DEEP REINFORCEMENT LEARNING: AN OVERVIEW

Yuxi Li (yuxili@gmail.com)

ABSTRACT

We give an overview of recent exciting achievements of deep reinforcement learning (RL). We start with background of deep learning and reinforcement learning, as well as introduction of testbeds. Next we discuss Deep Q-Network (DQN) and its extensions, asynchronous methods, policy optimization, reward, and planning. After that, we talk about attention and memory, unsupervised learning, and learning to learn. Then we discuss various applications of RL, including games, in particular, AlphaGo, robotics, spoken dialogue systems (a.k.a. chatbot), machine translation, text sequence prediction, neural architecture design, personalized web services, healthcare, finance, and music generation. We mention topics/papers not reviewed yet. After listing a collection of RL resources, we close with discussions.

1 INTRODUCTION

Reinforcement learning (RL) is usually about sequential decision making, solving problems in a wide range of fields in science, engineering and arts (Sutton and Barto, 2017).

The integration of reinforcement learning and neural networks dated back to 1990s (Tesauro, 1994; Bertsekas and Tsitsiklis, 1996; Schmidhuber, 2015). With recent exciting achievements of deep learning (LeCun et al., 2015; Goodfellow et al., 2016), benefiting from big data, powerful computation and new algorithmic techniques, we have been witnessing the renaissance of reinforcement learning (Krakovsky, 2016), especially, the combination of reinforcement learning and deep neural networks, i.e., deep reinforcement learning (deep RL).

We have been witnessing breakthroughs, like deep Q-network (Mnih et al., 2015), AlphaGo (Silver et al., 2016) and differentiable neural computer (Graves et al., 2016); and novel architectures and applications, like asynchronous methods (Mnih et al., 2016), dueling network architectures (Wang et al., 2016a), value iteration networks (Tamar et al., 2016), dual learning for machine translation (He et al., 2016a), spoken dialogue systems (Su et al., 2016b), information extraction (Narasimhan et al., 2016), guided policy search (Levine et al., 2016a), generative adversarial imitation learning (Ho and Ermon, 2016), unsupervised reinforcement and auxiliary learning (Jaderberg et al., 2017), and neural architecture design (Zoph and Le, 2017), etc. In this overview, we mainly focus on work in recent couple of years, and by no means complete.[1]

We refer readers to materials for further study: reinforcement learning (Sutton and Barto, 2017; Szepesvári, 2010; Bertsekas, 2012; Powell, 2011; Bertsekas and Tsitsiklis, 1996; Puterman, 2005; Littman, 2015; Kaelbling et al., 1996); deep learning (LeCun et al., 2015; Goodfellow et al., 2016; Bengio, 2009; Deng and Dong, 2014); machine learning (Jordan and Mitchell, 2015; Hastie et al., 2009; Bishop, 2011; Murphy, 2012; James et al., 2013); practical machine learning advices (Domingos, 2012; Zinkevich, 2017); artificial intelligence (Russell and Norvig, 2009), deep learning in neural networks (Schmidhuber, 2015); natural language processing (NLP) (Hirschberg and Manning, 2015; Deng and Liu, 2017); robotics (Kober et al., 2013); transfer learning (Taylor and Stone, 2009; Pan and Yang, 2010; Weiss et al., 2016); semi-supervised learning (Zhu and Goldberg, 2009); Bayesian RL (Ghavamzadeh et al., 2015); spoken dialogue systems (Hinton et al., 2012; He and Deng, 2013; Young et al., 2013); AI safety (Amodei et al., 2016; Garcìa and Fernàndez, 2015), Monte Carlo tree search (MCTS) (Browne et al., 2012;

[1] We consider this overview as incomplete, for time and timing reasons, in the sense that we may not discuss in depth all relevant work, and we will see fast and enormous growth in this field in the next couple of years. Yet we decide to make this overview public available, hoping it would be helpful for some people in the community and we would appreciate feedbacks for us to make improvements.

Gelly et al., 2012); multi-agent RL (Shoham et al., 2003; Busoniu et al., 2008); game theory (Leyton-Brown and Shoham, 2008), etc. We list RL resources in Section 23. See lists of RL applications at: goo.gl/KoXIQC, and goo.gl/1Q1lzg.

The outline of this overview follows: background of deep learning and reinforcement learning, as well as introduction of testbeds in Section 2; Deep Q-Network (DQN) and its extensions in Section 3; asynchronous methods in Section 4; policy optimization in Section 5; reward in Section 6; planning in Section 7; attention and memory, in particular differentiable neural computer (DNC), in Section 8; unsupervised learning in Section 9; learning to learn in Section 10; games, including board games, video games and imperfect information games, in Section 11; AlphaGo in Section 12; robotics in Section 13; spoken dialogue systems (a.k.a. chatbot) in Section 14; machine translation in Section 15; text sequence prediction in Section 16; neural architecture design in Section 17; personalized web services in Section 18; healthcare in Section 19; finance in Section 20; music generation in Section 21; a to-do list of topics/papers not reviewed yet in Section 22; and discussions in Section 24.[2]

In particular, we list a collection of RL resources including books, online courses, tutorials, conferences, journals and workshops, and blogs in Section 23. If picking a single RL resource, it is Professor Sutton's RL book (Sutton and Barto, 2017), 2nd edition in progress. It covers RL fundamentals and reflects new progress, e.g., in deep Q-network, AlphaGo, policy gradient methods, as well as in psychology and neuroscience. A single pick for deep learning is Goodfellow et al. (2016).

2 BACKGROUND

In this section, we briefly introduce concepts and fundamentals in deep learning (Goodfellow et al., 2016) and reinforcement learning (Sutton and Barto, 2017).

2.1 DEEP LEARNING

Deep learning is in contrast to "shallow" learning. For many machine learning algorithms, e.g., linear regression, logistic regression, support vector machines (SVMs), decision trees, and boosting, we have input layer and output layer, and the inputs may be transformed with manual feature engineering before training. In deep learning, between input and output layers, we have one or more hidden layers. At each layer except input layer, we compute the input to each unit, as the weighted sum of units from the previous layer; then we usually use nonlinear transformation, or activation function, such as logistic, tanh, or more popular recently, rectified linear unit (ReLU), to apply to the input of a unit, to obtain a new representation of the input from previous layer. We have weights on links between units from layer to layer. After computations flow forward from input to output, at output layer and each hidden layer, we can compute error derivatives backward, and backpropagate gradients towards the input layer, so that weights can be updated to optimize some loss function.

A feedforward deep neural network or multilayer perceptron (MLP) is to map a set of input values to output values with a mathematical function formed by composing many simpler functions at each layer. A convolutional neural network (CNN) is a feedforward deep network, with convolutional layers, pooling layers and fully connected layers. CNNs are designed to process data with multiple arrays, e.g., colour image, language, audio spectrogram, and video, benefit from the properties of such signals: local connections, shared weights, pooling and the use of many layers, and are inspired by simple cells and complex cells in visual neuroscience (LeCun et al., 2015). A recurrent neural network (RNN) is often used to process sequential inputs like speech and language, element

[2]We discuss how/why we organize the overview from Section 3 to Section 21 in the current way: starting with RL fundamentals: value function/control, policy, reward, and planning (model in to-do list); next attention and memory, unsupervised learning, and learning to learn, which, together with transfer/semi-supervised/one-shot learning, etc, would be critical mechanisms for RL; then various applications.

We basically make a flat organization of topics. Otherwise, there may be multiple ways to categorize the topics reviewed. For example, we can combine spoken dialogue systems, machine translation and text sequence prediction as a single section about language models. Another way is to combine these topics, together with learning to learn, neural architecture design and music generation as a section about sequence modelling. Dueling architecture, Value Iteration Netwroks, and differentiable neural computer (DNC) are novel neural networks architectures for RL.

by element, with hidden units to store history of past elements. A RNN can be seen as a multilayer network with all layers sharing the same weights, when being unfolded in time of forward computation. It is hard for RNN to store information for very long time and the gradient may vanish. Long short term memory networks (LSTM) and gated recurrent unit (GRU) were proposed to address such issues, with gating mechanisms to manipulate information through recurrent cells. Gradient backpropagation or its variants can be used for training all above deep neural networks.

Dropout is a regularization strategy to train an ensemble of sub-networks by removing non-output units randomly from the original network. Batch normalization performs the normalization for each training mini-batch, to accelerate training by reducing internal covariate shift, i.e., the change of parameters of previous layers will change each layer's inputs distribution.

Deep neural networks learn representations automatically from raw inputs to recover the compositional hierarchies in many natural signals, i.e., higher-level features are composed of lower-level ones, e.g., in images, the hierarch of objects, parts, motifs, and local combinations of edges. Distributed representation is a central idea in deep learning, which implies that many features may represent each input, and each feature may represent many inputs. The exponential advantages of deep, distributed representations combat the exponential challenges of the curse of dimensionality. The notion of end-to-end training refers to that a learning model uses raw inputs without manual feature engineering to generate outputs, e.g., AlexNet (Krizhevsky et al., 2012) with raw pixels for image classification, Seq2Seq (Sutskever et al., 2014) with raw sentences for machine translation, and DQN (Mnih et al., 2015) with raw pixels and score to play games.

2.2 Reinforcement Learning

Reinforcement learning usually solves sequential decision making problems. An RL agent interacts with an environment over time. At each time step t, the agent receives a state s_t and selects an action a_t from some action space $\mathcal{A}$, following a policy $\pi(a_t|s_t)$, which is the agent's behavior, i.e., a mapping from state s_t to actions a_t, receives a scalar reward r_t, and transitions to the next state s_{t+1}, according to the environment dynamics, or model, for reward function $R(s, a)$ and state transition probability $P(s_{t+1}|s_t, a_t)$ respectively. In an episodic problem, this process continues until the agent reaches a terminal state and then it restarts. The return $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ is the discounted, accumulated reward with the discount factor $\gamma \in (0, 1]$. The agent aims to maximize the expectation of such long term return from each state.

A value function is a prediction of the expected, accumulative, discounted, future reward, measuring how good is each state, or state-action pair. The action value $Q^\pi(s, a) = E[R_t|s_t = s, a_t = a]$ is the expected return for selecting action a in state s and then following policy π. An optimal action value function $Q^*(s, a)$ is the maximum action value achievable by any policy for state s and action a. We can define state value $V^\pi(s)$ and optimal state value $V^*(s)$ similarly.

Temporal difference (TD) learning is a central idea in RL. It learns value function $V(s)$ directly from experience with TD error, with bootstrapping, in a model-free, online, and fully incremental way. The update rule is $V(s_t) \leftarrow V(s_t) + \alpha[r_t + \gamma V(s_{t+1}) - V(s_t)]$, where α is a learning rate, and $r_t + \gamma V(s_{t+1}) - V(s_t)$ is called TD error. Similarly, Q-learning learns action value function, with the update rule, $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$. Q-learning is an off-policy control method. In contrast, SARSA, representing state, action, reward, (next) state, (next) action, is an on-policy control method, with the update rule, $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$. SARSA refines the policy greedily with respect to action values. TD-learning, Q-learning and SARSA converge under certain conditions. From optimal action value function, we can derive an optimal policy.

The above algorithms are referred to as TD(0) and Q(0), with one-step return. We have multi-step return variants or Monte-Carlo approach in the forward view. The eligibility trace from the backward view provides an online, incremental implementation, resulting in TD(λ) and Q(λ) algorithms, where $\lambda \in [0, 1]$. When $\lambda = 1$, it is the same as a Monte Carlo approach.

We discuss the tabular cases above, where a value function or a policy is stored in a tabular form. Function approximation is a way for generalization when the state and/or action spaces are large or continuous. Linear function approximation used to be a popular choice, esp. before the work of Deep Q-Network (Mnih et al., 2015).

In contrast to value-based methods like TD learning and Q-learning, policy-based methods optimize the policy $\pi(a|s;\theta)$ (with function approximation) directly, and update the parameters θ by gradient ascent on $E[R_t]$. REINFORCE is a policy gradient method, updating θ in the direction of $\nabla_\theta \log \pi(a_t|s_t;\theta)R_t$. Usually a baseline $b_t(s_t)$ is subtracted from the return to reduce the variance of gradient estimate, yet keeping its unbiasedness, to yield the gradient direction $\nabla_\theta \log \pi(a_t|s_t;\theta)(R_t - b_t(s_t))$. Using $V(s_t)$ as the baseline $b_t(s_t)$, we have the advantage function $A(a_t, s_t) = Q(a_t, s_t) - V(s_t)$, since R_t is an estimate of $Q(a_t, s_t)$. In actor-critic algorithms, the critic updates action-value function parameters, and the actor updates policy parameters, in the direction suggested by the critic.

We obtain deep reinforcement learning (deep RL) methods when we use deep neural networks to approximate any of the following component of reinforcement learning: value function, $V(s;\theta)$ or $Q(s, a;\theta)$, policy $\pi(a|s;\theta)$, and model (state transition and reward). Here, the parameters θ are the weights in deep neural networks. When we use "shallow" models, like linear function, decision trees, tile coding and so on as the function approximator, we obtain "shallow" RL, and the parameters θ are the weight parameters in these models. Note, a shallow model, e.g., decision trees, may be non-linear. The distinct difference between deep RL and "shallow" RL is what function approximator is used. This is similar to the difference between deep learning and "shallow" learning. We usually utilize stochastic gradient descent to update weight parameters in deep RL. When off-policy, function approximation, in particular, non-linear function approximation, and bootstrapping are combined together, instability and divergence may occur (Tsitsiklis and Van Roy, 1997). However, recent work like Deep Q-Network (Mnih et al., 2015) and AlphaGo (Silver et al., 2016) stabilized the learning and achieved outstanding results.

We explain some terms in RL parlance. The prediction problem, or policy evaluation, is to compute the state or action value function for a policy. The control problem is to find the optimal policy. Planning constructs a value function or a policy with a model. On-policy methods evaluate or improve the behavioural policy, e.g., SARSA fits the action-value function to the current policy, i.e., SARSA evaluates the policy based on samples from the same policy, then refines the policy greedily with respect to action values. In off-policy methods, an agent learns an optimal value function/policy, maybe following an unrelated behavioural policy, e.g., Q-learning attempts to find action values for the optimal policy directly, not necessarily fitting to the policy generating the data, i.e., the policy Q-learning obtains is usually different from the policy that generates the samples. The notion of on-policy and off-policy can be understood as same-policy and different-policy.The exploration-exploitation dilemma is about the agent needs to exploit the currently best action to obtain rewards, yet it has to explore the environment to find better actions. In model-free methods, the agent learns with trail-and-error from experience explicitly; the model (state transition function) is not known or learned from experience. RL methods that use models are model-based methods. In online mode, training algorithms are executed on data acquired in sequence. In batch mode, models are trained on the entire data set. With bootstrapping, an estimate of state or action value is updated from subsequent estimates.

2.3 TESTBEDS

The Arcade Learning Environment (ALE) (Bellemare et al., 2013) is a framework composed of Atari 2600 games to develop and evaluate AI agents.

DeepMind released a first-person 3D game platform DeepMind Lab (Beattie et al., 2016). Deepmind and Blizzard will collaborate to release the Starcraft II AI research environment (goo.gl/Ptiwfg).

OpenAI Gym (https://gym.openai.com) is a toolkit for the development of RL algorithms, consisting of environments, e.g., Atari games and simulated robots, and a site for the comparison and reproduction of results.

OpenAI Universe (https://universe.openai.com) is used to turn any program into a Gym environment. Universe has already integrated many environments, including Atari games, flash games, browser tasks like Mini World of Bits and real-world browser tasks. Recently, GTA V was added to Universe for self-driving vehicle simulation.

FAIR TorchCraft (Synnaeve et al., 2016) is a library for Real-Time Strategy (RTS) games such as StarCraft: Brood War.

ViZDoom is a Doom-based AI research platform for visual RL (Kempka et al., 2016).

TORCS is a car racing simulator (Bernhard Wymann et al., 2014).

MuJoCo, Multi-Joint dynamics with Contact, is a physics engine. See http://www.mujoco.org.

Duan et al. (2016) presented a benchmark for continuous control tasks. The open source is available at: https://github.com/openai/rllab.

Nogueira and Cho (2016) presented WebNav Challenge for Wikipedia links navigation.

3 DEEP Q-NETWORK

Mnih et al. (2015) introduced Deep Q-Network (DQN) and ignited the field of deep RL. Before DQN, it is well known that RL is unstable or even divergent when action-value Q function is approximated with a nonlinear function like neural networks. DQN made several important contributions: 1) stabilize the training of Q action value function approximation with deep neural networks (CNN) using experience replay (Lin, 1992) and target network; 2) designing an end-to-end RL approach, with only the pixels and the game score as inputs, so that only minimal domain knowledge is required; 3) training a flexible network with the same algorithm, network architecture and hyperparameters to perform well on many different tasks, i.e., 49 Atari games (Bellemare et al., 2013), and outperforming previous algorithms and performing comparably to a human professional tester.

See Chapter 16 in Sutton and Barto (2017) for a Sutton-style description of Deep Q-Network. See Deepmind's description of DQN at goo.gl/IWco9h. We present DQN pseudo code below.

Input: the pixels and the game score
Output: Q action value function (from which we obtain policy and select action)
Initialize replay memory D
Initialize action-value function Q with random weight θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$
for *episode = 1 to* M **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  for *t = 1 to* T **do**
    Following ϵ-greedy policy, select $a_t = \begin{cases} \text{a random action} & \text{with probability } \epsilon \\ \arg\max_a Q(\phi(s_t), a; \theta) & \text{otherwise} \end{cases}$
    Execute action a_i in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    `// experience replay`
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ w.r.t. the network parameter θ
    `// periodic update of target network`
    Every C steps reset $\hat{Q} = Q$, i.e., set $\theta^- = \theta$
  end
end

Algorithm 1: Deep Q-Nework (DQN), adapted from Mnih et al. (2015)

3.1 DOUBLE DQN

van Hasselt et al. (2016a) proposed Double DQN (D-DQN) to tackle the overestimate problem in Q-learning. In standard Q-learning, as well as in DQN, the parameters are updated as follows:

$$\theta_{t+1} = \theta_t + \alpha(y_t^Q - Q(s_t, a_t; \theta_t))\nabla_{\theta_t} Q(s_t, a_t; \theta_t),$$

where

$$y_t^Q = r_{t+1} + \gamma \max_a Q(s_{t+1}, a; \theta_t),$$

so that the max operator uses the same values to both select and evaluate an action. As a consequence, it is more likely to select overestimated values, and results in overoptimistic value estimates. van Hasselt et al. (2016a) proposed to evaluate the greedy policy according to the online network, but to use the target network to estimate its value. This can be achieved with a minor change to the DQN algorithm, replacing y_t^Q with

$$y_t^{D-DQN} = r_{t+1} + \gamma Q(s_{t+1}, \max_a Q(s_{t+1}, a_t; \theta_t); \theta_t^-),$$

where θ_t is the parameter for online network and θ_t^- is the parameter for target network. For reference, y_t^Q can be written as

$$y_t^Q = r_{t+1} + \gamma Q(s_{t+1}, \max_a Q(s_{t+1}, a_t; \theta_t); \theta_t).$$

D-DQN found better policies than DQN on Atari games.

3.2 Prioritized Experience Replay

In DQN, experience transitions are uniformly sampled from the replay memory, regardless of the significance of experiences. Schaul et al. (2016) proposed to prioritize experience replay, so that important experience transitions can be replayed more frequently, to learn more efficiently. The importance of experience transitions are measured by TD errors. The authors designed a stochastic prioritization based on the TD errors, using importance sampling to avoid the bias in the update distribution. The authors used prioritized experience replay in DQN and D-DQN, and improved their performance on Atari games.

3.3 Dueling Architecture

Wang et al. (2016b) proposed the dueling network architecture to estimate state value function $V(s)$ and associated advantage function $A(s, a)$, and then combine them to estimate action value function $Q(s, a)$, to converge faster than Q-learning. In DQN, a CNN layer is followed by a fully connected (FC) layer. In dueling architecture, a CNN layer is followed by two streams of FC layers, to estimate value function and advantage function separately; then the two streams are combined to estimate action function. Usually we use the following to combine $V(s)$ and $A(s, a)$ to obtain $Q(s, a)$,

$$Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + \big(A(s, a; \theta, \alpha) - \max_{a'} A(s, a'; \theta, \alpha)\big)$$

where α and β are parameters of the two streams of FC layers. Wang et al. (2016b) proposed to replace max operator with average as following for better stability,

$$Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + \big(A(s, a; \theta, \alpha) - \frac{a}{|\mathcal{A}|} A(s, a'; \theta, \alpha)\big)$$

Dueling architecture implemented with D-DQN and prioritized experience replay improved previous work, DQN and D-DQN with prioritized experience replay, on Atari games.

3.4 More Extensions

Mnih et al. (2016) proposed asynchronous methods for RL methods, in particular, the asynchronous advantage actor-critic (A3C) algorithm, as discussed in Section 4. Osband et al. (2016) designed better exploration strategy to improve DQN. O'Donoghue et al. (2017) proposed policy gradient and Q-learning (PGQ), as discussed in Section 5.6. He et al. (2017) proposed to accelerate DQN by optimality tightening, a constrained optimization approach, to propagate reward faster, and to improve accuracy over DQN. Babaeizadeh et al. (2017) proposed a hybrid CPU/GPU implementation of A3C. Liang et al. (2016) attempted to understand the success of DQN and reproduced results with shallow RL.

4 ASYNCHRONOUS METHODS

Mnih et al. (2016) proposed asynchronous methods for four RL methods, Q-learning, SARSA, n-step Q-learning and advantage actor-critic, and the asynchronous advantage actor-critic (A3C) algorithm performs the best. Parallel actors employ different exploration policies to stabilize training, so that experience replay is not utilized. Different from most deep learning algorithms, asynchronous methods can run on a single multi-core CPU. For Atari games, A3C ran much faster yet performed better than or comparably with DQN, Gorila, D-DQN, Dueling D-DQN, and Prioritized D-DQN. A3C also succeeded on continuous motor control problems: TORCS car racing games and MujoCo physics manipulation and locomotion, and Labyrinth, a navigating task in random 3D mazes using visual inputs, in which an agent will face a new maze in each new episode, so that it needs to learn a general strategy to explore random mazes.

We present pseudo code for asynchronous advantage actor-critic for each actor-learner thread. A3C maintains a policy $\pi(a_t|s_t;\theta)$ and an estimate of the value function $V(s_t;\theta_v)$, being updated with n-step returns in the forward view, after every t_{max} actions or reaching a terminal state, similar to using minibatches. The gradient update can be seen as $\nabla_{\theta'} \log \pi(a_t|s_t;\theta')A(s_t,a_t;\theta,\theta_v)$, where $A(s_t,a_t;\theta,\theta_v) = \sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k V(s_{t+k};\theta_v) - V(s_t;\theta_v)$ is an estimate of the advantage function, with k upbounded by t_{max}.

Global shared parameter vectors θ and θ_v, thread-specific parameter vectors θ' and θ'_v
Global shared counter $T = 0$, T_{max}
Initialize step counter $t \leftarrow 1$
for $T \leq T_{max}$ **do**
    Reset gradients, $d\theta \leftarrow 0$ and $d\theta_v \leftarrow 0$
    Synchronize thread-specific parameters $\theta' = \theta$ and $\theta'_v = \theta_v$
    Set $t_{start} = t$, get state s_t
    for s_t *not terminal and* $t - t_{start} \leq t_{max}$ **do**
        Take a_t according to policy $\pi(a_t|s_t;\theta')$
        Receive reward r_t and new state s_{t+1}
        $t \leftarrow t+1$, $T \leftarrow T+1$
    end
    $R = \begin{cases} 0 & \text{for terminal } s_t \\ V(s_t,\theta'_v) & \text{otherwise} \end{cases}$
    for $i \in \{t-1,...,t_{start}\}$ **do**
        $R \leftarrow r_i + \gamma R$
        accumulate gradients wrt θ': $d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi(a_i|s_i;\theta')(R - V(s_i;\theta'_v))$
        accumulate gradients wrt θ'_v: $d\theta_v \leftarrow d\theta_v + \nabla_{\theta'_v}(R - V(s_i;\theta'_v))^2$
    end
    Update asynchronously θ using $d\theta$, and θ_v using $d\theta_v$
end

Algorithm 2: A3C, each actor-learner thread, based on Mnih et al. (2016)

5 POLICY OPTIMIZATION

Policies are usually stochastic. However, Silver et al. (2014) introduced the deterministic policy gradient (DPG) for efficient estimation of policy gradient. Lillicrap et al. (2016) extended DPG with deep neural networks. We also introduce several recent work, including Guided Policy Search (Levine et al., 2016a), Trust Region Policy Optimization (Schulman et al., 2015), benchmark results (Duan et al., 2016) and policy gradient and Q-learning (O'Donoghue et al., 2017).

5.1 DETERMINISTIC POLICY GRADIENT

Silver et al. (2014) introduced the deterministic policy gradient (DPG) algorithm for RL problems with continuous action spaces. The deterministic policy gradient is the expected gradient of the action-value function, which integrates over the state space; whereas in the stochastic case, the policy gradient integrates over both state and action spaces. Consequently, the deterministic policy gradient can be estimated more efficiently than the stochastic policy gradient. The authors intro-

duced an off-policy actor-critic algorithm to learn a deterministic target policy from an exploratory behaviour policy, and to ensure unbiased policy gradient with the compatible function approximation for deterministic policy gradients. Empirical results showed its superior to stochastic policy gradients, in particular in high dimensional tasks, on several problems: a high-dimensional bandit; standard benchmark RL tasks of mountain car and pendulum and 2D puddle world with low dimensional action spaces; and controlling an octopus arm with a high-dimensional action space. The experiments were conducted with tile-coding and linear function approximators.

5.2 Deep Deterministic Policy Gradient

Lillicrap et al. (2016) proposed an actor-critic, model-free, deep deterministic policy gradient (DDPG) algorithm in continuous action spaces, by extending DQN (Mnih et al., 2015) and DPG (Silver et al., 2014). With actor-critic as in DPG, DDPG avoids the optimization of action at every time step to obtain a greedy policy as in Q-learning, which will make it infeasible in complex action spaces with large, unconstrained function approximators like deep neural networks. To make the learning stable and robust, similar to DQN, DDPQ deploy experience replay and an idea similar to target network, "soft" target, which, rather than copying the weights directly as in DQN, updates the soft target network weights θ' slowly to track the learned networks weights θ: $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$, with $\tau \ll 1$. The authors adapted batch normalization to handle the issue that the different components of the observation with different physical units. As an off-policy algorithm, DDPG learns an actor policy from experiences from an exploration policy by adding noise sampled from a noise process to the actor policy. More than 20 simulated physics tasks of varying difficulty in the MuJoCo environment were solved with the same learning algorithm, network architecture and hyper-parameters, and obtained policies with performance competitive with those found by a planning algorithm with full access to the underlying physical model and its derivatives. DDPG can solve problems with 20 times fewer steps of experience than DQN, although it still needs a large number of training episodes to find solutions, as in most model-free RL methods. It is end-to-end, with raw pixels as input. DDPQ paper also contains links to videos for illustration.

5.3 Guided Policy Search

Levine et al. (2016a) proposed to train the perception and control systems jointly end-to-end, to map raw image observations directly to torques at the robot's motors. The authors introduced guided policy search (GPS) to train policies represented as CNN, by transforming policy search into supervised learning to achieve data efficiency, with training data provided by a trajectory-centric RL method operating under unknown dynamics. GPS alternates between trajectory-centric RL and supervised learning, to obtain the training data coming from the policy's own state distribution, to address the issue that supervised learning usually does not achieve good, long-horizon performance. GPS utilizes pre-training to reduce the amount of experience data to train visuomotor policies. Good performance was achieved on a range of real-world manipulation tasks requiring localization, visual tracking, and handling complex contact dynamics, and simulated comparisons with previous policy search methods. As the authors mentioned, "this is the first method that can train deep visuomotor policies for complex, high-dimensional manipulation skills with direct torque control".

5.4 Trust Region Policy Optimization

Schulman et al. (2015) introduced an iterative procedure to monotonically improve policies, and proposed a practical algorithm, Trust Region Policy Optimization (TRPO), by making several approximations. The authors also unified policy iteration and policy gradient with analysis. In the experiments, TRPO methods performed well on simulated robotic tasks of swimming, hopping, and walking, as well as playing Atari games in an end-to-end manner directly from raw images.

5.5 Benchmark Results

Duan et al. (2016) presented a benchmark for continuous control tasks, including classic tasks like cart-pole, tasks with very large state and action spaces such as 3D humanoid locomotion and tasks with partial observations, and tasks with hierarchical structure, implemented various algorithms, including batch algorithms: REINFORCE, Truncated Natural Policy Gradient (TNPG), Reward-

Weighted Regression (RWR), Relative Entropy Policy Search (REPS), Trust Region Policy Optimization (TRPO), Cross Entropy Method (CEM), Covariance Matrix Adaption Evolution Strategy (CMA-ES); online algorithms: Deep Deterministic Policy Gradient (DDPG); and recurrent variants of batch algorithms. The open source is available at: https://github.com/ rllab/rllab.

Duan et al. (2016) compared various algorithms, and showed that DDPG, TRPO, and Truncated Natural Policy Gradient (TNPG) (Schulman et al., 2015) are effective in training deep neural network policies, yet better algorithms are called for hierarchical tasks.

5.6 COMBINING POLICY GRADIENT AND Q-LEARNING

O'Donoghue et al. (2017) proposed to combine policy gradient with off-policy Q-learning (PGQ), to benefit from experience replay. Usually actor-critic methods are on-policy. The authors also showed that action value fitting techniques and actor-critic methods are equivalent, and interpreted regularized policy gradient techniques as advantage function learning algorithms. Empirically, the authors showed that PGQ outperformed DQN and A3C on Atari games.

6 REWARD

Inverse reinforcement learning (IRL) is the problem of determining a reward function given observations of optimal behaviour (Ng and Russell, 2000). In imitation learning, or apprenticeship learning, an agent learns to perform a task from expert demonstrations, with samples of trajectories from the expert, without reinforcement signal, without additional data from the expert while training; two main approaches for imitation learning are behavioral cloning and inverse reinforcement learning; behavioral cloning is formulated as a supervised learning problem to map state-action pairs from expert trajectories to policy (Ho and Ermon, 2016).

6.1 GENERATIVE ADVERSARIAL NETWORKS

Goodfellow et al. (2014) proposed generative adversarial nets (GANs) to estimate generative models via an adversarial process by training two models simultaneously, a generative model G to capture the data distribution, and a discriminative model D to estimate the probability that a sample comes from the training data but not the generative model G.

Goodfellow et al. (2014) modelled G and D with multilayer perceptrons: $G(z : \theta_g)$ and $D(x : \theta_d)$, where θ_g and θ_d are parameters, x are data points, and z are input noise variables. Define a prior on input noise variable $p_z(z)$. G is a differentiable function and $D(x)$ outputs a scalar as the probability that x comes from the training data rather than p_g, the generative distribution we want to learn.

D will be trained to maximize the probability of assigning labels correctly to samples from both training data and G. Simultaneously, G will be trained to minimize such classification accuracy, $\log(1 - D(G(z)))$. As a result, D and G form the two-player minimax game as follows:

$$\min_G \max_D E_{x \sim p_{data}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

Goodfellow et al. (2014) showed that as G and D are given enough capacity, generative adversarial nets can recover the data generating distribution, and provided a training algorithm with backpropagation by minibatch stochastic gradient descent.

Generative adversarial networks have received much attention. See Goodfellow (2017) for Ian Goodfellow's summary of his NIPS 2016 Tutorial.

6.2 GENERATIVE ADVERSARIAL IMITATION LEARNING

With IRL, an agent learns a reward function first, then from which derives an optimal policy. Many IRL algorithms have high time complexity, with an RL problem in the inner loop.

Ho and Ermon (2016) proposed generative adversarial imitation learning algorithm to learn policies directly from data, bypassing the intermediate IRL step. Generative adversarial training was de-

ployed to fit the discriminator, the distribution of states and actions that defines expert behavior, and the generator, the policy.

Generative adversarial imitation learning finds a policy π_θ so that a discriminator $\mathcal{D}_R$ can not distinguish states following the expert policy π_E and states following the imitator policy π_θ, hence forcing $\mathcal{D}_R$ to take 0.5 in all cases and π_θ not distinguishable from π_E in the equillibrium. Such a game is formulated as:

$$\max_{\pi_\theta} \min_{\mathcal{D}_R} -E_{\pi_\theta}[\log \mathcal{D}_R(s)] - E_{\pi_E}[\log(1 - \mathcal{D}_R(s))]$$

The authors represented both π_θ and $\mathcal{D}_R$ as deep neural networks, and found an optimal solution by repeatedly performing gradient updates on each of them. $\mathcal{D}_R$ can be trained with supervised learning with a data set formed from traces from a current π_θ and expert traces. For a fixed $\mathcal{D}_R$, an optimal π_θ is sought. Hence it is a policy optimization problem, with $-\log \mathcal{D}_R(s)$ as the reward. The authors trained π_θ by trust region policy optimization (Schulman et al., 2015).

Finn et al. (2016) established a connection between GANs, IRL, and energy-based models. Pfau and Vinyals (2016) established the connection between GANs and actor-critic algorithms.

7 PLANNING

Planning constructs a value function or a policy usually with a model. Tamar et al. (2016) introduced Value Iteration Networks (VIN), a fully differentiable CNN planning module to approximate the value iteration algorithm, to learn to plan, e.g, policies in RL. In contrast to conventional planning, VIN is model-free, where reward and transition probability are part of the neural network to be learned, so that it avoids issues with system identification. VIN can be trained end-to-end with back-propagation. VIN can generalize in a diverse set of tasks: simple gridworlds, Mars Rover Navigation, continuous control and WebNav Challenge for Wikipedia links navigation (Nogueira and Cho, 2016). One merit of Value Iteration Network, as well as Dueling Network(Wang et al., 2016b), is that they design novel deep neural networks architectures for reinforcement learning problems. See a blog about VIN at goo.gl/Dr8gKL.

8 ATTENTION AND MEMORY

Attention and memory are two important mechanisms, which work together in many cases.

Mnih et al. (2014) introduced the recurrent attention model (RAM) to focus on selected sequence of regions or locations from an image or video for image classification and object detection. The authors used RL methods, in particular, REINFORCE algorithm, to train the model, to overcome the issue that the model is non-differentiable, and experimented on an image classification task and a dynamic visual control problem. Xu et al. (2015) integrated attention to image captioning, trained the hard version attention with the REINFORCE algorithm, and showed the effectiveness of attention on Flickr8k, Flickr30k, and MS COCO datasets. The attention mechanism is also deployed in NLP, e.g., in Bahdanau et al. (2015; 2017), and with external memory, in differentiable neural computer (Graves et al., 2016).

Graves et al. (2016) proposed differentiable neural computer (DNC), in which, a neural network can read from and write to an external memory, so that DNC can solve complex, structured problems, which a neural network without read-write memory can not solve. DNC minimizes memory allocation interference and enables long-term storage. Similar to a conventional computer, in a DNC, the neural network is the controller and the external memory is the random-access memory; and a DNC represents and manipulates complex data structures with the memory. Differently, a DNC learns such representation and manipulation end-to-end with gradient descent from data in a goal-directed manner. When trained with supervised learning, a DNC can solve synthetic question answering problems, for reasoning and inference in natural language; it can solve the shortest path finding problem between two stops in transportation networks and the relationship inference problem in a family tree. When trained with reinforcement learning, a DNC can solve a moving blocks puzzle with changing goals specified by symbol sequences. DNC outperformed normal neural network like

LSTM or DNC's precursor Neural Turing Machine (Graves et al., 2014); with harder problems, an LSTM may simply fail. Although these experiments are relatively small-scale, we expect to see further improvements and applications of DNC.

See Deepmind's description of DNC at goo.gl/58mgoX. See more work on attention and/or memory, e.g., Ba et al. (2014; 2016); Chen et al. (2016a); Danihelka et al. (2016); Eslami et al. (2016); Gregor et al. (2015); Jaderberg et al. (2015); Kaiser and Bengio (2016); Kadlec et al. (2016); Oquab et al. (2015); Weston et al. (2015); Sukhbaatar et al. (2015); Yang et al. (2015); Zagoruyko and Komodakis (2017); Zaremba and Sutskever (2015). See goo.gl/ArW2nE and goo.gl/UukROv for blogs about attention and memory.

9 UNSUPERVISED LEARNING

Jaderberg et al. (2017) proposed UNsupervised REinforcement and Auxiliary Learning (UNREAL) to improve learning efficiency by maximizing pseudo-reward functions, besides the usual cumulative reward, while sharing a common representation. UNREAL benefits from learning from the abundant possible training signals, especially when the extrinsic reward signals are rarely observed. UNREAL is composed of RNN-LSTM base agent, pixel control, reward prediction, and value function replay. The base agent is trained on-policy with A3C. Experiences of observations, rewards and actions are stored in a reply buffer, for being used by auxiliary tasks. The auxiliary policies use the base CNN and LSTM, together with a deconvolutional network, to maximize changes in pixel intensity of different regions of the input images. The reward prediction module predicts short-term extrinsic reward in next frame by observing the last three frames, to tackle the issue of reward sparsity. Value function replay further trains the value function. UNREAL improved A3C's performance on Atari games, and performed well on 3D Labyrinth game. See Deepmind's description of UNREAL at goo.gl/zhqBGy.

We discuss robotics navigation with similar unsupervised auxiliary learning in Section 13, and generative adversarial networks (GANs), a recent unsupervised learning framework, in Section 6. See Sutton et al. (2011) for Horde, a scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction.

10 LEARNING TO LEARN

Learning to learn is related to transfer learning, multi-task learning or representation learning, and is a core ingredient to achieve strong AI (Lake et al., 2016). Learning to learn is also related to meta learning or one-shot learning.

Duan et al. (2017) and Wang et al. (2016a) proposed to learn a flexible RNN model to handle a family of RL tasks, to improve sample efficiency, learn new tasks in a few samples, and benefit from prior knowledge. The agent is modelled with RNN, with inputs of observations, rewards, actions and termination flags; the weights of RNN are trained with RL,TRPO in Duan et al. (2017) and A3C in Wang et al. (2016a), and achieve similar performance for various problems to specific RL algorithms. Duan et al. (2017) experimented with multi-arm bandits, tabular MDPs and visual navigation, and discussed that for larger problems, better RL algorithms are needed to train RNN. Wang et al. (2016a) experimented with bandits with independent arms, bandits with dependant arms, restless arms and MDPs. A future work is to improve scalability.

Li and Malik (2017) proposed to automate unconstrained continuous optimization algorithms with guided policy search (Levine et al., 2016a) by representing a particular optimization algorithm as a policy, and convergence rate as reward.

11 GAMES

Games provide excellent testbeds for RL/AI algorithms. We discuss Deep Q-Network (DQN) in Section 3 and its extensions, all of which experimented with Atari games. We discuss Mnih et al. (2016) in Section 4, Jaderberg et al. (2017) in Section 9, and Mirowski et al. (2017) in Section 13, and they used Labyrinth as the testbed.

Backgammon and Go are perfect information games. We discuss briefly Backgammon in Section 11.1 about board games. We talk about video games like Doom in Section 11.2. We put poker, a board game, under Section 11.3 about imperfect information games, where game theory is concerned. Video games like Labyrinth and Doom are usually imperfect information games, whereas game theory is not (yet) used in these work to approach the problem. We single out AlphaGo (Silver et al., 2016) in Section 12, for its significance.

11.1 BOARD GAMES

Board games, e.g., backgammon, Go, chess, checker and othello, are classical testbeds for RL/AI algorithms. Tesauro (1994) approached backgammon by using neural networks to approximate value function learned with TD learning, and achieved human level performance.

11.2 VIDEO GAMES

Wu and Tian (2017) deployed A3C with CNN to train an agent in a partially observable 3D environment, Doom, from recent four raw frames and game variables, to predict next action and value function, following the curriculum learning (Bengio et al., 2009) approach of starting with simple tasks and gradually transition to harder ones. It is nontrivial to apply A3C to such 3D games directly, partly due to sparse and long term reward. The authors won the champion in Track 1 of ViZDoom Competition by a large margin, and plan the following future work: a map from an unknown environment, localization, a global plan to act, and visualization of the reasoning process.

Dosovitskiy and Koltun (2017) approached the problem of sensorimotor control in immersive environments with supervised learning, and won the Full Deathmatch track of the Visual Doom AI Competition. We list it here since it is usually an RL problem, yet it was solved with supervised learning. Lample and Chaplot (2016) also discussed how to tackle Doom.

Usunier et al. (2016) studied StarCraft and Tessler et al. (2017) studied Minecraft.

11.3 IMPERFECT INFORMATION GAMES

Heinrich and Silver (2016) proposed Neural Fictitious Self-Play (NFSP) to combine fictitious self-play with deep RL to learn approximate Nash equilibria for games of imperfect information in a scalable end-to-end approach without prior domain knowledge. NFSP was evaluated on two-player zero-sum games. In Leduc poker, NFSP approached a Nash equilibrium, while common RL methods diverged. In Limit Texas Hold'em, a real-world scale imperfect-information game, NFSP performed similarly from scratch to state-of-the-art, superhuman algorithms which are based on significant domain expertise.

Heads-up Limit Hold'em Poker was essentially solved (Bowling et al., 2015) with counterfactual regret minimization (CFR), which is an iterative method to approximate a Nash equilibrium of an extensive-form game with repeated self-play between two regret-minimizing algorithms.

Recently, significant progress has been made for Heads-up No-Limit Hold'em Poker (Moravčík et al., 2017), the DeepStack computer program defeated professional poker players for the first time. DeepStack utilized the recursive reasoning of CFR to handle information asymmetry, focusing computation on specific situations arising when making decisions and use of value functions trained automatically, with little domain knowledge or human expert games, without abstraction and offline computation of complete strategies as before (Sandholm, 2015).

Imperfect information games, or game theory in general, have many applications, e.g., security and medical decision support (Sandholm, 2015). It is interesting to see more progress of deep RL in such applications, and the full version of Texas Hold'em.

12 ALPHAGO

AlphaGo (Silver et al., 2016), a computer Go program, won the human European Go champion, 5 games to 0, in October 2015, and became the first computer Go program to won a human professional Go player without handicaps on a full-sized 19×19 board. Soon after that in March 2016,

AlphaGo defeated Lee Sedol, an 18-time world champion Go player, 4 games to 1, making headline news worldwide. This set a landmark in AI. The challenge of solving Go comes from not only the gigantic search space of about 250^{150}, an astronomical number, but also the hardness of position evaluation, which was successfully used in solving many other games, like backgammon and chess.

12.1 TRAINING PIPELINE AND MCTS

We discuss briefly how AlphaGo works based on Silver et al. (2016) and Sutton and Barto (2017). See Chapter 16 in Sutton and Barto (2017) for a detailed and intuitive description of AlphaGo. See Deepmind's description of AlphaGo at goo.gl/lZoQ1d.

AlphaGo was built with techniques of deep CNN, supervised learning, reinforcement learning, and Monte Carlo tree search (MCTS) (Browne et al., 2012; Gelly et al., 2012). AlphaGo is composed of two phases: neural network training pipeline and MCTS. The training pipeline phase includes training a supervised learning (SL) policy network from expert moves, a fast rollout policy, an RL policy network, and an RL value network.

The SL policy network has convolutional layers, ReLU nonlinearities, and an output softmax layer representing probability distribution over legal moves. The inputs to the CNN are $19 \times 19 \times 48$ image stacks, where 19 is the dimension of a Go board and 48 is the number of features. State-action pairs are sampled from expert moves to train the network with stochastic gradient ascent to maximize the likelihood of the move selected in a given state. The fast rollout policy uses a linear softmax with small pattern features.

The RL policy network improves SL policy network, with the same network architecture, and the weights of SL policy network as initial weight, and policy gradient for training. The reward function is +1 for winning and -1 for losing in the terminal states, and 0 otherwise. Games are played between the current policy network and a random, previous iteration of the policy network, to stabilize the learning and to avoid overfitting. Weights are updated by stochastic gradient ascent to maximize the expected outcome.

The RL value network still has the same network architecture as SL policy network, except the output is a single scalar predicting the value of a position. The value network is learned in a Monte Carlo policy evaluation approach. To tackle the overfitting problem caused by strongly correlated successive positions in games, data are generated by self-play between the RL policy network and itself until game termination. The weights are trained by regression on state-outcome pairs, using stochastic gradient descent to minimize the mean squared error between the prediction and the corresponding outcome.

In MCTS phase, AlphaGo selects moves by lookahead search. It builds a partial game tree starting from the current state, in the following stages: 1) select a promising node to explore further, 2) expand a leaf node guided by the SL policy network and collected statistics, 3) evaluate a leaf node with a mixture of the RL value network and the rollout policy, 4) backup evaluations to update the action values. A move is then selected.

12.2 DISCUSSIONS

The Deepmind team integrated several existing techniques together to engineered AlphaGo and it has achieved tremendous results. However, the RL policy network and RL value network are not strong/accurate enough, so that the RL value network, together with the SL policy network and the rollout network, assist MCTS to search for the move. This might explain the one game loss against Lee Sedol. Moverover, AlphaGo still requires manually defined features with human knowledge, so it is not entirely an end-to-end solution yet; in contrast, DQN requires only raw pixels and scores as inputs. Such a room for improvements would inspire intellectual inquisition for better computer Go programs, potentially with deep RL only, without MCTS, like TD-Gammon (Sutton and Barto, 2017). This would be based on a novel RL algorithm, a novel deep neural network architecture, and powerful computation. New RL algorithms are called for, for data efficiency, and possibly for better knowledge representation and reasoning. New deep neural network architectures are called for, for the sophistication to represent complex scenarios in Go and the elegance for learning in a reasonable time, so that an optimal policy and/or an optimal value function can be directly approximated to

make decisions without the help of MCTS to choose moves. Admittedly, such endeavour would be illusive at large currently.

Being more practical, we expect more applications/extensions of techniques in Silver et al. (2016) in solving problems requiring titanic search spaces, like classical AI problems, e.g., planning, scheduling, and constraint satisfaction, etc.

13 ROBOTICS

As we discuss in Section 5, Schulman et al. (2015) proposed Trust Region Policy Optimization (TRPO), and experimented with simulated robotic tasks, and Levine et al. (2016a) proposed Guided Policy Search (GPS) to handle physical robots.

Mirowski et al. (2017) obtained the navigation ability by solving an RL problem maximizing cumulative reward and jointly considering un/self-supervised tasks to improve data efficiency and task performance. The authors addressed the sparse reward issues by augmenting the loss with two auxiliary tasks, 1) unsupervised reconstruction of a low-dimensional depth map for representation learning to aid obstacle avoidance and short-term trajectory planning; 2) self-supervised loop closure classification task within a local trajectory. The authors incorporated a stacked LSTM to use memory at different time scales for dynamic elements in the environments. The proposed agent learn to navigate in complex 3D mazes end-to-end from raw sensory input, and performed similarly to human level, even when start/goal locations change frequently.

In this approach, navigation is a by-product of the goal-directed RL optimization problem, in contrast to conventional approaches such as Simultaneous Localisation and Mapping (SLAM), where explicit position inference and mapping are used for navigation. This may have the chance to replace the popular SLAM, which usually requires manual processing.

See more recent robotics papers, e.g., Chebotar et al. (2016); Finn and Levine (2016); Gu et al. (2016a); Levine et al. (2016b); Yahya et al. (2016); Zhu et al. (2016). See Kober et al. (2013) for a survey of RL in robotics. See Science Robotics.

14 SPOKEN DIALOGUE SYSTEMS

In spoken dialogue systems, conversational agent, or simply, chatbot, human and computer interacts with natural speech. There are usually two categories: chat-oriented and task-oriented systems; the former aims to converse with users in contextually reasonable way; the latter aims to assist users for specific goals (Su et al., 2016b).

Li et al. (2016c) proposed to use deep RL to generate dialogues to model future reward for better informativity, coherence, and ease of answering, to attempt to address the issues in the sequence to sequence models based on Sutskever et al. (2014): the myopia and misalignment of maximizing the probability of generating a response given the previous dialogue turn, and the infinite loop of repetitive responses. The authors designed a reward function to reflect the above desirable properties, and deployed policy gradient to optimize the long term reward. It would be interesting to investigate the reward model with the approach in Su et al. (2016b) below or with inverse RL and imitation learning as discussed in Section 6, although Su et al. (2016b) mentioned that such methods are costly, and humans may not act optimally.

Su et al. (2016b) proposed an on-line learning framework to train the dialogue policy jointly with the reward model via active learning with a Gaussian process model, to tackle the issue that it is unreliable and costly to use explicit user feedback as the reward signal. The authors showed empirically that the proposed framework reduced manual data annotations significantly and mitigated noisy user feedback in dialogue policy learning.

Li et al. (2016d) designed a user simulator for movie booking, with both rules and collected data, for movie ticket booking and movie seeking. The source code is available at: goo.gl/jOv4AR.

Some recent papers follow: Asri et al. (2016), Bordes and Weston (2016), Chen et al. (2016b), Dhingra et al. (2016), Fatemi et al. (2016), Li et al. (2016a), Lipton et al. (2016),

Mesnil et al. (2015), Mo et al. (2016), Shah et al. (2016), Su et al. (2016a), Wen et al. (2015a), Williams and Zweig (2016), Yang et al. (2016), Zhao and Eskenazi (2016).

See Li Deng's recent talk at goo.gl/BqzeIZ. See conferences like SIGDIAL and INTERSPEECH. See NIPS 2016 Workshop on End-to-end Learning for Speech and Audio Processing, and NIPS 2015 Workshop on Machine Learning for Spoken Language Understanding and Interactions.

15 MACHINE TRANSLATION

He et al. (2016a) proposed dual learning mechanism to tackle the data hunger issue in machine translation, inspired by the observation that the information feedback between the primal, translation from language A to language B, and the dual, translation from B to A, can help improve both translation models, with a policy gradient method, using the language model likelihood as the reward signal. Experiments showed that, with only 10% bilingual data for warm start and monolingual data, the dual learning approach performed comparably with previous neural machine translation methods with full bilingual data in English to French tasks. The dual learning mechanism may have extensions to many tasks, if the task has a dual form, e.g., speech recognition and text to speech, image caption and image generation, question answering and question generation, search and keyword extraction, etc.

See Sutskever et al. (2014); Bahdanau et al. (2015) for sequence to sequence neural machine translation. See Wu et al. (2016) for Google's Neural Machine Translation System.

16 TEXT SEQUENCE PREDICTION

Text generation models are usually based on n-gram, feed-forward neural networks, or recurrent neural networks, trained to predict next word given the previous ground truth words as inputs; then in testing, the trained models are used to generate a sequence word by word, using the generated words as inputs. The errors will accumulate on the way, causing the exposure bias issue. Moreover, these models are trained with word level losses, e.g., cross entropy, to maximize the probability of next word; however, the models are evaluated on a different metrics like BLEU.

Ranzato et al. (2016) proposed Mixed Incremental Cross-Entropy Reinforce (MIXER) for sequence prediction, with incremental learning and a loss function combining both REINFORCE and cross-entropy. MIXER is a sequence level training algorithm, aligning training and testing objective, such as BLEU, rather than predicting the next word as in previous works.

Bahdanau et al. (2017) proposed an actor-critic algorithm for sequence prediction, attempting to further improve Ranzato et al. (2016). The authors utilized a critic network to predict the value of a token, i.e., the expected score following the sequence prediction policy, defined by an actor network, trained by the predicted value of tokens. Some techniques are deployed to improve performance: SARSA rather than Monter-Carlo method to lessen the variance in estimating value functions; target network for stability; sampling prediction from a delayed actor whose weights are updated more slowly than the actor to be trained, to avoid the feedback loop when actor and critic need to be trained based on the output of each other; reward shaping to avoid the issue of sparse training signal.

Yu et al. (2017) proposed SeqGAN, sequence generative adversarial nets with policy gradient, integrating the adversarial scheme in Goodfellow et al. (2014). Li et al. (2017) proposed to improve sequence generation by considering the knowledge about the future.

17 NEURAL ARCHITECTURE DESIGN

Neural networks architecture design is a notorious, nontrivial engineering issue. Neural architecture search provides a promising avenue to explore.

Zoph and Le (2017) proposed the neural architecture search to generate neural networks architectures with an RNN trained by RL, in particular, REINFORCE, searching from scratch in variable-length architecture space, to maximize the expected accuracy of the generated architectures on a validation set. In the RL formulation, a controller generates hyperparameters as a sequence of tokens, which are actions chosen from hyperparameters spaces; each gradient update to the policy

parameters corresponds to training one generated network to convergence; an accuracy on a validation set is the reward signal. The neural architecture search can generate convolutional layers, with skip connections or branching layers, and recurrent cell architecture. The authors designed a parameter server approach to speed up training. Comparing with state of the art methods, the proposed approach achieved competitive results for an image classification task with CIFAR-10 dataset; and better results for a language modeling task with Penn Treebank. See also Baker et al. (2017).

18 PERSONALIZED WEB SERVICES

Li et al. (2010) formulated personalized news articles recommendation as a contextual bandit problem, to learn an algorithm to select articles sequentially for users based on contextual information of the user and articles, such as historical activities of the user and descriptive information and categories of content, and to take user-click feedback to adapt article selection policy to maximize total user clicks in the long run.

Theocharous et al. (2015) formulated a personalized Ad recommendation systems as an RL problem to maximize life-time value (LTV) with theoretical guarantees. This is in contrast to a myopic solution with supervised learning or contextual bandit formulation, usually with the performance metric of click through rate (CTR). As the models are hard to learn, the authors deployed a model-free approach to computes a lower-bound on the expected return of a policy to address the off-policy evaluation problem, i.e., how to evaluate a RL policy without deployment.

Li et al. (2015) also attempted to maximize lifetime value of customers. Silver et al. (2013) proposed concurrent reinforcement learning for the customer interaction problem. See Chapter 16 in Sutton and Barto (2017) for a detailed and intuitive description of personalized web services.

19 HEALTHCARE

There are many opportunities and challenges in healthcare for machine learning (Saria, 2014). Personalized medicine is getting popular in healthcare. It systematically optimizes the patient's health care, in particular, for chronic conditions and cancers using individual patient information, potentially from electronic health/medical record (EHR/EMR). Here dynamic treatment regimes (DTRs) or adaptive treatment strategies are sequential decision making problems. Some issues in DTRs are not in standard RL. Shortreed et al. (2011) tackled the missing data problem, and designed methods to quantify the evidence of the learned optimal policy. Goldberg and Kosorok (2012) proposed methods for censored data (patients may drop out during the trial) and flexible number of stages. See Chakraborty and Murphy (2014) for a recent survey, and Kosorok and Moodie (2015) for an edited book about recent progress in DTRs. Currently Q-learning is the RL method in DTRs. It is interesting to see the applications of deep RL methods in this field.

Some recent workshops at the intersection of machine learning and healthcare are: NIPS 2016 Workshop on Machine Learning for Health (http://www.nipsml4hc.ws) and NIPS 2015 Workshop on Machine Learning in Healthcare (https://sites.google.com/site/nipsmlhc15/).

20 FINANCE

RL is a natural solution to some finance and economics problems (Hull, 2014; Luenberger, 1997), like option pricing (Longstaff and Schwartz, 2001; Tsitsiklis and Van Roy, 2001; Li et al., 2009), and multi-period portfolio optimization (Brandt et al., 2005), where value function based RL methods were used. Moody and Saffell (2001) proposed to utilize policy gradient to learn to trade; Deng et al. (2016) extended it with deep neural networks. Deep (reinforcement) learning would provide better solutions in some issues in risk management (Hull, 2014; Yu et al., 2009). The market efficiency hypothesis is fundamental in finance. However, there are well-known behavioral biases in human decision-making under uncertainty. A reconciliation is the adaptive markets hypothesis (Lo, 2004), which may be approached by reinforcement learning.

It is nontrivial for finance and economics academia to accept blackbox methods like neural networks; Heaton et al. (2016) may be regarded as an exception. However, there is a lecture in AFA 2017

annual meeting: Machine Learning and Prediction in Economics and Finance (goo.gl/7xdePd). A (obvious) factor is financial firms would probably hold state-of-the-art research/application results.

21 MUSIC GENERATION

Jaques et al. (2017) proposed to combine maximum likelihood estimation with RL training, using RL to impose structure on an RNN trained on data by choosing reward functions, to attempt to ensure coherent global structure in multi-step generated sequences. A Note-RNN was trained to predict the next note in a musical sequence with a large corpus of songs. Then the Note-RNN was refined using RL to obtain RL Tuner, with a reward function considering both rules of music theory and output of another trained Note-RNN. RL Tuner produced more pleasant-sounding and subjectively pleasing melodies than alternative methods. The proposed approach has the potential for training sequence models other than music, by allowing for encoding high-level domain knowledge into the RNN.

22 TO-DO LIST

We list interesting and/or important directions/papers we have not discussed in this overview as below, hoping it would provide pointers for those who may be interested in studying them further.[3] This would be part of our future work.[4]

- understanding deep learning, Daniely et al. (2016); Li et al. (2016b); Zhang et al. (2017)
- exploration, e.g., Stadie et al. (2015); Bellemare et al. (2016); Kulkarni et al. (2016); Osband et al. (2016); Nachum et al. (2017)
- model-based learning, e.g., Oh et al. (2015); Gu et al. (2016b)
- retrace algorithm, Munos et al. (2016)
- predictron, Silver et al. (2017)
- hierarchical RL, e.g., Kulkarni et al. (2016); Vezhnevets et al. (2016); Tessler et al. (2017); Florensa et al. (2017)
- transfer/multitask RL, e.g., Maurer et al. (2016); Mo et al. (2016); Parisotto et al. (2016), NIPS 2015 Transfer and Multi-Task Learning: Trends and New Perspectives Workshop
- zero/one-shot learning, e.g., Vinyals et al. (2016); Lake et al. (2015); Johnson et al. (2016)
- semi-supervised RL, e.g., Finn et al. (2017)
- deep symbolic RL, Garnelo et al. (2016)
- intrinsic motivation, e.g., Stadie et al. (2015); Kulkarni et al. (2016); Oudeyer et al. (2016)
- hyperparameter learning, e.g. Andrychowicz et al. (2016)
- information extraction, e.g., Narasimhan et al. (2016)
- text games, e.g., He et al. (2016b); Narasimhan et al. (2015)
- language tree-structure learning, e.g., Yogatama et al. (2017)
- question answering, e.g., Shen et al. (2016); Trischler et al. (2016)
- large action space, e.g., Dulac-Arnold et al. (2016); He et al. (2016c)
- adaptive normalization, van Hasselt et al. (2016b)
- self-driving vehicle, e.g., Bojarski et al. (2016), NIPS 2016 Workshop on Machine Learning for Intelligent Transportation Systems
- smart grid, e.g., Wen et al. (2015b)
- physics experiments, e.g., Denil et al. (2016)

[3] Some topics/papers may not contain RL yet. However, we believe these are interesting and/or important directions for RL in the sense of either theory or application.

[4] It would be definitely more desirable if we could finish reviewing these before publishing this overview. One factor is we set the deadline for the first version before January 28, 2017, the Chinese Spring Festival.

- deep probabilistic programming, Tran et al. (2017)
- deep learning games, Schuurmans and Zinkevich (2016)
- program learning, e.g., Reed and de Freitas (2016)
- quantum RL, e.g., Crawford et al. (2016), NIPS 2015 Workshop on Quantum Machine Learning

23 RESOURCES

We list some resources for Deep RL, which by no means are complete.

23.1 BOOKS

- The definite and intuitive reinforcement learning book by Richard S. Sutton and Andrew G. Barto (Sutton and Barto, 2017)
- Concise and theoretical, Algorithms for Reinforcement Learning by Csaba Szepesvári (Szepesvári, 2010)
- A theoretical book about approximate dynamic programming by Dimitri P. Bertsekas (Bertsekas, 2012)
- An operations research oriented book, Approximate Dynamic Programming, by Warren B. Powell (Powell, 2011)
- Deep learning book by Ian Goodfellow, Yoshua Bengio, and Aaron Courville (Goodfellow et al., 2016)

23.2 COURSES

- David Silver, Reinforcement Learning, 2015, slides (goo.gl/UqaxlO), video-lectures (goo.gl/7BVRkT)
- Sergey Levine, John Schulman and Chelsea Finn, CS 294: Deep Reinforcement Learning, Spring 2017, http://rll.berkeley.edu/deeprlcourse/
- Charles Isbell, Michael Littman and Pushkar Kolhe, Udacity: Machine Learning: Reinforcement Learning, goo.gl/eyvLfg
- Fei-Fei Li, Andrej Karpathy and Justin Johnson, CS231n: Convolutional Neural Networks for Visual Recognition, http://cs231n.stanford.edu
- Richard Socher, CS224d: Deep Learning for Natural Language Processing, http://cs224d.stanford.edu
- Nando de Freitas, Deep Learning Lectures, https://www.youtube.com/user/ProfNandoDF

23.3 TUTORIALS

- David Silver, Deep Reinforcement Learning, ICML 2016
- Pieter Abbeel and John Schulman, Deep Reinforcement Learning Through Policy Optimization, NIPS 2016
- Andrew Ng, Nuts and Bolts of Building Applications using Deep Learning, NIPS 2016
- John Schulman, The Nuts and Bolts of Deep Reinforcement Learning Research, Deep Reinforcement Learning Workshop, NIPS 2016
- John Schulman, Deep Reinforcement Learning, Deep Learning School, 2016
- Pieter Abbeel, Deep Reinforcement Learning, Deep Learning Summer School, 2016; http://videolectures.net/deeplearning2016_abbeel_deep_reinforcement/
- David Silver, Deep Reinforcement Learning, 2nd Multidisciplinary Conference on Reinforcement Learning and Decision Making (RLDM), Edmonton 2015; http://videolectures.net/rldm2015_silver_reinforcement_learning/

- Rich Sutton, Introduction to Reinforcement Learning with Function Approximation, https://www.microsoft.com/en-us/research/video/tutorial-introduction-to-reinforcement-learning-with-function-approximation/
- Joelle Pineau, Introduction to Reinforcement Learning, Deep Learning Summer School, 2016; http://videolectures.net/deeplearning2016_pineau_reinforcement_learning/
- Deep Learning Summer School, 2016, 2015

23.4 Conferences, Journals and Workshops

- NIPS: Neural Information Processing Systems
- ICML: International Conference on Machine Learning
- ICLR: International Conference on Learning Representation
- RLDM: Multidisciplinary Conference on Reinforcement Learning and Decision Making
- AAAI, IJCAI, ACL, EMNLP, SIGDIAL, ICRA, IROS, KDD, SIGIR, CVPR, etc.
- Science Robotics, JMLR, MLJ, AIJ, JAIR, PAMI, etc
- Nature May 2015, Science July 2015, survey papers on machine learning/AI
- Deep Reinforcement Learning Workshop, NIPS 2016, 2015; IJCAI 2016
- Deep Learning Workshop, ICML 2016

23.5 Blogs

- Andrej Karpathy, karpathy.github.io, esp. goo.gl/1hkKrb
- Denny Britz, www.wildml.com, esp. goo.gl/MyrwDC
- Junling Hu, Reinforcement learning explained - learning to act based on long-term payoffs
- Li Deng, How deep reinforcement learning can help chatbots
- Christopher Olah, colah.github.io

In the current information/social media age, we are overwhelmed by information, e.g., from Twitter, Google+, WeChat, arXiv, etc. The skill to efficiently select the best information becomes essential.

24 Discussions

It is both the best and the worst of times for the field of deep RL, for the same reason: it has been growing so fast and so enormously. We have been witnessing breakthroughs, exciting new methods and applications, and we expect to see much more and much faster. As a consequence, this overview is incomplete, in the sense of both depth and width. However, we attempt to summarize important achievements and discuss potential directions and applications in this amazing field.

We have been witnessing breakthroughs, three papers about or using Deep RL published in Nature in less than two years: deep Q-network (Mnih et al., 2015), AlphaGo (Silver et al., 2016) and differentiable neural computer (Graves et al., 2016); We have already seen many extensions to, improvements for and applications of deep Q-network (Mnih et al., 2015). The mechanisms of attention and memory (Graves et al., 2016) has been attracting much attention.

Novel architectures and applications using deep RL were recognized in top tier conferences as best (student) papers in 2016: dueling network architectures (Wang et al., 2016a) at ICML, spoken dialogue systems (Su et al., 2016b) at ACL (student), information extraction (Narasimhan et al., 2016) at EMNLP, and value iteration networks (Tamar et al., 2016) at NIPS. Exciting achievements abound: asynchronous methods (Mnih et al., 2016), dual learning for machine translation (He et al., 2016a), guided policy search (Levine et al., 2016a), generative adversarial imitation learning (Ho and Ermon, 2016), unsupervised reinforcement and auxiliary learning (Jaderberg et al., 2017), and neural architecture design (Zoph and Le, 2017), etc.

Value function is central to reinforcement learning, e.g., in deep Q-network and its many extentions. Policy optimization approaches have been gaining traction, in many, diverse applications,

e.g., robotics, neural architecture design, spoken dialogue systems, machine translation, attention, and learning to learn, and this list is boundless. New learning mechanisms have emerged, e.g., using unsupervised/semi-supervised/tranfer learning to improve the quality and speed of learning, and more new mechanisms will be emerging. This is the renaissance of reinforcement learning (Krakovsky, 2016). In fact, reinforcement learning and deep learning have been making steady progress even in the AI winter.

It is essential to consider issues of learning models, like stability, convergence, accuracy, data efficiency, scalability, speed, simplicity, interpretability, robustness, and safety, etc. It is important to investigate comments/criticisms, e.g., from conginitive science, like intuitive physics, intuitive psychology, causal model, compositionality, learning to learn, and act in real time (Lake et al., 2016), for stronger AI. See also Peter Norvig's perspective at goo.gl/obvmVB.

Deep learning, in this third wave of AI, will have deeper influences, as we have already seen many achievements. Reinforcement learning, as a more general learning and decision making paradigm, will deeply influence deep learning, machine learning, and artificial intelligence in general.[5] It is interesting to mention that when Professor Rich Sutton started working in the University of Alberta in 2003, he named his lab RLAI: Reinforcement Learning and Artificial Intelligence.

ACKOWLEDGEMENT

I appreciate comments from Baochun Bai, Junling Hu, Ruitong Huang, Lihong Li, Dale Schuurmans, David Silver, Rich Sutton, Csaba Szepesvári, Yi Wan and Qing Yu. Any remaining issues and errors are my own. This document also benefits from discussions during various seminars/webinars, in particular, an AlphaGo seminar at MIT in April 2016, deep (reinforcement) learning seminars at the University of Toronto, McGill University and the University of Alberta in October 2016 as part of the North America tour of Synced (Jiqizhixin), and webinars using David Silver's slides in November and December 2016, and discussions in several WeChat groups.

REFERENCES

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. (2016). Concrete Problems in AI Safety. *ArXiv e-prints*.

Andrychowicz, M., Denil, M., Colmenarejo, S. G., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and de Freitas, N. (2016). Learning to learn by gradient descent by gradient descent. In *the Annual Conference on Neural Information Processing Systems (NIPS)*.

Asri, L. E., He, J., and Suleman, K. (2016). A sequence-to-sequence model for user simulation in spoken dialogue systems. In *Annual Meeting of the International Speech Communication Association (INTERSPEECH)*.

Ba, J., Hinton, G. E., Mnih, V., Leibo, J. Z., and Ionescu, C. (2016). Using fast weights to attend to the recent past. In *the Annual Conference on Neural Information Processing Systems (NIPS)*.

Ba, J., Mnih, V., and Kavukcuoglu, K. (2014). Multiple object recognition with visual attention. In *the International Conference on Learning Representations (ICLR)*.

Babaeizadeh, M., Frosio, I., Tyree, S., Clemons, J., and Kautz, J. (2017). Reinforcement learning through asynchronous advantage actor-critic on a gpu. *Submitted to Int'l Conference on Learning Representations*.

Bahdanau, D., Brakel, P., Xu, K., Goyal, A., Lowe, R., Pineau, J., Courville, A., and Bengio, Y. (2017). An actor-critic algorithm for sequence prediction. *Submitted to Int'l Conference on Learning Representations*.

[5] It is worthwhile to envision deep RL considering perspectives of government, academia and industry on AI, e.g., Artificial Intelligence, Automation, and the economy, Executive Office of the President, USA; Artificial Intelligence and Life in 2030 - One Hundred Year Study on Artificial Intelligence: Report of the 2015-2016 Study Panel, Stanford University; and AI, Machine Learning and Data Fuel the Future of Productivity by The Goldman Sachs Group, Inc., etc. See also the recent AI Frontiers Conference, https://www.aifrontiers.com.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *the International Conference on Learning Representations (ICLR)*.

Baker, B., Gupta, O., Naik, N., and Raskar, R. (2017). Designing neural network architectures using reinforcement learning. *Submitted to Int'l Conference on Learning Representations*.

Beattie, C., Leibo, J. Z., Teplyashin, D., Ward, T., Wainwright, M., Küttler, H., Lefrancq, A., Green, S., Valdés, V., Sadik, A., Schrittwieser, J., Anderson, K., York, S., Cant, M., Cain, A., Bolton, A., Gaffney, S., King, H., Hassabis, D., Legg, S., and Petersen, S. (2016). DeepMind Lab. *ArXiv e-prints*.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253?279.

Bellemare, M. G., Schaul, T., Srinivasan, S., Saxton, D., Ostrovski, G., and Munos, R. (2016). Unifying count-based exploration and intrinsic motivation. In *the Annual Conference on Neural Information Processing Systems (NIPS)*.

Bengio, Y. (2009). Learning deep architectures for ai. *Foundations and trends®in Machine Learning*, 2(1):1–127.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *the International Conference on Machine Learning (ICML)*.

Bernhard Wymann, E. E., Guionneau, C., Dimitrakakis, C., and Rémi Coulom, A. S. (2014). TORCS, The Open Racing Car Simulator. "http://www.torcs.org".

Bertsekas, D. P. (2012). *Dynamic programming and optimal control (Vol. II, 4th Edition: Approximate Dynamic Programming)*. Athena Scientific, Massachusetts, USA.

Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming*. Athena Scientific.

Bishop, C. (2011). *Pattern Recognition and Machine Learning*. Springer.

Bojarski, M., Testa, D. D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., and Zieba, K. (2016). End to End Learning for Self-Driving Cars. *ArXiv e-prints*.

Bordes, A. and Weston, J. (2016). Learning End-to-End Goal-Oriented Dialog. *ArXiv e-prints*.

Bowling, M., Burch, N., Johanson, M., and Tammelin, O. (2015). Heads-up limit hold'em poker is solved. *Science*, 347(6218):145–149.

Brandt, M. W., Goyal, A., Santa-Clara, P., and Stroud, J. R. (2005). A simulation approach to dynamic portfolio choice with an application to learning about return predictability. *The Review of Financial Studies*, 18(3):831–873.

Browne, C., Powley, E., Whitehouse, D., Lucas, S., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., and Colton, S. (2012). A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43.

Busoniu, L., Babuska, R., and Schutter, B. D. (2008). A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, 38(2).

Chakraborty, B. and Murphy, S. A. (2014). Dynamic treatment regimes. *Annual Review of Statistics and Its Application*, 1:447–464.

Chebotar, Y., Kalakrishnan, M., Yahya, A., Li, A., Schaal, S., and Levine, S. (2016). Path integral guided policy search. *ArXiv e-prints*.

Chen, Y.-N., Hakkani-Tur, D., Tur, G., Celikyilmaz, A., Gao, J., and Deng, L. (2016a). Knowledge as a Teacher: Knowledge-Guided Structural Attention Networks. *ArXiv e-prints*.

Chen, Y.-N. V., Hakkani-Tür, D., Tur, G., Gao, J., and Deng, L. (2016b). End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding. In *Annual Meeting of the International Speech Communication Association (INTERSPEECH)*.

Crawford, D., Levit, A., Ghadermarzy, N., Oberoi, J. S., and Ronagh, P. (2016). Reinforcement Learning Using Quantum Boltzmann Machines. *ArXiv e-prints*.

Daniely, A., Frostig, R., and Singer, Y. (2016). Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity. In *the Annual Conference on Neural Information Processing Systems (NIPS)*.

Danihelka, I., Wayne, G., Uria, B., Kalchbrenner, N., and Graves, A. (2016). Associative long short-term memory. In *the International Conference on Machine Learning (ICML)*.

Deng, L. and Dong, Y. (2014). *Deep Learning: Methods and Applications*. Now Publishers Inc.

Deng, L. and Liu, Y. (2017). *Deep Learning in Natural Language Processing (edited book, scheduled August 2017)*. Springer.

Deng, Y., Bao, F., Kong, Y., Ren, Z., and Dai, Q. (2016). Deep direct reinforcement learning for financial signal representation and trading. *IEEE Transactions on Neural Networks and Learning Systems*.

Denil, M., Agrawal, P., Kulkarni, T. D., Erez, T., Battaglia, P., and de Freitas, N. (2016). Learning to perform physics experiments via deep reinforcement learning. In *NIPS 2016 Deep Reinforcement Learning Workshop*.

Dhingra, B., Li, L., Li, X., Gao, J., Chen, Y.-N., Ahmed, F., and Deng, L. (2016). End-to-End Reinforcement Learning of Dialogue Agents for Information Access. *ArXiv e-prints*.

Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87.

Dosovitskiy, A. and Koltun, V. (2017). Learning to act by predicting the future. *Submitted to Int'l Conference on Learning Representations*.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. (2016). Benchmarking deep reinforcement learning for continuous control. In *the International Conference on Machine Learning (ICML)*.

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. (2017). RL^2: Fast reinforcement learning via slow reinforcement learning. *Submitted to Int'l Conference on Learning Representations*.

Dulac-Arnold, G., Evans, R., van Hasselt, H., Sunehag, P., Lillicrap, T., Hunt, J., Mann, T., Weber, T., Degris, T., and Coppin, B. (2016). Deep reinforcement learning in large discrete action spaces. In *the International Conference on Machine Learning (ICML)*.

Eslami, S. M. A., Heess, N., Weber, T., Tassa, Y., Szepesvári, D., Kavukcuoglu, K., and Hinton, G. E. (2016). Attend, infer, repeat: Fast scene understanding with generative models. In *the Annual Conference on Neural Information Processing Systems (NIPS)*.

Fatemi, M., Asri, L. E., Schulz, H., He, J., and Suleman, K. (2016). Policy networks with two-stage training for dialogue systems. In *the Annual SIGdial Meeting on Discourse and Dialogue (SIGDIAL)*.

Finn, C., Christiano, P., Abbeel, P., and Levine, S. (2016). A connection between GANs, inverse reinforcement learning, and energy-based models. In *NIPS 2016 Workshop on Adversarial Training*.

Finn, C. and Levine, S. (2016). Deep visual foresight for planning robot motion. *ArXiv e-prints*.

Finn, C., Yu, T., Fu, J., Abbeel, P., and Levine, S. (2017). Generalizing skills with semi-supervised reinforcement learning. *Submitted to Int'l Conference on Learning Representations*.

Florensa, C., Duan, Y., and Abbeel, P. (2017). Stochastic neural networks for hierarchical reinforcement learning. *Submitted to Int'l Conference on Learning Representations*.

Garcìa, J. and Fernàndez, F. (2015). A comprehensive survey on safe reinforcement learning. *The Journal of Machine Learning Research*, 16:1437–1480.

Garnelo, M., Arulkumaran, K., and Shanahan, M. (2016). Towards Deep Symbolic Reinforcement Learning. *ArXiv e-prints*.

Gelly, S., Schoenauer, M., Sebag, M., Teytaud, O., Kocsis, L., Silver, D., and Szepesvári, C. (2012). The grand challenge of computer go: Monte carlo tree search and extensions. *Communications of the ACM*, 55(3):106–113.

Ghavamzadeh, M., Mannor, S., Pineau, J., and Tamar, A. (2015). Bayesian reinforcement learning: a survey. *Foundations and Trends in Machine Learning*, 8(5-6):359–483.

Goldberg, Y. and Kosorok, M. R. (2012). Q-learning with censored data. *Annals of Statistics*, 40(1):529–560.

Goodfellow, I. (2017). NIPS 2016 Tutorial: Generative Adversarial Networks. *ArXiv e-prints*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., , and Bengio, Y. (2014). Generative adversarial nets. In *the Annual Conference on Neural Information Processing Systems (NIPS)*, page 2672?2680.

Graves, A., Wayne, G., and Danihelka, I. (2014). Neural Turing Machines. *ArXiv e-prints*.

Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Colmenarejo, S. G., Grefenstette, E., Ramalho, T., Agapiou, J., nech Badia, A. P., Hermann, K. M., Zwols, Y., Ostrovski, G., Cain, A., King, H., Summerfield, C., Blunsom, P., Kavukcuoglu, K., and Hassabis, D. (2016). Hybrid computing using a neural network with dynamic external memory. *Nature*, 538:471–476.

Gregor, K., Danihelka, I., Graves, A., Rezende, D., and Wierstra, D. (2015). Draw: A recurrent neural network for image generation. In *the International Conference on Machine Learning (ICML)*.

Gu, S., Holly, E., Lillicrap, T., and Levine, S. (2016a). Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. *ArXiv e-prints*.

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

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

He, D., Xia, Y., Qin, T., Wang, L., Yu, N., Liu, T.-Y., and Ma, W.-Y. (2016a). Dual learning for machine translation. In *the Annual Conference on Neural Information Processing Systems (NIPS)*.

He, F. S., Liu, Y., Schwing, A. G., and Peng, J. (2017). Learning to play in a day: Faster deep reinforcement learning by optimality tightening. *Submitted to Int'l Conference on Learning Representations*.

He, J., Chen, J., He, X., Gao, J., Li, L., Deng, L., and Ostendorf, M. (2016b). Deep reinforcement learning with a natural language action space. In *the Association for Computational Linguistics annual meeting (ACL)*.

He, J., Ostendorf, M., He, X., Chen, J., Gao, J., Li, L., and Deng, L. (2016c). Deep reinforcement learning with a combinatorial action space for predicting popular reddit threads. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

He, X. and Deng, L. (2013). Speech-centric information processing: An optimization-oriented approach. *Proceedings of the IEEE — Vol. 101, No. 5, May 2013*, 101(5):1116–1135.

Heaton, J. B., Polson, N. G., and Witte, J. H. (2016). Deep learning for finance: deep portfolios. *Applied Stochastic Models in Business and Industry*.

Heinrich, J. and Silver, D. (2016). Deep reinforcement learning from self-play in imperfect-information games. In *NIPS 2016 Deep Reinforcement Learning Workshop*.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., rahman Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., , and Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 82.

Hirschberg, J. and Manning, C. D. (2015). Advances in natural language processing. *Science*, 349(6245):261–266.

Ho, J. and Ermon, S. (2016). Generative adversarial imitation learning. In *the Annual Conference on Neural Information Processing Systems (NIPS)*.

Hull, J. C. (2014). *Options, Futures and Other Derivatives (9th edition)*. Prentice Hall.

Jaderberg, M., Mnih, V., Czarnecki, W., Schaul, T., Leibo, J. Z., Silver, D., and Kavukcuoglu, K. (2017). Reinforcement learning with unsupervised auxiliary tasks. *Submitted to Int'l Conference on Learning Representations*.

Jaderberg, M., Simonyan, K., Zisserman, A., and Kavukcuoglu, K. (2015). Spatial transformer networks. In *the Annual Conference on Neural Information Processing Systems (NIPS)*.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An Introduction to Statistical Learning with Applications in R*. Springer.

Jaques, N., Gu, S., Turner, R. E., and Eck, D. (2017). Tuning recurrent neural networks with reinforcement learning. *Submitted to Int'l Conference on Learning Representations*.

Johnson, M., Schuster, M., Le, Q. V., Krikun, M., Wu, Y., Chen, Z., Thorat, N., Viégas, F., Wattenberg, M., Corrado, G., Hughes, M., and Dean, J. (2016). Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. *ArXiv e-prints*.

Jordan, M. I. and Mitchell, T. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260.

Kadlec, R., Schmid, M., Bajgar, O., and Kleindienst, J. (2016). Text Understanding with the Attention Sum Reader Network. *ArXiv e-prints*.

Kaelbling, L. P., Littman, M. L., and Moore, A. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285.

Kaiser, L. and Bengio, S. (2016). Can active memory replace attention? In *the Annual Conference on Neural Information Processing Systems (NIPS)*.

Kempka, M., Wydmuch, M., Runc, G., Toczek, J., and Jaśkowski, W. (2016). ViZDoom: A Doom-based AI research platform for visual reinforcement learning. In *IEEE Conference on Computational Intelligence and Games*.

Kober, J., Bagnell, J. A., and Peters, J. (2013). Reinforcement learning in robotics: A survey. *International Journal of Robotics Research*, 32(11):1238–1278.

Kosorok, M. R. and Moodie, E. E. M. (2015). *Adaptive Treatment Strategies in Practice: Planning Trials and Analyzing Data for Personalized Medicine*. ASA-SIAM Series on Statistics and Applied Probability.

Krakovsky, M. (2016). Reinforcement renaissance. *Communications of the ACM*, 59(8):12–14.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *the Annual Conference on Neural Information Processing Systems (NIPS)*.

Kulkarni, T. D., Narasimhan, K. R., Saeedi, A., and Tenenbaum, J. B. (2016). Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *the Annual Conference on Neural Information Processing Systems (NIPS)*.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. (2016). Building Machines That Learn and Think Like People. *ArXiv e-prints*.

Lample, G. and Chaplot, D. S. (2016). Playing FPS Games with Deep Reinforcement Learning. *ArXiv e-prints*.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521:436–444.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. (2016a). End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17:1–40.

Levine, S., Pastor, P., Krizhevsky, A., and Quillen, D. (2016b). Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection. *ArXiv e-prints*.

Leyton-Brown, K. and Shoham, Y. (2008). *Essentials of Game Theory: A Concise, Multidisciplinary Introduction*. Morgan & Claypool Publishers.

Li, J., Monroe, W., and Jurafsky, D. (2016a). A Simple, Fast Diverse Decoding Algorithm for Neural Generation. *ArXiv e-prints*.

Li, J., Monroe, W., and Jurafsky, D. (2016b). Understanding Neural Networks through Representation Erasure. *ArXiv e-prints*.

Li, J., Monroe, W., and Jurafsky, D. (2017). Learning to Decode for Future Success. *ArXiv e-prints*.

Li, J., Monroe, W., Ritter, A., Galley, M., Gao, J., and Jurafsky, D. (2016c). Deep reinforcement learning for dialogue generation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Li, K. and Malik, J. (2017). Learning to optimize. *Submitted to Int'l Conference on Learning Representations*.

Li, L., Chu, W., Langford, J., and Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. In *the International World Wide Web Conference (WWW)*.

Li, X., Li, L., Gao, J., He, X., Chen, J., Deng, L., and He, J. (2015). Recurrent Reinforcement Learning: A Hybrid Approach. *ArXiv e-prints*.

Li, X., Lipton, Z. C., Dhingra, B., Li, L., Gao, J., and Chen, Y.-N. (2016d). A User Simulator for Task-Completion Dialogues. *ArXiv e-prints*.

Li, Y., Szepesvári, C., and Schuurmans, D. (2009). Learning exercise policies for American options. In *International Conference on Artificial Intelligence and Statistics (AISTATS09)*.

Liang, Y., Machado, M. C., Talvitie, E., and Bowling, M. (2016). State of the art control of atari games using shallow reinforcement learning. In *the International Conference on Autonomous Agents & Multiagent Systems (AAMAS)*.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2016). Continuous control with deep reinforcement learning. In *the International Conference on Learning Representations (ICLR)*.

Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3):293–321.

Lipton, Z. C., Gao, J., Li, L., Li, X., Ahmed, F., and Deng, L. (2016). Efficient Exploration for Dialogue Policy Learning with BBQ Networks & Replay Buffer Spiking. *ArXiv e-prints*.

Littman, M. L. (2015). Reinforcement learning improves behaviour from evaluative feedback. *Nature*, 521:445–451.

Lo, A. W. (2004). The Adaptive Markets Hypothesis: Market efficiency from an evolutionary perspective. *Journal of Portfolio Management*, 30:15–29.

Longstaff, F. A. and Schwartz, E. S. (2001). Valuing American options by simulation: a simple least-squares approach. *The Review of Financial Studies*, 14(1):113–147.

Luenberger, D. G. (1997). *Investment Science*. Oxford University Press.

Maurer, A., Pontil, M., and Romera-Paredes, B. (2016). The benefit of multitask representation learning. *The Journal of Machine Learning Research*, 17(81):1–32.

Mesnil, G., Dauphin, Y., Yao, K., Bengio, Y., Deng, L., He, X., Heck, L., Tur, G., Hakkani-Tür, D., Yu, D., and Zweig, G. (2015). Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539.

Mirowski, P., Pascanu, R., Viola, F., Soyer, H., Ballard, A., Banino, A., Denil, M., Goroshin, R., Sifre, L., Kavukcuoglu, K., Kumaran, D., and Hadsell, R. (2017). Learning to navigate in complex environments. *Submitted to Int'l Conference on Learning Representations*.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Harley, T., Lillicrap, T. P., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *the International Conference on Machine Learning (ICML)*.

Mnih, V., Heess, N., Graves, A., and Kavukcuoglu, K. (2014). Recurrent models of visual attention. In *the Annual Conference on Neural Information Processing Systems (NIPS)*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Mo, K., Li, S., Zhang, Y., Li, J., and Yang, Q. (2016). Personalizing a Dialogue System with Transfer Learning. *ArXiv e-prints*.

Moody, J. and Saffell, M. (2001). Learning to trade via direct reinforcement. *IEEE Transactions on Neural Networks*, 12(4):875–889.

Moravčík, M., Schmid, M., Burch, N., Lisý, V., Morrill, D., Bard, N., Davis, T., Waugh, K., Johanson, M., and Bowling, M. (2017). DeepStack: Expert-Level Artificial Intelligence in No-Limit Poker. *ArXiv e-prints*.

Munos, R., Stepleton, T., Harutyunyan, A., and Bellemare, M. G. (2016). Safe and efficient off-policy reinforcement learning. In *the Annual Conference on Neural Information Processing Systems (NIPS)*.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press.

Nachum, O., Norouzi, M., and Schuurmans, D. (2017). Improving policy gradient by exploring under-appreciated rewards. *Submitted to Int'l Conference on Learning Representations*.

Narasimhan, K., Kulkarni, T., and Barzilay, R. (2015). Language understanding for text-based games using deep reinforcement learning. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Narasimhan, K., Yala, A., and Barzilay, R. (2016). Improving information extraction by acquiring external evidence with reinforcement learning. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ng, A. and Russell, S. (2000). Algorithms for inverse reinforcement learning. In *the International Conference on Machine Learning (ICML)*.

Nogueira, R. and Cho, K. (2016). End-to-End Goal-Driven Web Navigation. *ArXiv e-prints*.

O'Donoghue, B., Munos, R., Kavukcuoglu, K., and Mnih, V. (2017). PGQ: Combining policy gradient and q-learning. *Submitted to Int'l Conference on Learning Representations*.

Oh, J., Guo, X., Lee, H., Lewis, R., and Singh, S. (2015). Action-conditional video prediction using deep networks in atari games. In *the Annual Conference on Neural Information Processing Systems (NIPS)*.

Oquab, M., Bottou, L., Laptev, I., and Sivic, J. (2015). Is object localization for free? ? weakly-supervised learning with convolutional neural networks. In *the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Osband, I., Blundell, C., Pritzel, A., and Roy, B. V. (2016). Deep exploration via bootstrapped DQN. In *the Annual Conference on Neural Information Processing Systems (NIPS)*.

Oudeyer, P.-Y., Gottlieb, J., and Lopes, M. (2016). Intrinsic motivation, curiosity and learning: theory and applications in educational technologies. *Progress in brain research, Elsevier*, 229:257–284.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345 – 1359.

Parisotto, E., Ba, J. L., and Salakhutdinov, R. (2016). Actor-mimic: Deep multitask and transfer reinforcement learning. In *the International Conference on Learning Representations (ICLR)*.

Pfau, D. and Vinyals, O. (2016). Connecting Generative Adversarial Networks and Actor-Critic Methods. *ArXiv e-prints*.

Powell, W. B. (2011). *Approximate Dynamic Programming: Solving the curses of dimensionality (2nd Edition)*. John Wiley and Sons.

Puterman, M. L. (2005). *Markov decision processes : discrete stochastic dynamic programming*. Wiley-Interscience.

Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. (2016). Sequence level training with recurrent neural networks. In *the International Conference on Learning Representations (ICLR)*.

Reed, S. and de Freitas, N. (2016). Neural programmer-interpreters. In *the International Conference on Learning Representations (ICLR)*.

Russell, S. and Norvig, P. (2009). *Artificial Intelligence: A Modern Approach (3rd edition)*. Pearson.

Sandholm, T. (2015). Solving imperfect-information games. *Science*, 347(6218):122–123.

Saria, S. (2014). A $3 trillion challenge to computational scientists: Transforming healthcare delivery. *IEEE Intelligent Systems*, 29(4):82–87.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2016). Prioritized experience replay. In *the International Conference on Learning Representations (ICLR)*.

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

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., and Abbeel, P. (2015). Trust region policy optimization. In *the International Conference on Machine Learning (ICML)*.

Schuurmans, D. and Zinkevich, M. (2016). Deep learning games. In *the Annual Conference on Neural Information Processing Systems (NIPS)*.

Shah, P., Hakkani-Tür, D., and Heck, L. (2016). Interactive reinforcement learning for task-oriented dialogue management. In *NIPS 2016 Deep Learning for Action and Interaction Workshop*.

Shen, Y., Huang, P.-S., Gao, J., and Chen, W. (2016). ReasoNet: Learning to Stop Reading in Machine Comprehension. *ArXiv e-prints*.

Shoham, Y., Powers, R., and Grenager, T. (2003). *Multi-Agent Reinforcement Learning: a critical survey*. Web manuscript.

Shortreed, S. M., Laber, E., Lizotte, D. J., Stroup, T. S., Pineau, J., and Murphy, S. A. (2011). Informing sequential clinical decision-making through reinforcement learning: an empirical study. *Machine Learning*, 84:109–136.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). Deterministic policy gradient algorithms. In *the International Conference on Machine Learning (ICML)*.

Silver, D., Newnham, L., Barker, D., Weller, S., and McFall, J. (2013). Concurrent reinforcement learning from customer interactions. In *the International Conference on Machine Learning (ICML)*.

Silver, D., van Hasselt, H., Hessel, M., Schaul, T., Guez, A., Harley, T., Dulac-Arnold, G., Reichert, D., Rabinowitz, N., Barreto, A., and Degris, T. (2017). The predictron: End-to-end learning and planning. *Submitted to Int'l Conference on Learning Representations*.

Stadie, B. C., Levine, S., and Abbeel, P. (2015). Incentivizing exploration in reinforcement learning with deep predictive models. In *NIPS 2015 Deep Reinforcement Learning Workshop*.

Su, P.-H., Gasic, M., Mrksic, N., Rojas-Barahona, L., Ultes, S., Vandyke, D., Wen, T.-H., and Young, S. (2016a). Continuously Learning Neural Dialogue Management. *ArXiv e-prints*.

Su, P.-H., Gasˇić, M., Mrksˇić, N., Rojas-Barahona, L., Ultes, S., Vandyke, D., Wen, T.-H., and Young, S. (2016b). On-line active reward learning for policy optimisation in spoken dialogue systems. In *the Association for Computational Linguistics annual meeting (ACL)*.

Sukhbaatar, S., Weston, J., and Fergus, R. (2015). End-to-end memory networks. In *the Annual Conference on Neural Information Processing Systems (NIPS)*.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *the Annual Conference on Neural Information Processing Systems (NIPS)*.

Sutton, R. S. and Barto, A. G. (2017). *Reinforcement Learning: An Introduction (2nd Edition, in preparation)*. MIT Press.

Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., White, A., and Precup, D. (2011). Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction, , proc. of 10th. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Synnaeve, G., Nardelli, N., Auvolat, A., Chintala, S., Lacroix, T., Lin, Z., Richoux, F., and Usunier, N. (2016). TorchCraft: a Library for Machine Learning Research on Real-Time Strategy Games. *ArXiv e-prints*.

Szepesvári, C. (2010). *Algorithms for Reinforcement Learning*. Morgan & Claypool.

Tamar, A., Wu, Y., Thomas, G., Levine, S., and Abbeel, P. (2016). Value iteration networks. In *the Annual Conference on Neural Information Processing Systems (NIPS)*.

Taylor, M. E. and Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10:1633–1685.

Tesauro, G. (1994). TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6(2):215–219.

Tessler, C., Givony, S., Zahavy, T., Mankowitz, D. J., and Mannor, S. (2017). A deep hierarchical approach to lifelong learning in minecraft. In *the AAAI Conference on Artificial Intelligence (AAAI)*.

Theocharous, G., Thomas, P. S., and Ghavamzadeh, M. (2015). Personalized ad recommendation systems for life-time value optimization with guarantees. In *the International Joint Conference on Artificial Intelligence (IJCAI)*.

Tran, D., Hoffman, M. D., Saurous, R. A., Brevdo, E., Murphy, K., and Blei, D. M. (2017). Deep Probabilistic Programming. *ArXiv e-prints*.

Trischler, A., Ye, Z., Yuan, X., and Suleman, K. (2016). Natural language comprehension with the epireader. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Tsitsiklis, J. N. and Van Roy, B. (1997). An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690.

Tsitsiklis, J. N. and Van Roy, B. (2001). Regression methods for pricing complex American-style options. *IEEE Transactions on Neural Networks*, 12(4):694–703.

Usunier, N., Synnaeve, G., Lin, Z., and Chintala, S. (2016). Episodic Exploration for Deep Deterministic Policies: An Application to StarCraft Micromanagement Tasks. *ArXiv e-prints*.

van Hasselt, H., Guez, A., , and Silver, D. (2016a). Deep reinforcement learning with double q-learning. In *the AAAI Conference on Artificial Intelligence (AAAI)*.

van Hasselt, H., Guez, A., Hessel, M., Mnih, V., and Silver, D. (2016b). Learning values across many orders of magnitude. In *the Annual Conference on Neural Information Processing Systems (NIPS)*.

Vezhnevets, A. S., Mnih, V., Agapiou, J., Osindero, S., Graves, A., Vinyals, O., and Kavukcuoglu, K. (2016). Strategic attentive writer for learning macro-actions. In *the Annual Conference on Neural Information Processing Systems (NIPS)*.

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., and Wierstra, D. (2016). Matching networks for one shot learning. In *the Annual Conference on Neural Information Processing Systems (NIPS)*.

Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., Blundell, C., Kumaran, D., and Botvinick, M. (2016a). Learning to reinforcement learn. *arXiv:1611.05763v1*.

Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M., and de Freitas, N. (2016b). Dueling network architectures for deep reinforcement learning. In *the International Conference on Machine Learning (ICML)*.

Weiss, K., Khoshgoftaar, T. M., and Wang, D. (2016). A survey of transfer learning. *Journal of Big Data*, 3(9).

Wen, T.-H., Gasic, M., Mrksic, N., Su, P.-H., Vandyke, D., and Young, S. (2015a). Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Wen, Z., O'Neill, D., and Maei, H. (2015b). Optimal demand response using device-based reinforcement learning. *IEEE Transactions on Smart Grid*, 6(5):2312–2324.

Weston, J., Chopra, S., and Bordes, A. (2015). Memory networks. In *the International Conference on Learning Representations (ICLR)*.

Williams, J. D. and Zweig, G. (2016). End-to-end LSTM-based dialog control optimized with supervised and reinforcement learning. *ArXiv e-prints*.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *ArXiv e-prints*.

Wu, Y. and Tian, Y. (2017). Training agent for first-person shooter game with actor-critic curriculum learning. *Submitted to Int'l Conference on Learning Representations*.

Xu, K., Ba, J. L., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R. S., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *the International Conference on Machine Learning (ICML)*.

Yahya, A., Li, A., Kalakrishnan, M., Chebotar, Y., and Levine, S. (2016). Collective robot reinforcement learning with distributed asynchronous guided policy search. *ArXiv e-prints*.

Yang, X., Chen, Y.-N., Hakkani-Tur, D., Crook, P., Li, X., Gao, J., and Deng, L. (2016). End-to-End Joint Learning of Natural Language Understanding and Dialogue Manager. *ArXiv e-prints*.

Yang, Z., He, X., Gao, J., Deng, L., and Smola, A. (2015). Stacked Attention Networks for Image Question Answering. *ArXiv e-prints*.

Yogatama, D., Blunsom, P., Dyer, C., Grefenstette, E., and Ling, W. (2017). Learning to compose words into sentences with reinforcement learning. *Submitted to Int'l Conference on Learning Representations*.

Young, S., Gašić, M., Thomson, B., and Williams, J. D. (2013). POMDP-based statistical spoken dialogue systems: a review. *PROC IEEE*, 101(5):1160–1179.

Yu, L., Zhang, W., Wang, J., and Yu, Y. (2017). Seqgan: Sequence generative adversarial nets with policy gradient. In *the AAAI Conference on Artificial Intelligence (AAAI)*.

Yu, Y.-L., Li, Y., Szepesvári, C., and Schuurmans, D. (2009). A general projection property for distribution families. In *the Annual Conference on Neural Information Processing Systems (NIPS)*.

Zagoruyko, S. and Komodakis, N. (2017). Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *Submitted to Int'l Conference on Learning Representations*.

Zaremba, W. and Sutskever, I. (2015). Reinforcement Learning Neural Turing Machines - Revised. *ArXiv e-prints*.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2017). Understanding deep learning requires rethinking generalization. *Submitted to Int'l Conference on Learning Representations*.

Zhao, T. and Eskenazi, M. (2016). Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. In *the Annual SIGdial Meeting on Discourse and Dialogue (SIGDIAL)*.

Zhu, X. and Goldberg, A. B. (2009). *Introduction to semi-supervised learning*. Morgan & Claypool.

Zhu, Y., Mottaghi, R., Kolve, E., Lim, J. J., Gupta, A., Li, F.-F., and Farhadi, A. (2016). Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning. *ArXiv e-prints*.

Zinkevich, M. (2017). *Rules of Machine Learning: Best Practices for ML Engineering*. http://martin.zinkevich.org/rules_of_ml/rules_of_ml.pdf.

Zoph, B. and Le, Q. V. (2017). Neural architecture search with reinforcement learning. *Submitted to Int'l Conference on Learning Representations*.