., Desjardins, Guillaume, Soyer, Hubert, Kirkpatrick, James, Kavukcuoglu, Koray, et al. (2016). Progressive neural networks. CoRR abs/1606.04671. arXiv:1606.04671. URL: http://arxiv.org/abs/1606.04671.

Rusu, Andrei A., Vecerík, Matej, Rothörl, Thomas, Heess, Nicolas, Pascanu, Razvan, & Hadsell, Raia (2016a). Sim-to-real robot learning from pixels with progressive nets. CoRR abs/1610.04286. arXiv:1610.04286. URL: http://arxiv.org/abs/1610.04286.

Rusu, Andrei A., Vecerík, Matej, Rothörl, Thomas, Heess, Nicolas, Pascanu, Razvan, & Hadsell, Raia (2016b). Sim-to-real robot learning from pixels with progressive nets. (p. 3). CoRR abs/1610.04286. Copyright 2016 DeepMind. arXiv:1610.04286. URL: http://arxiv.org/abs/1610.04286.

Sallab, AhmadEL, Abdou, Mohammed, Perot, Etienne, & Yogamani, Senthil (2017). Deep reinforcement learning framework for autonomous driving. *Electronic Imaging*, [ISSN: 2470-1173] *2017*(19), 70–76. http://dx.doi.org/10.2352/issn.2470-1173.2017.19.avm-023.

Schaul, Tom, Quan, John, Antonoglou, Ioannis, & Silver, David (2016). Prioritized experience replay. arXiv:1511.05952 [cs.LG].

Smith, M., Duncan, G., & Druitt, S. (1995). Paramics: microscopic traffic simulation for congestion management. In *IEE colloquium on dynamic control of strategic inter-urban road networks* (pp. 8/1–8/3). http://dx.doi.org/10.1049/ic:19950249.

Song, Xuan, Kanasugi, Hiroshi, & Shibasaki, Ryosuke (2016). DeepTransport: Prediction and simulation of human mobility and transportation mode at a citywide level. In *IJCAI* (pp. 2618–2624). URL: http://www.ijcai.org/Abstract/16/372.

Sun, Y., Peng, M., & Mao, S. (2019). Deep reinforcement learning-based mode selection and resource management for green fog radio access networks. *IEEE Internet of Things Journal*, *6*(2), 1960–1971. http://dx.doi.org/10.1109/JIOT.2018.2871020.

Sutton, Richard S., Precup, Doina, & Singh, Satinder (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, [ISSN: 0004-3702] *112*(1), 181–211. http://dx.doi.org/10.1016/S0004-3702(99)00052-1, URL: https://www.sciencedirect.com/science/article/pii/S0004370299000521.

Tai, Lei, Paolo, Giuseppe, & Liu, Ming (2017). Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. arXiv:1703.00420 [cs.RO].

Tan, X., Chng, C., Su, Y., Lim, K., & Chui, C. (2019). Robot-assisted training in laparoscopy using deep reinforcement learning. *IEEE Robotics and Automation Letters*, *4*(2), 485–492. http://dx.doi.org/10.1109/LRA.2019.2891311.

Tassa, Yuval, Doron, Yotam, Muldal, Alistair, Erez, Tom, Li, Yazhe, de Las Casas, Diego, et al. (2018). DeepMind control suite. arXiv:1801.00690 [cs.AI].

Todorov, E., Erez, T., & Tassa, Y. (2012). MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems* (pp. 5026–5033). http://dx.doi.org/10.1109/IROS.2012.6386109.

van der Pol, Elise (2016). *Deep reinforcement learning for coordination in traffic light control* (MSc thesis), MA thesis.

Van der Pol, Elise, & Oliehoek, Frans A. (2016). Coordinated deep reinforcement learners for traffic light control. In *NIPS'16 workshop on learning, inference and control of multi-agent systems*. URL: https://sites.google.com/site/malicnips2016/papers.

van Hasselt, Hado, Guez, Arthur, & Silver, David (2015). Deep reinforcement learning with double Q-learning. arXiv:1509.06461 [cs.LG].

Wang, Sen, Clark, Ronald, Wen, Hongkai, & Trigoni, Niki (2017). DeepVO: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, ISBN: 9781509046331, http://dx.doi.org/10.1109/icra.2017.7989236.

Wang, Ziyu, Schaul, Tom, Hessel, Matteo, van Hasselt, Hado, Lanctot, Marc, & de Freitas, Nando (2016). Dueling network architectures for deep reinforcement learning. arXiv:1511.06581 [cs.LG].

Wang, Y., Velswamy, Kirubakaran, & Huang, Biao (2017). A long-short term memory recurrent neural network based reinforcement learning controller for office heating ventilation and air conditioning systems. In *Processes, Vol. 5*. http://dx.doi.org/10.3390/pr5030046.

Wang, Yetang, Ye, Shaokai, He, Zhezhi, Ma, Xiaolong, Zhang, Linfeng, Lin, Sheng, et al. (2019). Non-structured DNN weight pruning – is it beneficial in any platform?

Wei, T., Ren, S., & Zhu, Q. (2019). Deep reinforcement learning for joint datacenter and HVAC load control in distributed mixed-use buildings. *IEEE Transactions on Sustainable Computing*, 1. http://dx.doi.org/10.1109/TSUSC.2019.2910533.

Wei, F., Wan, Z., & He, H. (2020). Cyber-attack recovery strategy for smart grid based on deep reinforcement learning. *IEEE Transactions on Smart Grid*, *11*(3), 2476–2486. http://dx.doi.org/10.1109/TSG.2019.2956161.

Wei, T., Yanzhi Wang, & Zhu, Q. (2017). Deep reinforcement learning for building HVAC control. In *2017 54th ACM EDAC IEEE design automation conference (DAC)* (pp. 1–6). http://dx.doi.org/10.1145/3061639.3062224.

Wen, Shuhuan, Zhao, Yanfang, Yuan, Xiao, Wang, Zongtao, Zhang, Dan, & Manfredi, Luigi (2020). Path planning for active SLAM based on deep reinforcement learning under unknown environments. *Intelligent Service Robotics*, *13*, http://dx.doi.org/10.1007/s11370-019-00310-w.

Xiao, Yao, Nazarian, Shahin, & Bogdan, Paul (2019). Self-optimizing and self-programming computing systems: A combined compiler, complex networks, and machine learning approach. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *27*(6), 1416–1427. http://dx.doi.org/10.1109/TVLSI.2019.2897650.

Xiao, Yao, Nazarian, Shahin, & Bogdan, Paul (2021). Plasticity-on-chip design: Exploiting self-similarity for data communications. *IEEE Transactions on Computers*, *70*(6), 950–962. http://dx.doi.org/10.1109/TC.2021.3071507.

Xu, Z., Tang, J., Yin, C., Wang, Y., & Xue, G. (2019). Experience-driven congestion control: When multi-path TCP meets deep reinforcement learning. *IEEE Journal on Selected Areas in Communications*, *37*(6), 1325–1336. http://dx.doi.org/10.1109/JSAC.2019.2904358.

Xu, Z., Wang, Y., Tang, J., Wang, J., & Gursoy, M. C. (2017). A deep reinforcement learning based framework for power-efficient resource allocation in cloud RANs. In *2017 IEEE international conference on communications (ICC)* (pp. 1–6). http://dx.doi.org/10.1109/ICC.2017.7997286.

Xu, Bingyu, Wang, Yaowei, Wang, Zhaozhi, Jia, Huizhu, & Lu, Zongqing (2021). Hierarchically and cooperatively learning traffic signal control. In *Proceedings of the AAAI conference on artificial intelligence, Vol. 35.1* (pp. 669–677). URL: https://ojs.aaai.org/index.php/AAAI/article/view/16147.

Xu, Shichao, Wang, Yixuan, Wang, Yanzhi, O'Neill, Zheng, & Zhu, Qi (2020). One for many. In *Proceedings of the 7th ACM international conference on systems for energy-efficient buildings, cities, and transportation*. ACM, ISBN: 9781450380614, http://dx.doi.org/10.1145/3408308.3427617.

Yahya, Ali, Li, Adrian, Kalakrishnan, Mrinal, Chebotar, Yevgen, & Levine, Sergey (2017). Collective robot reinforcement learning with distributed asynchronous guided policy search. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, ISBN: 9781538626825, http://dx.doi.org/10.1109/iros.2017.8202141.

Yan, Chao, Xiang, Xiaojia, Wang, Chang, & Lan, Zhen (2021). Flocking and collision avoidance for a dynamic squad of fixed-wing uavs using deep reinforcement learning. In *2021 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 4738–4744). http://dx.doi.org/10.1109/IROS51168.2021.9636183.

Yang, Yaodong, Hao, Jianye, Sun, Mingyang, Wang, Zan, Fan, Changjie, & Strbac, Goran (2018). Recurrent deep multiagent Q-learning for autonomous brokers in smart grid. In *Proceedings of the twenty-seventh international joint conference on artificial intelligence, IJCAI-18* (pp. 569–575). International Joint Conferences on Artificial Intelligence Organization, http://dx.doi.org/10.24963/ijcai.2018/79.

Yu, Liang, Sun, Yi, Xu, Zhanbo, Shen, Chao, Yue, Dong, Jiang, Tao, et al. (2020). Multi-agent deep reinforcement learning for HVAC control in commercial buildings. arXiv:2006.14156 [eess.SY].

Yu, L., Yu, X., Chen, X., & Zhang, F. (2019). Laparoscope arm automatic positioning for robot-assisted surgery based on reinforcement learning. *Mechanical Sciences*, *10*(1), 119–131. http://dx.doi.org/10.5194/ms-10-119-2019, URL: https://ms.copernicus.org/articles/10/119/2019/.

Zhang, Huichu, Feng, Siyuan, Liu, Chang, Ding, Yaoyao, Zhu, Yichen, Zhou, Zihan, et al. (2019). CityFlow: A multi-agent reinforcement learning environment for large scale city traffic scenario. In *The world wide web conference*. ACM, http://dx.doi.org/10.1145/3308558.3314139.

Zhang, Jingwei, Tai, Lei, Liu, Ming, Boedecker, Joschka, & Burgard, Wolfram (2020). Neural SLAM: Learning to explore with external memory. arXiv:1706.09520 [cs.LG].

Zhang, Dawei, Zheng, Zhonglong, Jia, Riheng, & Li, Minglu (2021). Visual tracking via hierarchical deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence, Vol. 35.4* (pp. 3315–3323). URL: https://ojs.aaai.org/index.php/AAAI/article/view/16443.

Zhao, Hang, She, Qijin, Zhu, Chenyang, Yang, Yin, & Xu, Kai (2021). Online 3D bin packing with constrained deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence, Vol. 35.1* (pp. 741–749). URL: https://ojs.aaai.org/index.php/AAAI/article/view/16155.

Zhu, Zhaoxuan, Gupta, Shobhit, Gupta, Abhishek, & Canova, Marcello (2021). A deep reinforcement learning framework for eco-driving in connected and automated hybrid electric vehicles. http://dx.doi.org/10.48550/ARXIV.2101.05372, URL: https://arxiv.org/abs/2101.05372.