
Deep Reinforcement Learning in Robotics and Dialog Systems

Rui Zhao



München 2020

Deep Reinforcement Learning in Robotics and Dialog Systems

Rui Zhao

Dissertation

an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig-Maximilians-Universität

München

vorgelegt von

Rui Zhao

aus Shenyang, Liaoning, China VR

München, den 19.06.2020

Erstgutachter: Prof. Dr. Volker Tresp

Zweitgutachter: Prof. Dr. Sepp Hochreiter

Drittgutachter: Prof. Dr. Kurt Driessens

Tag der mündlichen Prüfung: 10 September 2020

Eidesstattliche Versicherung

(Siehe Promotionsordnung vom 12.07.11, § 8, Abs. 2 Pkt. .5.)

Hiermit erkläre ich an Eidesstatt, dass die Dissertation von mir selbstständig, ohne unerlaubte Beihilfe angefertigt ist.

Zhao, Rui

Name, Vorname

München, 19 June 2020

Ort, Datum

Unterschrift Doktorand/in

Formular 3.2

Contents

Abstract	xi
Zusammenfassung	xii
Acknowledgment	xiii
Publications and Declaration of Authorship	xiv
1 Introduction	1
1.1 Reinforcement Learning	1
1.1.1 Reinforcement Learning Terminology	3
1.1.2 Reinforcement Learning Methods	4
1.2 Deep Reinforcement Learning of Dialogues	6
1.2.1 Deep Neural Networks	6
1.2.2 Policy Gradient	7
1.2.3 Tempered Policy Gradient	8
1.2.4 Positive Memory Retention	10
1.3 Deep Reinforcement Learning for Robotics	13
1.3.1 Deep Deterministic Policy Gradient	13
1.3.2 Hindsight Experience Replay	15
1.3.3 Energy-based Hindsight Experience Prioritization	15
1.3.4 Curiosity-driven Experience Prioritization	17
1.3.5 Maximum Goal Entropy Reinforcement Learning	19
1.3.6 Mutual Information-based State-Control	22

2	Learning Goal-Oriented Visual Dialog via Tempered Policy Gradient	27
2.1	Introduction	28
2.2	Preliminaries	29
2.3	Tempered Policy Gradient	29
2.3.1	Exploration and Exploitation	29
2.3.2	Temperature Sampling	29
2.3.3	Tempered Policy Gradient Methods	29
2.4	Guesswhat?! Game	30
2.4.1	Models and Pretraining	30
2.4.2	Reinforcement Learning	31
2.5	Experiment	32
2.5.1	Pretraining	32
2.5.2	Reinforcement Learning	32
2.6	Conclusion	33
2.7	References	34
3	Efficient Dialog Policy Learning via Positive Memory Retention	36
3.1	Introduction	37
3.2	Background	38
3.2.1	Recurrent Language Models	38
3.2.2	Markov Decision Process and Policy Gradient	38
3.2.3	Importance Sampling	38
3.3	Positive Memory Retention	39
3.3.1	Positive Memory Matters	39
3.3.2	Policy Gradient with Importance Sampling	39
3.3.3	Bounded Importance Weight Proposal	39
3.3.4	Probability Updating	39
3.3.5	Policy Search via Early Stopping	39
3.3.6	Complete Training Algorithm	40
3.4	Experiments	40

3.4.1	MNIST GuessNumber Dataset	40
3.4.2	GuessWhat?! Game	41
3.5	Related Work	42
3.5.1	Goal-Oriented Dialogs	42
3.5.2	Sample-Efficient RL	42
3.5.3	Memory Retention	42
3.6	Conclusion	42
3.7	References	43
4	Energy-Based Hindsight Experience Prioritization	45
4.1	Introduction	46
4.2	Background	47
4.2.1	Markov Decision Process	47
4.2.2	Deep Deterministic Policy Gradient	47
4.2.3	Hindsight Experience Replay	48
4.2.4	Work-Energy Principle	48
4.3	Method	48
4.3.1	Motivation	48
4.3.2	Trajectory Energy Function	49
4.3.3	Energy-Based Prioritization	50
4.3.4	Comparison with Prioritized Experience Replay	51
4.4	Experiments	51
4.4.1	Environments	51
4.4.2	Performance	51
4.4.3	Sample-Efficiency	52
4.4.4	Insights	52
4.5	Related Work	53
4.6	Conclusion	53
4.7	References	54

5	Curiosity-Driven Experience Prioritization via Density Estimation	56
5.1	Introduction	57
5.2	Background	58
5.2.1	Markov Decision Process	58
5.2.2	Deep Deterministic Policy Gradient	58
5.2.3	Hindsight Experience Replay	59
5.2.4	Gaussian Mixture Model	59
5.3	Method	59
5.3.1	Motivation	59
5.3.2	Curiosity-Driven Prioritization	60
5.3.3	Comparison with Prioritized Experience Replay	61
5.4	Experiments	61
5.4.1	Environments	61
5.4.2	Performance	62
5.4.3	Sample-Efficiency	63
5.4.4	Insights	64
5.5	Related Work	64
5.6	Conclusion	64
5.7	References	64
6	Maximum Entropy-Regularized Multi-Goal Reinforcement Learning	68
6.1	Introduction	69
6.2	Preliminary	70
6.2.1	Settings	70
6.2.2	Reinforcement Learning	70
6.2.3	Weighted Entropy	71
6.3	Method	71
6.3.1	Multi-Goal RL	71
6.3.2	Maximum Entropy-Regularized Multi-Goal RL	71

6.3.3	Surrogate Objective	72
6.3.4	Prioritized Sampling	72
6.3.5	Estimation of Distribution	72
6.3.6	Maximum Entropy-Based Prioritization	73
6.4	Experiments	73
6.4.1	Performance	74
6.4.2	Sample-Efficiency	75
6.4.3	Goal Entropy	75
6.5	Related Work	75
6.6	Conclusion	76
6.7	References	77
6.8	Appendix	79
6.8.1	Proof of Theorem 1	79
6.8.2	Proof of Theorem 2	80
6.8.3	Insights	82
7	Mutual Information-based State-Control for Intrinsically Motivated Reinforcement Learning	83
7.1	Introduction	84
7.2	Preliminary	85
7.2.1	Environments	85
7.2.2	Goal States and Controllable States	85
7.2.3	Reinforcement Learning Settings	85
7.2.4	Notations	85
7.3	Method	85
7.3.1	Mutual Information Reward Function	85
7.3.2	Efficient Learning State-Control	86
7.3.3	Implementation	87
7.3.4	Complete Algorithm	87
7.3.5	MISC Variants with Task Rewards	87
7.3.6	Skill Discovery with MISC and DIAYN	87

7.4	Experiments	87
7.4.1	Analysis of the Learned Behaviors	88
7.4.2	Accelerating Learning with MISC	88
7.4.3	Transfer Learning with MISC	89
7.4.4	Insights and More	90
7.4.5	Summary	91
7.5	Related Work	91
7.6	Conclusion	91
7.7	References	92
7.8	Appendix	94
7.8.1	Connection to Empowerment	94
7.8.2	Learned Control Behaviors without Supervision	94
7.8.3	Experimental Details	94

Abstract

The capability of learning from rewards obtained by interactions with the environment is one of the key features of human intelligence. The goal of reinforcement learning is to provide similar abilities to artificial agents. This dissertation explores state-of-the-art reinforcement learning methods and extends them to be applicable to real-world dialog settings and sparse-reward robotic tasks. The thesis starts by introducing the paradigm of reinforcement learning and classical algorithmic learning approaches. Subsequently, we discuss how a dialog agent can be trained efficiently to complete tasks by means of deep reinforcement learning in the visually-ground dialog settings. Furthermore, we show that importance sampling helps to increase sample-efficiency in real-world dialog systems. The second part of the thesis focuses on developing core deep reinforcement learning approaches and evaluates them in the continuous control domain. We derive the trajectory energy function and use it for improving the learning-efficiency of a robotic agent. We also developed a curiosity-based experience prioritization strategy, which improves the agent’s learning efficiency by a factor of two. Subsequently, we derive the mathematical link between the curiosity-driven prioritization framework and the maximum entropy principle. Finally, we consider the more general case, when the external reward signal is unavailable, and we investigate the topic of intrinsically motivated reinforcement learning and propose a novel mutual information-based state-control method. By using this novel method, the agent is able to learn control behaviors and discover a diverse set of control skills without hand-engineered task rewards.

Zusammenfassung

Die Fähigkeit, die durch Interaktionen mit der Umwelt erzielt werden, ist eines der Hauptmerkmale der menschlichen Intelligenz. Das Ziel des Reinforcement Learnings besteht darin, künstlichen Intelligenz ähnliche Fähigkeiten zu verleihen. In dieser Dissertation werden die neuesten Methoden von Reinforcement Learning untersucht und erweitert, um sie auf reale Dialogeinstellungen und Roboteraufgaben mit geringer Reward anzuwenden. Die Arbeit beginnt mit der Einführung des Paradigmas des Reinforcement Learnings und klassischer algorithmischer Lernansätze. Anschließend diskutieren wir, wie ein Dialogagent effizient geschult werden kann. Darüber hinaus zeigen wir, dass wichtige Stichproben dazu beitragen, die Lerneffizienz in realen Dialogsystemen zu steigern. Der zweite Teil der Arbeit konzentriert sich auf die Entwicklung zentraler Lernansätze von Reinforcement Learning und bewertet diese im Bereich der kontinuierlichen Kontrolle. Wir leiten die Trajektorienenergiefunktion ab und verwenden sie zur Verbesserung der Lerneffizienz eines Roboteragenten. Wir haben auch eine neugierige Strategie zur Priorisierung von Erfahrungen entwickelt, die die Lerneffizienz des Agenten verbessert. Anschließend leiten wir den mathematischen Zusammenhang zwischen dem neugierigen Priorisierungsrahmen und dem Maximum-Entropie-Prinzip ab. Schließlich betrachten wir den allgemeineren Fall, in dem das externe Rewardssignal nicht verfügbar ist, und untersuchen das Thema des intrinsisch motivierten Reinforcement Learning und schlagen eine neuartige Methode zur Mutual-Informationsbasierten Zustandskontrolle vor. Mithilfe dieser neuartigen Methode kann der Agent Kontrollverhalten erlernen und eine Vielzahl von Kontrollfähigkeiten ohne handgefertigte Rewardsfunktionen entdecken.

Acknowledgment

This Ph.D. is a joint program between the Ludwig Maximilian University of Munich and Siemens AG. I have many people to thank. First of all, I want to thank my supervisor, Prof. Volker Tresp for offering me this opportunity of pursuing a Ph.D. in Machine Learning. Prof. Tresp has been a very kind and supportive advisor in my Ph.D. career, who has given me valuable feedback on each research draft and some enlightening advice through the discussions. I am very honored that Prof. Dr. Sepp Hochreiter and Prof. Dr. Kurt Driessens agreed to be external examiners of my thesis. I also want to thank Dr. Ulli Waltinger for being a supportive manager and organizing the team-building events. In general, I want to thank Siemens for providing the Ph.D. position and the infrastructure for completing a Ph.D. I also want to thank my colleges. For example, Yinchong has been a mentor, who guided me at the beginning of my Ph.D. Denis has organized the journal club for us to exchange general research ideas. There were interesting discussions with Stephan Baier about recurrent neural networks and variational auto-encoders. Stefan Depeweg and I had insightful discussions about reinforcement learning and Bayesian neural networks. Jindong, Zhiliang, Yushan, and I had chatted over lunch about our daily Ph.D. lives. I also want to thank Xudong for assisting me before the ICML 2019 deadline. Furthermore, my internship at Horizon Robotics in California, USA was a great and rewarding experience. Thank Dr. Wei Xu for hosting me as a research intern and the colleges there for helping me to settle down in the US for six months. Most importantly, I want to thank my family and friends for being there with me through ups and downs.

Publications and Declaration of Authorship

- **Rui Zhao** and Volker Tresp. Learning goal-oriented visual dialog via tempered policy gradient. Published in *Proceedings of the IEEE Spoken Language Technology (SLT)*, Athens, Greece, 18-21 Dec 2018. pages 868–875. IEEE

I conceived of the original research contributions and performed all implementations and evaluations. I wrote the initial draft of the manuscript and did most of the subsequent corrections. I discussed this work with my supervisor, Prof. Volker Tresp, who assisted me in improving the manuscript. This published work serves as Chapter 2.

- **Rui Zhao** and Volker Tresp. Efficient dialog policy learning via positive memory retention. Published in *Proceedings of the IEEE Spoken Language Technology (SLT)*, Athens, Greece, 18-21 Dec 2018. pages 823–830. IEEE

I conceived of the original research contributions and performed all implementations and evaluations. I wrote the initial draft of the manuscript and did most of the subsequent corrections. I discussed this work with my supervisor, Prof. Volker Tresp, who assisted me in improving the manuscript. This published work serves as Chapter 3.

- **Rui Zhao** and Volker Tresp. Energy-based hindsight experience prioritization, Published in *Proceedings of the 2nd Conference on Robot Learning (CoRL)*, volume 87 of the *Proceedings of Machine Learning Research*

(PMLR), pages 113–122, Zürich, Switzerland, 29-31 Oct 2018, (Oral presentation, 7% acceptance rate).

I conceived of the original research contributions and performed all implementations and evaluations. I wrote the initial draft of the manuscript and did most of the subsequent corrections. I discussed this work with my supervisor, Prof. Volker Tresp, who assisted me in improving the manuscript. This published work serves as Chapter 4.

- **Rui Zhao** and Volker Tresp. Curiosity-driven experience prioritization via density estimation. Presented in *Proceedings of the 32nd Conference on Neural Information Processing Systems Deep Reinforcement Learning Workshop* (NeurIPS Deep RL workshop), Montreal, Canada, 3-8 Dec 2018.

I conceived of the original research contributions and performed all implementations and evaluations. I wrote the initial draft of the manuscript and did most of the subsequent corrections. I discussed this work with my supervisor, Prof. Volker Tresp, who assisted me in improving the manuscript. This work serves as Chapter 5.

- **Rui Zhao**, Xudong Sun, and Volker Tresp. Maximum entropy-regularized multi-goal reinforcement learning. Published in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of the *Proceedings of Machine Learning Research (PMLR)*, pages 7553-7562, Long Beach, California, USA, 09-15 Jun 2019.

I conceived of the original research contributions and performed all implementations and evaluations. I wrote the initial draft of the manuscript and did most of the subsequent corrections. I discussed this work with the co-author Xudong Sun and my supervisor, Prof. Volker Tresp. Xudong Sun contributed to the proof of the theorems. Prof. Volker Tresp assisted me in improving the manuscript. This published work serves as Chapter 6.

- **Rui Zhao**, Volker Tresp, and Wei Xu. Mutual information-based state-control for intrinsically motivated reinforcement learning. Available at *arXiv preprint arXiv:2002.01963v1*, Under Review at a Conference, 2020

I conceived of the original research contributions and performed all implementations and evaluations. I wrote the initial draft of the manuscript and did most of the subsequent corrections. I discussed this work with my supervisors, Dr. Wei Xu (at Horizon Robotics) and Prof. Volker Tresp, who assisted me in improving the manuscript. This work serves as Chapter 7.

Chapter 1

Introduction

1.1 Reinforcement Learning

Reinforcement learning is about learning by interacting with the environment. This learning scheme is very similar to how we as humans learn in our daily lives. For example, in our childhood, we interacted with the environment with our hands by playing with toys and eating food. The interactions include sequences of actions, observations, and consequences. We can learn how to interact with the world without an explicit teacher, only based on the rich information about actions and their effects. Through these experiences, we also learn to achieve goals. Learning from interactions is a common idea underlying many intelligent organisms.

In this dissertation we explore this computation approach of learning from interactions, mapping from observations to actions, and maximizing a numerical reward signal. This paradigm is reinforcement learning. In reinforcement learning, the agent is not told which actions to take but must discover the actions that lead to the maximum accumulated future reward by itself. In challenging cases, the rewards are delayed and sparse. This is also the setting that we consider in this dissertation, Chapter 2-7, including the real-world dialog settings and the robotic tasks in simulations. A reinforcement learning problem is often formulated as a Markov decision process. The Markov decision process includes three

important aspects—state, action, and goal—in its simplest form. Any approach that solves such problems can be considered as a reinforcement learning method.

Reinforcement learning is different from supervised learning. In supervised learning, a training set with the labeled data is given. Each data point describes the situation and the label can be seen as the action that should be taken given the observation. The objective of supervised learning is to generalize the agent’s responses when the situation is not presented in the dataset. Through supervised learning, the agent learns to imitate expert behaviors but this doesn’t give an optimal solution in many cases. Therefore, the agent must learn from its own interactions with the environment and search for the optimal policy via the reward signal.

Reinforcement learning is also different from unsupervised learning. In unsupervised learning, the objective is to discover the underlying structure in the unlabeled data. In comparison, reinforcement learning attempts to find an optimal policy that maximizes a reward signal instead of finding the hidden structure in the data. However, understanding the structure of the collected data will certainly help a reinforcement learning agent to achieve high rewards. We consider reinforcement learning as a third machine learning paradigm, along with supervised learning and unsupervised learning.

One of the challenges in reinforcement learning is to find a balance between exploration and exploitation. An agent needs to choose the optimal actions which have led to high rewards. Concurrently, it also needs to try new actions in order to discover better policies. The dilemma is that neither exploration nor exploitation alone will solve the tasks. The agent must try different actions and progressively favor these actions that lead to high rewards. Normally, in a stochastic environment, an action has to be tried many times until the agent has a reliable estimation of the expected accumulated future rewards. The exploration and exploitation trade-off has been studied for many years in the reinforcement learning literature. In Chapter 2, we study and explore such trade-off in a real-world dialog setting.

Another key challenge in reinforcement learning is how to learn efficiently from

the collected experience. Depending on how the experience is collected, the reinforcement learning methods can be classified as on-policy and off-policy methods. On-policy means that the behavior policy used for collecting the experience is the same as the policy that we aim to optimize. On the contrary, off-policy means that the behavior policy used for exploration is different from the policy being optimized. To make the on-policy reinforcement learning methods off-policy and sample-efficient, we can use importance sampling to correct the bias introduced by the behavior policy. We explore this direction in the dialog settings in Chapter 3. Furthermore, to make the off-policy methods more sample-efficient, we can use various prioritized sampling frameworks. We develop and evaluate a variety of sampling strategies in Chapter 4, Chapter 5, and Chapter 6.

Beyond the traditional reinforcement learning paradigm of learning from an external reward signal, the topic of intrinsically motivated reinforcement learning is even more challenging and interesting. This learning paradigm is more closer to how the human learns. For example, an infant learns to explore and interact with the world only through its curiosity. These collected experiences help the infant to build up its knowledge base about the world and accelerate its learning process for achieving goals in future tasks. When there is no external reward signal, an intrinsic reward can guide the agent to explore the environment and learn useful interaction behaviors or skills. Later, when the task is defined, the agent should be able to learn to solve the down-stream tasks more efficiently using the learned behaviors or skills. In Chapter 7, we research the intrinsic motivation for the robotic agents in manipulation and navigation tasks.

1.1.1 Reinforcement Learning Terminology

Similar to the term *agent* and *environment*, there are some other terms in reinforcement learning, such as a *policy*, a *reward signal*, and a *value function*.

A *policy* means the agent's behavior at a given state. More specifically, a policy is a mapping from a specific state or observation to the action. The policy can be represented as a lookup table or a function. If the function is a deep neural

network, then it is called *deep reinforcement learning*, which takes advantage of using deep learning techniques. The policy is a key concept in reinforcement learning, as it alone can define the agent’s behavior given a state. The policy can be deterministic or stochastic.

A *reward signal* defines what the goal the agent should achieve. At each step, the agent receives a scalar feedback signal from the environment, which is the reward. For example, if the agent achieves the goal, then it receives a positive reward, otherwise, it receives a negative reward signal. The objective of the agent is to maximize the total reward over the trajectory. The reward defines what events are good and what are bad, which is analog to the pleasure and pain sensations in the human brain.

A similar but different concept is *return*, which means the expected accumulated reward over future. Intuitively, return represents how good it looks like in the long run. Whereas the reward defines how good it is at the current step. A *value function* predicts the *value* of a state, which is the total reward that the agent expected to collect over the future, starting from the current state. The value of the state takes into account the follow-up states and their corresponding rewards. For example, if the immediate reward of the current state is low, but the follow-up state has a high reward, then the value of the current state could still be high.

1.1.2 Reinforcement Learning Methods

An important reinforcement learning method based on the *value functions* is temporal-difference learning. Imagine a 2D-maze environment, the goal for the agent is to navigate from the left-bottom corner to the right-upper corner. During learning, the agent constantly updates the value estimation for each state. The updating mechanism “back-ups” the value of the next state after a greedy action to the current state. In another word, the value of the current state is updated to be closer to the value of the later state. This is done by adding a fraction of

the value difference between the current and the next state, mathematically,

$$V(s) \leftarrow V(s) + \alpha[r + V(s') - V(s)], \quad (1.1)$$

where we use s to denote the state and s' to denote the next state. $V(s)$ represents the value function of state s and α represents the step-size parameter, which is a small positive fraction. This is an example of the temporal-difference learning methods, which uses the value difference $V(s') - V(s)$ to update the value function.

Another example of the temporal-difference learning method is Q-learning. A Q-function $Q(s, a)$ represents how good it is at state s and selecting the action a . The Q-function can also be updated through temporal-difference learning, as follows:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \max_a Q(s', a) - Q(s, a)], \quad (1.2)$$

where s denotes state; a denotes action; r denotes the immediate reward at the current step.

Another class of reinforcement learning method is called policy gradient, which learns directly a mapping from states to actions instead of learning a value function or a Q-function. An example of policy gradient methods, REINFORCE or Monte Carlo policy gradient, is introduced in this Chapter, Section 1.2.2. Policy gradient and value or Q-functions can be combined, which results in a new class of reinforcement learning methods, namely Actor-Critic methods. The Actor means the policy and the Critic means the value function or the Q-function. For instance, the actor is trained to maximize the value estimation from the critic. An example of the actor-critic methods, Deep Deterministic Policy Gradient (DDPG), is introduced in this Chapter, Section 1.3.1.

In the following sections, I will introduce deep reinforcement learning methods that I used in dialog systems and robotic tasks, including some baseline methods and the method that I newly developed.

1.2 Deep Reinforcement Learning of Dialogues

A cognitive dialog system consists of a visual perception module and a language processing module [Das et al., 2017a, de Vries et al., 2017]. The visual perception module is represented by a Convolution Neural Network (CNN) [LeCun et al., 1998, Simonyan and Zisserman, 2014] and the language processing module is modeled with a Recurrent Neural Network (RNN) [Hochreiter and Schmidhuber, 1997, Graves et al., 2013].

1.2.1 Deep Neural Networks

A CNN simulates human visual system and extracts a hierarchy of features from the image input [Zeiler and Fergus, 2014]. An RNN mimics the human language competence and models the inter-correlations among phrases in a sentence [Karpathy et al., 2015]. With the CNN and RNN, a dialog agent can process image and text inputs and results into two fixed-length latent representations [Das et al., 2017a, de Vries et al., 2017]. A full-connected layer combines these two kinds of representations and transforms them into a fixed-length latent representation. With deep neural networks, raw image and text inputs can be encoded into more structured latent representations.

Based on RNNs, a Memory Network utilizes multiple RNN heads to process the dialog sequence in parallel [Weston et al., 2014, Sukhbaatar et al., 2015, Tresp et al., 2015]. Each head processes a different question-answer pair. The processed the information is stored into a memory bank. An attention mechanism helps to query the memory bank for a given key [Mnih et al., 2014, Xu et al., 2015]. An attention mask is generated based on the input key and then acts over the memory bank and emphasizes on the most relevant information.

An RNN decoder is trained to generate a sentence word by word via unrolling over time, given a context representation [Sutskever et al., 2014, Das et al., 2017a]. The context information comes from the raw images and/or the history dialogs, which are processed by deep neural nets. The RNN decoder can be trained with a maximum likelihood objective to match the distribution of a given dataset [Mur-

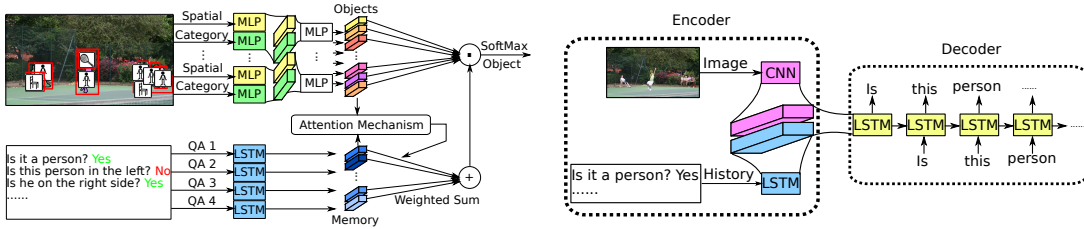


Figure 1.1: **Deep neural nets-based dialog models:** Here are two examples of the dialog models used in Chapter 2 “Learning goal-oriented visual dialog via tempered policy gradient”.

phy, 2012]. The RNN decoder can also be trained via reinforcement learning, such as policy gradient, to shape the policy for maximizing the future accumulated rewards [Strub et al., 2017, Sutton and Barto, 2018]. An example of the dialog system, including the memory network [Weston et al., 2014], the attention mechanism [Xu et al., 2015], and the sequence-to-sequence encoder-decoder model [Sutskever et al., 2014] is shown in Figure 1.1.

1.2.2 Policy Gradient

Compared to imitating human utterances, directly maximizing the task complete rate with reinforcement learning has a better performance [Strub et al., 2017, Zhao and Tresp, 2018d]. A standard policy gradient method used widely in the dialog setting is the Monte Carlo Policy Gradient (REINFORCE) [Williams, 1992, Das et al., 2017a, Strub et al., 2017, Zhao and Tresp, 2018d]. It is a weight update rule that makes weight adjustments to the agent in the direction of the estimated gradient of expected reward signals $E(r | w)$. The REINFORCE updating rule is:

$$\Delta w_i = \alpha_i (r - b_i) \partial \ln f / \partial w_i, \quad (1.3)$$

where Δw_i denotes the weight adjustment of weight w_i , α_i is a nonnegative learning rate factor, and b_i is a reinforcement baseline [Williams, 1992].

The particular task we are addressing in Chapter 2 “Learning goal-oriented visual dialog via tempered policy gradient” and Chapter 3 “Efficient Dialog Policy Learning via Positive Memory Retention” is to efficiently train goal-oriented

dialogue agents, which generate words per time-step to formulate complete sentences in order to achieve a goal.

1.2.3 Tempered Policy Gradient

In this section, I introduce the work in paper “Learning Goal-Oriented Visual Dialog via Tempered Policy Gradient” [Zhao and Tresp, 2018d], included in Chapter 2. The previous works on visual dialog policy learning use behavior clone [Strub and de Vries, 2017] and vanilla policy gradient [Strub et al., 2017]. To improve the current dialog agents, we made two contributions in Chapter 2 “Learning Goal-Oriented Visual Dialog via Tempered Policy Gradient”.

The **first contribution** is using the advanced RNN structures, including the memory networks [Weston et al., 2014] with attention [Xu et al., 2015], and the sequence-to-sequence model [Sutskever et al., 2014], as shown in Figure 1.1.

The **second contribution** is the development of the Tempered Policy Gradient methods, which improves the exploration of the dialog agents. In general, we use a parameter τ , the sampling temperature of the probabilistic language model, which allows us to explicitly control the strengths of the exploration. We use the temperature parameter $\tau \geq 0$ to adjust the language model to be more conservative or more diversified to achieve exploitation or exploration. The output probability of each word is parameterized by a temperature scalar as:

$$f^\tau(y = n | \pi(x | w)) = \frac{f(y = n | \pi(x | w))^{\frac{1}{\tau}}}{\sum_{m=1}^N f(y = m | \pi(x | w))^{\frac{1}{\tau}}}. \quad (1.4)$$

We use notation f^τ to denote a probability mass function f that is parameterized by a temperature scalar τ . When the temperature is high, $\tau > 1$, the distribution becomes more uniform; when the temperature is low, $\tau < 1$, the distribution appears more spiky. We improve the sampling strategy during exploration based on the term frequency inverse document frequency. The essential idea is that we use a heuristic function h to assign the temperature τ to the word distribution at each time step, t . The temperature is bounded in a predefined range $[\tau_{min}, \tau_{max}]$.

The heuristic function we used here is based upon the term frequency inverse document frequency, $tf-idf$ [Leskovec et al., 2014]. In the context of goal-oriented dialogues, we use the counted number of each word as term frequency tf and the total number of generated dialogues during training as document frequency df . We use the word that has the highest probability to be sampled at current time-step, y_t^* , as the input to the heuristic function h . Here, y_t^* is the maximizer of the probability mass function f , which is defined as $y_t^* = \operatorname{argmax}(f(\pi(x | w)))$. We propose that $tf-idf(y_t^*)$ approximates the concentration level of the distribution, which means that if the same word is always sampled from a distribution, then the distribution is very concentrated. Too much concentration prevents the model from exploration, so that a higher temperature is needed. In order to achieve this effect, the heuristic function is defined as

$$\tau_t^h = h(tf-idf(y_t^*)) = \tau_{min} + (\tau_{max} - \tau_{min}) \frac{tf-idf(y_t^*) - tf-idf_{min}}{tf-idf_{max} - tf-idf_{min}}. \quad (1.5)$$

With this heuristic, words that occur very often are depressed by applying a higher temperature to those words, making them less likely to be selected in the near future. In the forward pass, a word is sampled using $y_t^h \sim f^{\tau_t^h}(\pi(x | w))$. In the backward pass, the weights are updated correspondingly:

$$\Delta w_i = \alpha_i (r - b_i) \partial \ln f(y_t^h | \pi(x | w)) / \partial w_i, \quad (1.6)$$

where τ_t^h is the temperature calculated from the heuristic function.

In experiments, we found that Tempered Policy Gradient improving the performance as well as helping to produce diverse utterances, see Table 1.1. Tempered Policy Gradient is a generic strategy to encourage word exploration on top of policy gradients and can be applied to any dialog agents. For more details about the method, please refer to Chapter 2 “Learning Goal-Oriented Visual Dialog via Tempered Policy Gradient”.



Image	Policy Gradient	Tempered Policy Gradient
	Is it in left? No	Is it a person? No
	Is it in front? No	Is it a vehicle ? Yes
	Is it in right? Yes	Is it a truck ? Yes
	Is it in middle? Yes	Is it in front of photo? No
	Is it person? No	In the left half? No
	Is it ball? No	In the middle of photo? Yes
	Is it bat? No	Is it to the right photo? Yes
	Is it car ? Yes	Is it in the middle of photo? Yes
	Status: Failure	Status: Success
		Is it in left ? No
Is it in front? Yes		In front of photo? Yes
Is it in right? No		In the left half ? Yes
Is it in middle? Yes		Is it in the middle of photo? Yes
Is it person? No		Is it to the left of photo? Yes
Is it giraffe? Yes		Is it to the right photo? No
Is in middle? Yes		In the left in photo? No
Is in middle? Yes		In the middle of photo? Yes
Status: Failure		Status: Success

Table 1.1: Some samples generated by our improved models using REINFORCE (left column: “Policy Gradient”) and Dynamic-TPG (right column: “Tempered Policy Gradient”). The green bounding boxes highlight the target objects; the red boxes highlight the wrong guesses.

1.2.4 Positive Memory Retention

In this section, I introduce the work in paper “Efficient Dialog Policy Learning via Positive Memory Retention” [Zhao and Tresp, 2018a], included in Chapter 3. In Chapter 3, we research on improving the sample-efficiency of a dialog agent. A recent neuroscience literature [Gruber et al., 2016] concludes that in human memory retention, focusing on *rewarded* events has been discovered to be a preferred strategy in the post-learning phase happening in the hippocampus area of the brain. Additionally, Tresp et al. [2015] argue that the brain’s memory functions might inspire technical solutions requiring memory traces.

We believe that this fact also intuitively applies to RL since non-rewarded trajectories do not contribute directly to the estimated gradient to increase the

expected return, since the return of the trajectory is zero. The *high efficiency* of positive memory retention can also be derived from Importance Sampling (IS) [Murphy, 2012]. Consider that the expectation we want to estimate is the expected return $\mathbb{E}[R(\tau)]$, where τ represents the trajectory. Motivated by the concept of importance sampling, to estimate the expected return $\mathbb{E}[R(\tau)]$, it is more efficient to sample from $q(\tau) \propto R(\tau)p(\tau)$ than to sample from $p(\tau)$.

In order to reuse the past positive trajectories in the memory, we need to correct the bias introduced by using samples from the proposal distribution, i.e. the behavior policy. As a second application of importance sampling, we can sample from a behavior policy and then simply re-weight the samples to obtain an unbiased estimate [Jie and Abbeel, 2010]. We can estimate the expected return of a target policy $p(\tau^{(i)}|\pi_\theta)$ as:

$$\hat{J}(\theta) = \frac{1}{n} \sum_{i=1}^n \omega(\tau^{(i)})R(\tau^{(i)}), \text{ with } \tau^{(i)} \sim q \text{ and } \omega(\tau^{(i)}) = \frac{p(\tau^{(i)}|\pi_\theta)}{q(\tau^{(i)}|\pi_{\theta'})} = \frac{\prod_{t=1}^T \pi_{\theta}(a_t|s_t)}{\prod_{t=1}^T \pi_{\theta'}(a_t|s_t)}$$

where n is the number of trajectories used to estimate the expected return $J(\theta) = \mathbb{E}[R(\tau)|\pi_\theta]$ and $\prod_{t=1}^T \pi_{\theta}(a_t|s_t)$ needs to be calculated from the target policy, and $\prod_{t=1}^T \pi_{\theta'}(a_t|s_t)$ has already been calculated from the behavior policy. Therefore, the importance weighted policy gradient is:

$$\nabla_{\theta} \hat{J}(\theta) = \nabla_{\theta} \mathbb{E}_q [\omega(\tau)(R(\tau) - b) \log \pi_{\theta}(a_t|s_t)]. \quad (1.7)$$

This estimator is unbiased, but it suffers from very high variances because it involves a product of a series of unbounded importance weights. To prevent the importance weight from exploding, we select to use the samples that are not far from the target policy.

Through literature [Murphy, 2012], we know that Kullback–Leibler divergence is used to measure the dissimilarity of two probability distributions p and q : $\mathbb{KL}(p \parallel q) \approx \sum_{k=1}^K p_k \log \frac{p_k}{q_k}$. However, KL-divergence is asymmetric, so it cannot directly be used as a distance. To evaluate the distance, we use a symmetric


	#	Question	Answer
	1	Is it 2 in the image?	No
	2	Is it in a yellow background?	No
	3	Is it 9 in the image?	Yes
	4	Is it in a white background?	Yes
	5	Is it a stroke style digit?	Yes
	6	Is it a digit in blue?	No
		Guess: row 1 column 3	✓

Figure 1.2: **MNIST GuessNumber dataset example:** Each sample consists of an image (left), a set of sequential questions with answers (right), and a target digit. The goal of this game is to find out the target digit by a multi-round question-answering. The dataset is available at <https://github.com/ruizhaogit/MNIST-GuessNumber>

version of the KL-divergence, i.e. the Jensen-Shannon divergence [Lin, 1991]:

$$\text{JS}(p, q) = 0.5\text{KL}(p \parallel 0.5(p + q)) + 0.5\text{KL}(q \parallel 0.5(p + q)). \quad (1.8)$$

We now derive a formulation of the JS-divergence, as a distance metric, which is related to the importance weight ω :

$$\text{JS}(p, q) \approx 0.5 \sum_{k=1}^K p_k \log \frac{2}{1 + \omega_k} + 0.5 \sum_{k=1}^K q_k \log \frac{2}{\frac{1}{\omega_k} + 1} \quad (1.9)$$

We can see that the distance between the proposal distribution q and the optimal solution p depends on both ω_k and $1/\omega_k$. To limit the variance of the importance sampling, we limit the importance weight as $\omega_k \leq \omega_{max}$ and its inverse as $1/\omega_k \leq \omega_{max}$. Subsequently, we define a trust region of importance weights, $\omega_k \in [1/\omega_{max}, \omega_{max}]$ and only use trajectories whose importance weights fall within this range. This stabilizes the training. Similar to this Bounded Importance Weight Proposal, we propose to use several other tricks to stabilize the training, such as Probability Updating and Policy Search via Early Stopping, for more detail see Chapter 3 “Efficient Dialog Policy Learning via Positive Memory Retention”.

To test the proposed method, we synthesize a visual dialog dataset based on

MNIST, named MNIST GuessNumber, see Figure 1.2. On this synthetic dataset, we run the ablation study of each module. Afterwards, we show that with positive memory retention, the sample-efficiency is improved by a factor of two in both the synthetic dataset and the real-world dataset.

The **main contribution** of Chapter 3 is that we show that our proposed mechanisms permit an efficient reuse of past samples in on-policy policy gradients methods. These extensions also work well in dialog settings, which are challenging due to the sparse reward and the large action space. For more detailed information, please refer to Chapter 3 “Efficient Dialog Policy Learning via Positive Memory Retention”.

1.3 Deep Reinforcement Learning for Robotics

In this section, we are more focused on the core reinforcement learning design and evaluate the developed methods in the continuous control domain. First, we will briefly introduce the baseline methods, including Deep Deterministic Policy Gradient [Lillicrap et al., 2016] in Section 1.3.1 and Hindsight Experience Replay [Andrychowicz et al., 2017] in Section 1.3.2. Then, we will dive into the newly developed reinforcement learning frameworks in Section 1.3.3, 1.3.4, 1.3.5, and 1.3.6. Some of these works draw inspirations from the cognitive neuroscience [Gazzaniga et al., 2006, Sansone and Harackiewicz, 2000, Gruber et al., 2014], such as the curiosity-driven experience prioritization framework in Section 1.3.4, which is also proven to connect to the maximum entropy principle in Section 1.3.5. Furthermore, we design an internal drive for reinforcement learning agents based on the mutual information in Section 1.3.6, to enable the agent to learn meaningful behaviors.

1.3.1 Deep Deterministic Policy Gradient

First of all, we consider the robotic tasks, shown in Figure 1.3, as Markov Decision Processes (MDP). We assume the environment is fully observable, including a set

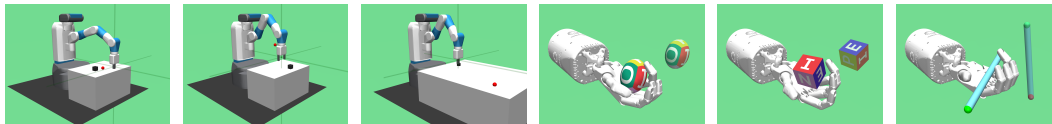


Figure 1.3: Robot arm Fetch and Shadow Dextrous hand environment in OpenAI Gym: FetchPush, FetchPickAndPlace, FetchSlide, HandManipulateEgg, HandManipulateBlock, and HandManipulatePen.

of state \mathcal{S} , a set of action \mathcal{A} , a distribution of initial states $p(s_0)$, transition probabilities $p(s_{t+1}|s_t, a_t)$, a reward function $r: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, and also a discount factor $\gamma \in [0, 1]$. These components formulate a Markov decision process represented as a tuple, $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$. A policy π maps a state to an action, $\pi: \mathcal{S} \rightarrow \mathcal{A}$.

For each episode, an initial state s_0 is sampled from the distribution $p(s_0)$. At each timestep t , an agent performs an action a_t at the current state s_t , which follows a policy $a_t = \pi(s_t)$. Afterwards, a reward $r_t = r(s_t, a_t)$ is produced by the environment and the next state s_{t+1} is sampled from the distribution $p(\cdot|s_t, a_t)$. The reward might be discounted by a factor γ at each timestep. The goal of the agent is to maximize the accumulated reward, $R_t = \sum_{i=t}^{\infty} \gamma^{i-t} r_i$, over all episodes, which is equivalent to maximizing the expected return, $\mathbb{E}_{s_0}[R_0|s_0]$.

The objective $\mathbb{E}_{s_0}[R_0|s_0]$ can be maximized using temporal difference learning, policy gradients, or the combination of both, i.e. the actor-critic methods [Sutton and Barto, 2018]. For continuous control tasks, Deep Deterministic Policy Gradient (DDPG) shows promising performance, which is essentially an off-policy actor-critic method [Lillicrap et al., 2016]. DDPG has an actor network, $\pi: \mathcal{S} \rightarrow \mathcal{A}$, that learns the policy directly. It also has a critic network, $Q: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, that learns the action-value function, i.e. Q-function Q^π . During training, the actor network uses a behavior policy to explore the environment, which is the target policy plus some noise, $\pi_b = \pi(s) + \mathcal{N}(0, 1)$. The critic is trained using temporal difference learning with the actions produced by the actor: $y_t = r_t + \gamma Q(s_{t+1}, \pi(s_{t+1}))$. The actor is trained using policy gradients by descending on the gradients of the loss function, $\mathcal{L}_a = -\mathbb{E}_s[Q(s, \pi(s))]$, where s is sampled from the replay buffer. For stability reasons, the target y_t for the actor is usually calculated using a separate network, i.e. an averaged version of

the previous Q-function networks [Mnih et al., 2015, Lillicrap et al., 2016, Polyak and Juditsky, 1992]. The parameters of the actor and critic are updated using backpropagation.

1.3.2 Hindsight Experience Replay

For multi-goal continuous control tasks, DDPG can be extended with Universal Value Function Approximators (UVFA) [Schaul et al., 2015]. UVFA essentially generalizes the Q-function to multiple goal states $g \in \mathcal{G}$. For the critic network, the Q-value depends not only on the state-action pairs, but also depends on the goals: $Q^\pi(s_t, a_t, g) = \mathbb{E}[R_t | s_t, a_t, g]$.

For robotic tasks, if the goal is challenging and the reward is sparse, then the agent could perform badly for a long time before learning anything. Hindsight Experience Replay (HER) encourages the agent to learn something instead of nothing. During exploration, the agent samples some trajectories conditioned on the real goal g . The main idea of HER is that during replay, the selected transitions are substituted with achieved goals g' instead of the real goals. In this way, the agent could get a sufficient amount of reward signal to begin learning. Andrychowicz et al. [2017] show that HER makes training possible in challenging robotic environments. However, the episodes are uniformly sampled in the replay buffer, and subsequently, the virtual goals are sampled from the episodes. More sophisticated replay strategies are requested for improving sample-efficiency [Plappert et al., 2018].

1.3.3 Energy-based Hindsight Experience Prioritization

In this section, I introduce the work in paper “Energy-based Hindsight Experience Prioritization” [Zhao and Tresp, 2018b], included in Chapter 4. Prior to the energy-based hindsight experience prioritization method, we illustrate here the work-energy principle using robotic manipulation examples. In physics, a force is said to do work if, when acting, there is a displacement of the point of application in the direction of the force [Tipler and Mosca, 2007]. For instance, a robot arm

picks up an object from the floor, and places it on the shelf. The work done on the object is equal to the weight of the object multiplied by the vertical distance to the floor. As a result, the potential energy of the object becomes higher.

The work-energy principle states that the work done by all forces acting on a particle equals the change in the kinetic energy of the particle [Meriam and Kraige, 2012]. That is, the work W done by a force on an object (simplified as a particle) equals the change in the object’s kinetic energy E_k [Young et al., 2006], $W = \Delta E_k = \frac{1}{2}mv_2^2 - \frac{1}{2}mv_1^2$, where v_1 and v_2 are the speeds of the object before and after the work is done, respectively, and m is its mass. As the robot arm is moving the object towards the shelf, the work is being done by the robot on the object. Consequently, the kinetic energy of the object increases.

Consider a robotic manipulation task. We observe that in order to complete the tasks, the robot needs to apply force and do work on the object. Typically, the more difficult a task is, the more work from the robot is required. Consequently, the energy of the object is changed by the robot. Thus, our hypothesis is that, in robotic manipulation tasks, the trajectory energy of the object indicates the difficulty level of the tasks.

From the perspective of curriculum learning, we want to assign the right level of curriculum to the agent. The curriculum should not be too difficult to achieve, also not too simple to learn. We use the trajectory energy to evaluate the difficulty level of the curriculum, and then prioritize the difficult but still achievable tasks for the agent. In this way, the agent might learn with higher sample-efficiency. In robotic tasks, training samples are expensive to acquire, making sample-efficiency in learning important.

Now, we formally introduce the trajectory energy function. First, the total energy of a state s_t includes the potential energy, the kinetic energy, and the rotation energy: $E(s_t) = E_p(s_t) + E_k(s_t) + E_r(s_t)$.

We define the transition energy as the total energy increase from the previous state s_{t-1} to the current state s_t : $E_{tran}(s_{t-1}, s_t) = \text{clip}(E(s_t) - E(s_{t-1}), 0, E_{tran}^{max})$, where $t \geq 1$ and E_{tran}^{max} is the predefined maximal transition energy value. The clip function limits the transition energy value in an interval of $[0, E_{tran}^{max}]$. Here, we

are only interested in the positive transition energy because the energy increase of the object is only due to the work done by the robot. The robot does work on the object, consequently, the total energy of the object increases. In practice, to mitigate the influence of some particular large transition energy, we find it useful to clip the transition energy with a threshold value E_{tran}^{max} .

Given the definition of the transition energy, we define the trajectory energy as the sum of the transition energy over all the timesteps in the trajectory, mathematically: $E_{traj}(\mathcal{T}) = E_{traj}(s_0, s_1, \dots, s_T) = \sum_{t=1}^T E_{tran}(s_{t-1}, s_t)$

During experience replay, the agent uses the trajectory energy values directly as the probability for sampling. This means that the high energy trajectories have higher priorities to be replayed. Mathematically, the probability of a trajectory \mathcal{T}_i to be replayed after the prioritization is: $p(\mathcal{T}_i) = E_{traj}(\mathcal{T}_i) / \sum_{n=1}^N E_{traj}(\mathcal{T}_n)$ where N is the total number of trajectories in the buffer.

The **main contribution** here is the invention of the energy-based prioritization framework. Our empirical results show that our proposed method surpasses state-of-the-art approaches in terms of both performance and sample-efficiency on different robotic tasks, without increasing computational time. We also find that the trajectory energy is correlated with the TD-errors during training. For more details, please refer to Chapter 4 “Energy-based hindsight experience prioritization”.

1.3.4 Curiosity-driven Experience Prioritization

In this section, I introduce the work in paper “Curiosity-Driven Experience Prioritization via Density Estimation” [Zhao and Tresp, 2019], included in Chapter 5.

Moving further with the energy-based prioritization approach, we think about that how can we enable the agent to self-learn to prioritize the experiences. The recent neuroscience research [Gruber et al., 2014] has shown that curiosity can enhance learning. They discovered that when curiosity motivated learning was activated, there was increased activity in the hippocampus, a brain region that is important for human memory. During memory replay, people are more cu-

rious about the episodes that are relatively different and focus more on those. This curiosity mechanism could help a reinforcement learning agent learn more efficiently.

Therefore, we propose the curiosity-driven prioritization framework. In a nutshell, we first estimate the density of each trajectory according to its achieved goal states, then prioritize the trajectories with lower density for replay. We estimate the distribution of the data in the replay buffer using a density model, such as a Gaussian Mixture Model (GMM). The density model fits on the data in the memory buffer every epoch and refreshes the density for each trajectory in the buffer. $\rho = \text{GMM}(\mathcal{T}) = \sum_{k=1}^K c_k \mathcal{N}(\mathcal{T} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ where $\mathcal{T} = (s_0 || s_1 || \dots || s_T)$ and each trajectory \mathcal{T} has the same length. The symbol $||$ denotes concatenation. We normalize the trajectory densities using $\rho_i = \rho_i / \sum_{n=1}^N \rho_n$ where N is the number of trajectories in the memory buffer. Now the density ρ is between zero and one, i.e. $0 \leq \rho \leq 1$. After calculating the trajectory density, the agent stores the density value along with the trajectory in the memory buffer for later prioritization. During replay, the agent puts more focus on the under-represented achieved states and prioritizes the according trajectories. We defined the complementary trajectory density as: $\bar{\rho} \propto 1 - \rho$. When the agent replays the samples, it first ranks all the trajectories with respect to their complementary density values $\bar{\rho}$, and then uses the ranking number (starting from zero) directly as the probability for sampling. Mathematically, the probability of a trajectory to be replayed after the prioritization is: $p(\mathcal{T}_i) = \text{rank}(\bar{\rho}(\mathcal{T}_i)) / \sum_{n=1}^N \text{rank}(\bar{\rho}(\mathcal{T}_n))$ where N is the total number of trajectories in the buffer, and $\text{rank}(\cdot) \in \{0, 1, \dots, N - 1\}$.

The **main contribution** here is the development of the curiosity-driven prioritization framework. Compared to the energy-based approach, the curiosity-based variant learns to estimate the distribution of the trajectories in the replay buffer and prioritize these rare trajectories by itself. However, this learning process does take more computational time in comparison to the energy-based approach. This curiosity-based method also improve the final performance and the sample-efficiency and could be applied to a wider range of tasks. For more details, please refer to Chapter 5 “Curiosity-Driven Experience Prioritization via

Density Estimation”.

1.3.5 Maximum Goal Entropy Reinforcement Learning

In this section, I introduce the work in paper “Maximum Entropy-Regularized Multi-Goal Reinforcement Learning” [Zhao et al., 2019], included in Chapter 6.

Take a deeper look at the curiosity-driven prioritization framework, we derive a connection to the maximum entropy principle in the Multi-Goal Reinforcement Learning settings. In Multi-Goal Reinforcement Learning, an agent learns to achieve multiple goals with a goal-conditioned policy. During learning, the agent first collects the trajectories into a replay buffer, and later these trajectories are selected randomly for replay. However, the achieved goals in the replay buffer are often biased towards the behavior policies. From a Bayesian perspective, when there is no prior knowledge about the target goal distribution, the agent should learn uniformly from diverse achieved goals. Therefore, we first propose a novel multi-goal RL objective based on weighted entropy.

We want to encourage the agent to traverse diverse goal-state trajectories, and at the same time, maximize the expected return. This is like maximizing the empowerment [Mohamed and Rezende, 2015] of an agent attempting to achieve multiple goals. We propose the reward-weighted entropy objective for multi-goal RL, which is given as

$$\eta^{\mathcal{H}}(\boldsymbol{\theta}) = \mathcal{H}_p^w(\mathcal{T}^g) = \mathbb{E}_p \left[\log \frac{1}{p(\boldsymbol{\tau}^g)} \sum_{t=1}^T r(S_t, G^e) \mid \boldsymbol{\theta} \right]. \quad (1.10)$$

For simplicity, we use $p(\boldsymbol{\tau}^g)$ to represent $\sum_{g^e} p_{\mathcal{R}}(\boldsymbol{\tau}^g, g^e \mid \boldsymbol{\theta})$, which is the occurrence probability of the goal-state trajectory $\boldsymbol{\tau}^g$. The expectation is calculated based on $p(\boldsymbol{\tau}^g)$ as well, so the proposed objective is the weighted entropy [Guaşu, 1971, Kelbert et al., 2017] of $\boldsymbol{\tau}^g$, which we denote as $\mathcal{H}_p^w(\mathcal{T}^g)$, where the weight w is the accumulated reward $\sum_{t=1}^T r(s_t, g^e)$ in our case.

The objective function, Equation (1.10), has two interpretations. The first interpretation is to maximize the weighted expected return, where the rare trajec-

ories have larger weights. Note that when all trajectories occur uniformly, this weighting mechanism has no effect. The second interpretation is to maximize a reward-weighted entropy, where the more rewarded trajectories have higher weights. This objective encourages the agent to learn how to achieve diverse goal-states, as well as to maximize the expected return.

In Equation (1.10), the weight, $\log(1/p(\boldsymbol{\tau}^g))$, is unbounded, which makes the training of the universal function approximator unstable. Therefore, we propose a safe surrogate objective, $\eta^{\mathcal{L}}$, which is essentially a lower bound of the original objective. To construct the safe surrogate objective, we sample the trajectories from the replay buffer with a proposal distribution, $q(\boldsymbol{\tau}^g) = \frac{1}{Z}p(\boldsymbol{\tau}^g)(1 - p(\boldsymbol{\tau}^g))$. $p(\boldsymbol{\tau}^g)$ represents the distribution of the goal trajectories in the replay buffer. The surrogate objective is given in Theorem 1, which is proved to be a lower bound of the original objective, Equation (1.10).

Theorem 1. *The surrogate $\eta^{\mathcal{L}}(\boldsymbol{\theta})$ is a lower bound of the objective function $\eta^{\mathcal{H}}(\boldsymbol{\theta})$, i.e., $\eta^{\mathcal{L}}(\boldsymbol{\theta}) < \eta^{\mathcal{H}}(\boldsymbol{\theta})$, where*

$$\eta^{\mathcal{H}}(\boldsymbol{\theta}) = \mathcal{H}_p^w(\boldsymbol{\tau}^g) = \mathbb{E}_p \left[\log \frac{1}{p(\boldsymbol{\tau}^g)} \sum_{t=1}^T r(S_t, G^e) \mid \boldsymbol{\theta} \right] \quad (1.11)$$

$$\eta^{\mathcal{L}}(\boldsymbol{\theta}) = Z \cdot \mathbb{E}_q \left[\sum_{t=1}^T r(S_t, G^e) \mid \boldsymbol{\theta} \right] \quad \text{where } q(\boldsymbol{\tau}^g) = \frac{1}{Z}p(\boldsymbol{\tau}^g)(1 - p(\boldsymbol{\tau}^g)) \quad (1.12)$$

Z is the normalization factor for $q(\boldsymbol{\tau}^g)$. $\mathcal{H}_p^w(\boldsymbol{\tau}^g)$ is the weighted entropy [Guiaşu, 1971, Kelbert et al., 2017], where the weight is the return $\sum_{t=1}^T r(S_t, G^e)$.

Proof. See the appendix of Chapter 6 “Maximum Entropy-Regularized Multi-Goal Reinforcement Learning”. \square

To optimize the surrogate objective, Equation (1.12), we cast the optimization process into a prioritized sampling framework, which is very similar to the curiosity-driven prioritization in Section 1.3.4. We name this as the Maximum Entropy-based Prioritization (MEP) and summarize the process in Figure 1.4.

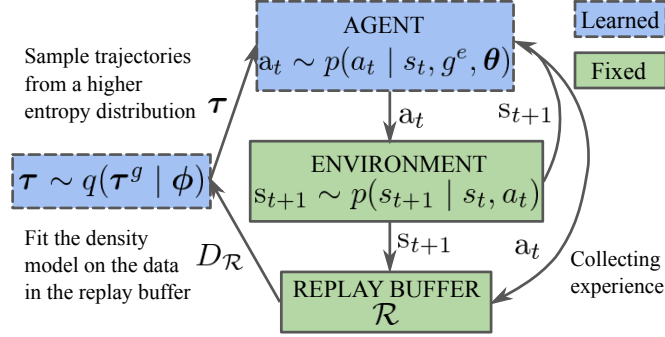


Figure 1.4: **MEP Algorithm:** We update the density model to construct a higher entropy distribution of achieved goals and update the agent with the more diversified training distribution.

At each iteration, we first construct the proposal distribution $q(\tau^g)$, which has a higher entropy than $p(\tau^g)$. This ensures that the agent learns from a more diverse goal-state distribution. In Theorem 2, we prove that the entropy with respect to $q(\tau^g)$ is higher than the entropy with respect to $p(\tau^g)$.

Theorem 2. *Let the probability density function of goals in the replay buffer be*

$$p(\tau^g), \text{ where } p(\tau_i^g) \in (0, 1) \text{ and } \sum_{i=1}^N p(\tau_i^g) = 1. \quad (1.13)$$

Let the proposal probability density function be defined as

$$q(\tau_i^g) = \frac{1}{Z} p(\tau_i^g) (1 - p(\tau_i^g)), \text{ where } \sum_{i=1}^N q(\tau_i^g) = 1. \quad (1.14)$$

Then, the proposal goal distribution has an equal or higher entropy

$$\mathcal{H}_q(\mathcal{T}^g) - \mathcal{H}_p(\mathcal{T}^g) \geq 0. \quad (1.15)$$

Proof. See the appendix of Chapter 6 “Maximum Entropy-Regularized Multi-Goal Reinforcement Learning”. \square

To optimize the surrogate objective with prioritized sampling, we need to know the probability distribution of a goal-state trajectory $p(\tau^g)$. We use a

Latent Variable Model (LVM) [Murphy, 2012] to model the underlying distribution of $p(\boldsymbol{\tau}^g)$, since LVM is suitable for modeling complex distributions. Specifically, we use $p(\boldsymbol{\tau}^g | z_k)$ to denote the latent-variable-conditioned goal-state trajectory distribution, which we assume to be Gaussians. z_k is the k -th latent variable, where $k \in \{1, \dots, K\}$ and K is the number of the latent variables. The resulting model is a Mixture of Gaussians (MoG), mathematically, $p(\boldsymbol{\tau}^g | \boldsymbol{\phi}) = \frac{1}{Z} \sum_{i=k}^K c_k \mathcal{N}(\boldsymbol{\tau}^g | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, where each Gaussian, $\mathcal{N}(\boldsymbol{\tau}^g | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, has its own mean $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma}_k$, c_k represents the mixing coefficients, and Z is the partition function. The model parameter $\boldsymbol{\phi}$ includes all mean $\boldsymbol{\mu}_i$, covariance $\boldsymbol{\Sigma}_i$, and mixing coefficients c_k .

In prioritized sampling, we use the complementary predictive density of a goal-state trajectory $\boldsymbol{\tau}^g$ as the priority, which is given as $\bar{p}(\boldsymbol{\tau}^g | \boldsymbol{\phi}) \propto 1 - p(\boldsymbol{\tau}^g | \boldsymbol{\phi})$. The complementary density describes the likelihood that a goal-state trajectory $\boldsymbol{\tau}^g$ occurs in the replay buffer. A high complementary density corresponds to a rare occurrence of the goal trajectory. We use the complementary density to construct the proposal distribution as a joint distribution $q(\boldsymbol{\tau}^g) \propto \bar{p}(\boldsymbol{\tau}^g | \boldsymbol{\phi}) p(\boldsymbol{\tau}^g) \propto (1 - p(\boldsymbol{\tau}^g | \boldsymbol{\phi})) p(\boldsymbol{\tau}^g)$. With prioritized sampling, the agent learns to maximize the return of a more diverse goal distribution.

The **main contribution** here is the derivation of the mathematical connection between a proposed maximum-entropy regularized multi-goal RL objective and the developed prioritization framework. The Maximum Entropy-based Prioritization method improves both the final performance and the sample-efficiency and is applicable to any multi-goal RL agents. For more details, please refer to Chapter 6 “Maximum Entropy-Regularized Multi-Goal Reinforcement Learning”.

1.3.6 Mutual Information-based State-Control

In this section, I introduce the work in paper “Mutual Information-based State-Control for Intrinsically Motivated Reinforcement Learning” [Zhao et al., 2020], included in Chapter 7.

In psychology [Sansone and Harackiewicz, 2000], behavior is considered in-

trinsically motivated when it originates from an internal drive. An intrinsic motivation is essential to develop behaviors required for accomplishing a broad range of tasks rather than solving a specific problem guided by an external reward.

Motivated by the idea that an agent should be “prepared” to control the goal state with its own directly controllable state, we formulate the problem of learning without external supervision as one of learning a policy $\pi_\theta(a_t | s_t)$ with parameters θ to maximize intrinsic MI rewards, $r = I(S^g; S^c)$. Our framework simultaneously learns a policy and an intrinsic reward function by maximizing the MI between the goal states and the controllable states. Mathematically, the MI between the goal state random variable S^g and the controllable state random variable S^c is represented as follows:

$$I(S^g; S^c) = H(S^g) - H(S^g | S^c) \quad (1.16)$$

$$= D_{KL}(\mathbb{P}_{S^g S^c} || \mathbb{P}_{S^g} \otimes \mathbb{P}_{S^c}) \quad (1.17)$$

$$= \sup_{T: \Omega \rightarrow \mathbb{R}} \mathbb{E}_{\mathbb{P}_{S^g S^c}}[T] - \log(\mathbb{E}_{\mathbb{P}_{S^g} \otimes \mathbb{P}_{S^c}}[e^T]) \quad (1.18)$$

$$\geq \sup_{\phi \in \Phi} \mathbb{E}_{\mathbb{P}_{S^g S^c}}[T_\phi] - \log(\mathbb{E}_{\mathbb{P}_{S^g} \otimes \mathbb{P}_{S^c}}[e^{T_\phi}]) = I_\Phi(S^g; S^c), \quad (1.19)$$

where $\mathbb{P}_{S^g S^c}$ is the joint probability distribution; $\mathbb{P}_{S^g} \otimes \mathbb{P}_{S^c}$ is the product of the marginal distributions \mathbb{P}_{S^g} and \mathbb{P}_{S^c} ; KL denotes the Kullback-Leibler (KL) divergence. Equation (1.16) tells us that the agent should maximize the entropy of goal states $H(S^g)$, and concurrently, should minimize the conditional entropy of goal states given the controllable states $H(S^g | S^c)$. When the conditional entropy $H(S^g | S^c)$ is small, it becomes easy to predict the goal states based on the controllable states. Equation (1.17) gives us MI in the KL divergence form.

MI is notoriously difficult to compute in real-world settings [Hjelm et al., 2019]. Motivated by MINE [Belghazi et al., 2018], we use a lower bound to approximate the MI quantity $I(S^g; S^c)$. First, we rewrite Equation (1.17), the KL formulation of the MI objective, using the Donsker-Varadhan representation, to Equation (1.18) [Donsker and Varadhan, 1975]. The input space Ω is a compact domain of \mathbb{R}^d , i.e., $\Omega \subset \mathbb{R}^d$, and the supremum is taken over all functions T such

that the two expectations are finite. Secondly, we lower bound the MI in the Donsker-Varadhan representation with the compression lemma in the PAC-Bayes literature and then derive Equation (1.19) [Banerjee, 2006, Belghazi et al., 2018]. The expectations in Equation (1.19) are estimated by using empirical samples from $\mathbb{P}_{S^g S^c}$ and $\mathbb{P}_{S^g} \otimes \mathbb{P}_{S^c}$. We can also sample the marginal distributions by shuffling the samples from the joint distribution along the axis [Belghazi et al., 2018]. The derived MI reward function, $r = I_\phi(S^g; S^c)$, can be trained by gradient ascent. The statistics model T_ϕ is parameterized by a deep neural network with parameters $\phi \in \Phi$, which is capable of estimating the MI with arbitrary accuracy.

However, for evaluating the MI, this lower bound, Equation (1.20) Left-Hand Side (LHS), is time-consuming to calculate because it needs to process on all the samples from the whole trajectory. To improve its scalability and efficiency, we derive a surrogate objective, Equation (1.20) Right-Hand Side (RHS), which is computed much more efficiently. Each time, to calculate the MI reward for the transition $r = I_\phi(S^g; S^c | \mathcal{T}')$, the new objective only needs to calculate over a small fraction of the complete trajectory, τ' . The trajectory fraction, τ' , is defined as adjacent state pairs, $\tau' = \{s_t, s_{t+1}\}$, and \mathcal{T}' represents its corresponding random variable. In the paper we derive the following Lemma 3:

Lemma 3. *The mutual information quantity $I_\phi(S^g; S^c | \mathcal{T})$ increases when we maximize the surrogate objective $\mathbb{E}_{\mathbb{P}_{\mathcal{T}'}}[I_\phi(S^g; S^c | \mathcal{T}')$, mathematically,*

$$I_\phi(S^g; S^c | \mathcal{T}) \times \mathbb{E}_{\mathbb{P}_{\mathcal{T}'}}[I_\phi(S^g; S^c | \mathcal{T}')], \quad (1.20)$$

where S^g , S^c , and \mathcal{T} denote goal states, controllable states, and trajectories, respectively. The trajectory fractions are defined as the adjacent state pairs, namely $\mathcal{T}' = \{S_t, S_{t+1}\}$. The symbol \times denotes a monotonically increasing relationship between two variables and ϕ represents the parameter of the statistics model in MINE. Proof. See Chapter 7 “Mutual Information-based State-Control for Intrinsically Motivated Reinforcement Learning” Section “Efficient Learning State-Control”. \square

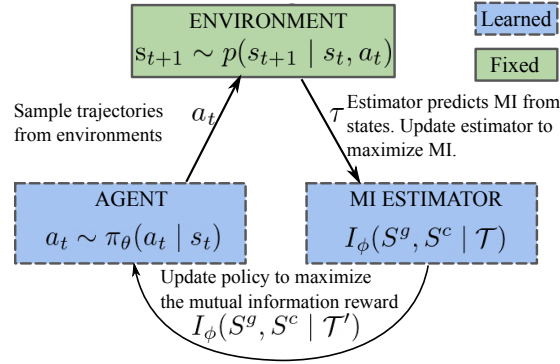


Figure 1.5: **MISC Algorithm:** We update the estimator to better predict the mutual information (MI), and update the agent to control goal states to have higher MI with the controllable states.

The derived MI surrogate objective, Equation (1.20) RHS, brings us two important benefits. First, it enables us to estimate the MI reward for each transition with much less computational time because we only use the trajectory fraction, instead of the trajectory. This approximately reduces the complexity from $\mathcal{O}(t^*)$ to $\mathcal{O}(1)$ with respect to the trajectory length t^* . Secondly, this way of estimating MI also enables us to assign rewards more accurately at the transition level because now we use only the relevant state pair to calculate the transition reward. Formally, we define the transition MI reward as the MI estimation of each trajectory fraction, namely

$$r_\phi(a_t, s_t) := I_\phi(S^g; S^c | \mathcal{T}') = 0.5 \sum_{i=t}^{t+1} T_\phi(s_i^g, s_i^c) - \log(0.5 \sum_{i=t}^{t+1} e^{T_\phi(s_i^g, \bar{s}_i^c)}), \quad (1.21)$$

where $(s_i^g, s_i^c) \sim \mathbb{P}_{S^g S^c | \mathcal{T}'}$, $\bar{s}_i^c \sim \mathbb{P}_{S^c | \mathcal{T}'}$, and $\mathcal{T}' = \{s_t, s_{t+1}\}$. In case that the estimated MI value is particularly small, we scale the reward with a hyper-parameter α and clip the reward between 0 and 1. Overall, the agent is rewarded for controlling the goal states to have higher mutual information with its controllable states, which is considered the “preparedness” to achieve any future goal. We summarize the complete training algorithm in Figure 1.5.

The **main contribution** here is the introduce of Mutual Information-based State-Control (MISC), an unsupervised RL framework for learning useful control behaviors. The derived efficient mutual information-based theoretical ob-

jective encourages the agent to control states without any task reward. MISC enables the agent to self-learn different control behaviors, which are non-trivial, intuitively meaningful, and useful for learning and planning. Additionally, the pretrained policy and the mutual information estimator significantly accelerate learning in the presence of task rewards. For more details, please refer to Chapter 7 “Mutual Information-based State-Control for Intrinsically Motivated Reinforcement Learning”. A video showing experimental results is available at <https://youtu.be/CT4CKMWBZ0>.

Chapter 2

Learning Goal-Oriented Visual Dialog via Tempered Policy Gradient

LEARNING GOAL-ORIENTED VISUAL DIALOG VIA TEMPERED POLICY GRADIENT

Rui Zhao, Volker Tresp

Ludwig Maximilian University, Oettingenstr. 67, 80538 Munich, Germany

Siemens AG, Corporate Technology, Otto-Hahn-Ring 6, 81739 Munich, Germany

ABSTRACT

Learning goal-oriented dialogues by means of deep reinforcement learning has recently become a popular research topic. However, commonly used policy-based dialogue agents often end up focusing on simple utterances and suboptimal policies. To mitigate this problem, we propose a class of novel temperature-based extensions for policy gradient methods, which are referred to as Tempered Policy Gradients (TPGs). On a recent AI-testbed, i.e., the GuessWhat?! game, we achieve significant improvements with two innovations. The first one is an extension of the state-of-the-art solutions with Seq2Seq and Memory Network structures that leads to an improvement of 7%. The second one is the application of our newly developed TPG methods, which improves the performance additionally by around 5% and, even more importantly, helps produce more convincing utterances.

Index Terms— Goal-Oriented Dialog System, Deep Reinforcement Learning, Recurrent Neural Network

1. INTRODUCTION

In recent years, deep learning has shown convincing performance in various areas such as image recognition, speech recognition, and natural language processing (NLP). Deep neural nets are capable of learning complex dependencies from huge amounts of data and its human generated annotations in a supervised way. In contrast, reinforcement learning agents [2] can learn directly from their interactions with the environment without any supervision and surpass human performance in several domains, for instance in the game of GO [3], as well as many computer games [4]. In this paper we are concerned with the application of both approaches to goal-oriented dialogue systems [5, 6, 7, 8, 9, 10, 11, 7, 12, 13, 14], a problem that has recently caught the attention of machine learning researchers. De Vries et al. [6] have proposed as AI-testbed a visual grounded object guessing game called GuessWhat?!. Das et al. [7] formulated a visual dialogue system which is about two chatbots asking and answering questions to identify a specific image within a group of images. More practically, dialogue agents have been applied

to negotiate a deal [12] and access certain information from knowledge bases [13]. The essential idea in these systems is to train different dialogue agents to accomplish the tasks. In those papers, the agents have been trained with policy gradients, i.e. REINFORCE [15].

In order to improve the exploration quality of policy gradients, we present three instances of temperature-based methods. The first one is a single-temperature approach which is very easy to apply. The second one is a parallel approach with multiple temperature policies running concurrently. This second approach is more demanding on computational resources, but results in more stable solutions. The third one is a temperature policy approach that dynamically adjusts the temperature for each action at each time-step, based on action frequencies. This dynamic method is more sophisticated and proves more efficient in the experiments. In the experiments, all these methods demonstrate better exploration strategies in comparison to the plain policy gradient.

We demonstrate our approaches using a real-world dataset called GuessWhat?!. The GuessWhat?! game [6] is a visual object discovery game between two players, the Oracle and the Questioner. The Questioner tries to identify an object by asking the Oracle questions. The original works [6, 8] first proposed supervised learning to simulate and optimize the game. Strub et al. [8] showed that the performance could be improved by applying plain policy gradient reinforcement learning, which maximizes the game success rate, as a second processing step. Building on these previous works, we propose two network architecture extensions. We utilize a Seq2Seq model [16] to process the image along with the historical dialogues for question generation. For the guessing task, we develop a Memory Network [17] with Attention Mechanism [18] to process the generated question-answer pairs. We first train these two models using the plain policy gradient and use them as our baselines. Subsequently, we train the models with our new TPG methods and compare the performances with the baselines. We show that the TPG method is compatible with state-of-the-art architectures such as Seq2Seq and Memory Networks and contributes orthogonally to these advanced neural architectures. To the best of our knowledge, the presented work is the first to propose temperature-based policy gradient methods to leverage explo-

This paper is an extended version of the IJCAI workshop paper [1].

ration and exploitation in the field of goal-oriented dialogue systems. We demonstrate the superior performance of our TPG methods by applying it to the GuessWhat?! game.

2. PRELIMINARIES

In our notation, we use \mathbf{x} to denote the input to a policy network π , and x_i to denote the i -th element of the input vector. Similarly, \mathbf{w} denotes the weight vector of π , and w_i denotes the i -th element of the weight vector of that π . The output y is a multinoulli random variable with N states that follows a probability mass function, $f(y = n | \pi(\mathbf{x} | \mathbf{w}))$, where $\sum_{n=1}^N f(y = n | \pi(\mathbf{x} | \mathbf{w})) = 1$ and $f(\cdot) \geq 0$. In a nutshell, a policy network parametrizes a probabilistic unit that produces the sampled output, mathematically, $y \sim f(\pi(\mathbf{x} | \mathbf{w}))$.

Typically, the expected value of the accumulated reward, i.e. return, conditioned on the policy network parameters $E(r | \mathbf{w})$ is used. Here, E denotes the expectation operator, r the accumulated reward signal, and \mathbf{w} the network weight vector. The objective of reinforcement learning is to update the weights in a way that maximizes the expected return at each trial. In particular, the REINFORCE updating rule is: $\Delta w_i = \alpha_i(r - b_i)e_i$, where Δw_i denotes the weight adjustment of weight w_i , α_i is a nonnegative learning rate factor, and b_i is a reinforcement baseline. The e_i is the *characteristic eligibility* of w_i , defined as $e_i = (\partial f / \partial w_i) / f = \partial \ln f / \partial w_i$. Williams [15] has proved that the updating quantity, $(r - b_i) \partial \ln f / \partial w_i$, represents an unbiased estimate of $\partial E(r | \mathbf{w}) / \partial w_i$.

3. TEMPERED POLICY GRADIENT

In order to improve the exploration quality of REINFORCE in the task of optimizing policy-based dialogue agents, we attempt to find the optimal compromise between exploration and exploitation. In TPGs we introduce a parameter τ , the sampling temperature of the probabilistic output unit, which allows us to explicitly control the strengths of the exploration.

3.1. Exploration and Exploitation

The trade-off between exploration and exploitation is one of the great challenges in reinforcement learning [2]. To obtain a high reward, an agent must exploit the actions that have already proved effective in getting more rewards. However, to discover such actions, the agent must try actions, which appear suboptimal, to explore the action space. In a stochastic task like text generation, each action, i.e. a word, must be tried many times to find out whether it is a reliable choice or not. The exploration-exploitation dilemma has been intensively studied over many decades [19, 20, 21]. Finding the balance between exploration and exploitation is considered crucial for the success of reinforcement learning [22].

3.2. Temperature Sampling

In text generation, it is well-known that the simple trick of temperature adjustment is sufficient to shift the language model to be more conservative or more diversified [23]. In order to control the trade-off between exploration and exploitation, we borrow the strength of the temperature parameter $\tau \geq 0$ to control the sampling. The output probability of each word is transformed by a temperature function as:

$$f^\tau(y = n | \pi(\mathbf{x} | \mathbf{w})) = \frac{f(y = n | \pi(\mathbf{x} | \mathbf{w}))^{\frac{1}{\tau}}}{\sum_{m=1}^N f(y = m | \pi(\mathbf{x} | \mathbf{w}))^{\frac{1}{\tau}}}.$$

We use notation f^τ to denote a probability mass function f that is transferred by a temperature function with temperature τ . When the temperature is high, $\tau > 1$, the distribution becomes more uniform; when the temperature is low, $\tau < 1$, the distribution appears more spiky.

3.3. Tempered Policy Gradient Methods

Here, we introduce three instances of TPGs in the domain of goal-oriented dialogues, including single, parallel, and dynamic tempered policy gradient methods.

Single-TPG: The Single-TPG method simply uses a global temperature τ_{global} during the whole training process, i.e., we use $\tau_{global} > 1$ to encourage exploration. The forward pass is represented mathematically as: $y^{\tau_{global}} \sim f^{\tau_{global}}(\pi(\mathbf{x} | \mathbf{w}))$, where $\pi(\mathbf{x} | \mathbf{w})$ represents a policy neural network that parametrizes a distribution $f^{\tau_{global}}$ over the vocabulary, and $y^{\tau_{global}}$ means the word sampled from this tempered word distribution. After sampling, the weight of the neural net is updated using,

$$\Delta w_i = \alpha_i(r - b_i) \partial \ln f(y^{\tau_{global}} | \pi(\mathbf{x} | \mathbf{w})) / \partial w_i.$$

Noteworthy is that the actual gradient, $\partial \ln f(y^{\tau_{global}} | \pi(\mathbf{x} | \mathbf{w})) / \partial w_i$, depends on the sampled word, $y^{\tau_{global}}$, however, does not depend directly on the temperature, τ . With Single-TPG and $\tau > 1$, the entire vocabulary of a dialogue agent is explored more efficiently than by REINFORCE, because non-preferred words have a higher probability of being explored.

Parallel-TPG: A more advanced version of Single-TPG is the Parallel-TPG that deploys several Single-TPGs concurrently with different temperatures, τ_1, \dots, τ_n , and updates the weights based on all generated samples. During the forward pass, multiple copies of the neural nets parameterize multiple word distributions. The words are sampled in parallel at various temperatures, mathematically, $y^{\tau_1}, \dots, y^{\tau_n} \sim f^{\tau_1, \dots, \tau_n}(\pi(\mathbf{x} | \mathbf{w}))$. After sampling, in the backward pass the weights are updated with the sum of gradients. The formula is given by

$$\Delta w_i = \sum_k \alpha_i(r - b_i) \partial \ln f(y^{\tau_k} | \pi(\mathbf{x} | \mathbf{w})) / \partial w_i,$$

where $k \in \{1, \dots, n\}$. The combinational use of higher and lower temperatures ensures both exploration and exploitation

at the same time. The sum over weight updates of parallel policies gives a more accurate Monte Carlo estimate of $\partial E(r \mid \mathbf{w}) / \partial w_i$, due to the nature of Monte Carlo methods [24]. Thus, compared to Single-TPG, we would argue that Parallel-TPG is more robust and stable, although Parallel-TPG needs more computational power. However, these computations can easily be distributed in a parallel fashion using state-of-the-art graphics processing units.

Dynamic-TPG: As a third variant, we introduce the Dynamic-TPG, which is the most sophisticated approach in the current TPG family. The essential idea is that we use a heuristic function h to assign the temperature τ to the word distribution at each time step, t . The temperature is bounded in a predefined range $[\tau_{min}, \tau_{max}]$. The heuristic function we used here is based upon the term frequency inverse document frequency, *tf-idf* [25]. In the context of goal-oriented dialogues, we use the counted number of each word as term frequency *tf* and the total number of generated dialogues during training as document frequency *df*. We use the word that has the highest probability to be sampled at current time-step, y_t^* , as the input to the heuristic function h . Here, y_t^* is the maximizer of the probability mass function f . Mathematically, it is defined as $y_t^* = \operatorname{argmax}(f(\pi(\mathbf{x} \mid \mathbf{w})))$. We propose that $tf\text{-idf}(y_t^*)$ approximates the concentration level of the distribution, which means that if the same word is always sampled from a distribution, then the distribution is very concentrated. Too much concentration prevents the model from exploration, so that a higher temperature is needed. In order to achieve this effect, the heuristic function is defined as

$$\begin{aligned} \tau_t^h &= h(tf\text{-idf}(y_t^*)) \\ &= \tau_{min} + (\tau_{max} - \tau_{min}) \frac{tf\text{-idf}(y_t^*) - tf\text{-idf}_{min}}{tf\text{-idf}_{max} - tf\text{-idf}_{min}}. \end{aligned}$$

With this heuristic, words that occur very often are depressed by applying a higher temperature to those words, making them less likely to be selected in the near future. In the forward pass, a word is sampled using $y_t^{\tau_t^h} \sim f^{\tau_t^h}(\pi(\mathbf{x} \mid \mathbf{w}))$. In the backward pass, the weights are updated correspondingly:

$$\Delta w_i = \alpha_i (r - b_i) \partial \ln f(y_t^{\tau_t^h} \mid \pi(\mathbf{x} \mid \mathbf{w})) / \partial w_i,$$

where τ_t^h is the temperature calculated from the heuristic function. Compared to Parallel-TPG, the advantage of Dynamic-TPG is that it assigns temperature more appropriately, without increasing the computational load.

4. GUESSWHAT?! GAME

We evaluate our methods using a recent testbed for AI, called the GuessWhat?! game [6], available at <https://guesswhat.ai>. The dataset consists of 155k dialogues, including 822k question-answer pairs, each composed of around 5k words, about 67k images [26] and 134k

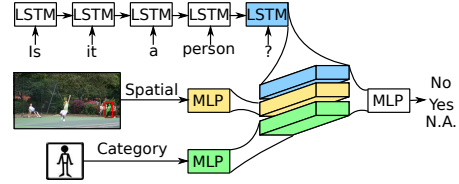


Fig. 1: Oracle model

objects. The game is about visual object discovery through a multi-round QA among different players.

Formally, a GuessWhat?! game is represented by a tuple (I, D, O, o^*) , where $I \in \mathbb{R}^{H \times W}$ denotes an image of height H and width W ; D represents a dialogue composed of M rounds of question-answer pairs (QAs), $D = (\mathbf{q}_m, a_m)_{m=1}^M$; O stands for a list of K objects $O = (o_k)_{k=1}^K$; and o^* is the target object. Each question is a sequence of words, $\mathbf{q}_m = \{y_{m,1}, \dots, y_{m,N_m}\}$ with length N_m . The words are taken from a defined vocabulary V , which consists of the words and a start token and an end token. Each answer is either yes, no, or not applicable, i.e. $a_m \in \{yes, no, n.a.\}$. For each object o_k , there is a corresponding object category $c_k \in \{1, \dots, C\}$ and a pixel-wise segmentation mask $S_k \in \{0, 1\}^{H \times W}$. Finally, we use colon notation ($:$) to select a subset of a sequence, for instance, $(\mathbf{q}, a)_{1:m}$ refers to the first m rounds of QAs in a dialogue.

4.1. Models and Pretraining

Following [8], we first train all three models in a supervised fashion.

Oracle: The task of the Oracle is to answer questions regarding to the target object. We outline here the simple neural network architecture that achieved the best performance in the study of [6], and which we also used in our experiments. The input information used here is of three modalities, namely the question \mathbf{q} , the spatial information $x^*_{spatial}$ and the category c^* of the target object. For encoding the question, de Vries et al. first use a lookup table to learn the embedding of words, then use a one layer long-short-term-memory (LSTM) [27] to encode the whole question. For spatial information, de Vries et al. extract an 8-dimensional vector of the location of the bounding box $[x_{min}, y_{min}, x_{max}, y_{max}, x_{center}, y_{center}, w_{box}, h_{box}]$, where x, y denote the coordinates and w_{box}, h_{box} denote the width and height of the bounding box, respectively. De Vries et al. normalize the image width and height so that the coordinates range from -1 to 1. The origin is at the image center. The category embedding of the object is also learned with a lookup table during training. At the last step, de Vries et al. concatenate all three embeddings into one feature vector and fed it into a one hidden layer multilayer perceptron (MLP). The softmax output layer predicts the distribution, $\text{Oracle} := p(a \mid \mathbf{q}, c^*, x^*_{spatial})$, over the three classes,

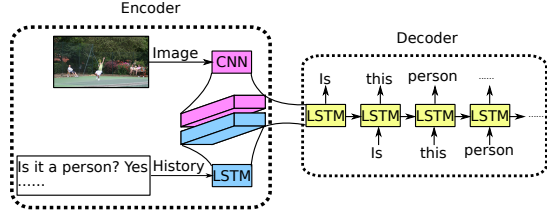


Fig. 2: Question-Generator model

including no, yes, and not applicable. The model is trained using the negative log-likelihood criterion. The Oracle structure is shown in Fig. 1.

Question-Generator: The goal of the Question-Generator (QGen) is to ask the Oracle meaningful questions, \mathbf{q}_{m+1} , given the whole image, I , and the historical question-answer pairs, $(\mathbf{q}, a)_{1:m}$. In previous work [8], the state transition function was modelled as an LSTM, which was trained using whole dialogues so that the model memorizes the historical QAs. We refer to this as dialogue level training. We develop a novel QGen architecture using a modified version of the Seq2Seq model [16]. The modified Seq2Seq model enables *question level training*, which means that the model is fed with historical QAs, and then learns to produce a new question. Following [8], we first encode the whole image into a fixed-size feature vector using the VGG-net [28]. The features come from the fc-8 layer of the VGG-net. For processing historical QAs, we use a lookup table to learn the word embeddings, then again use an LSTM encoder to encode the history information into a fixed-size latent representation, and concatenate it with the image representation:

$$\mathbf{s}_{m,N_m}^{enc} = \text{encoder}(\text{LSTM}(\mathbf{q}, a)_{1:m}, \text{VGG}(I)).$$

The encoder and decoder are coupled by initializing the decoder state with the last encoder state, mathematically, $\mathbf{s}_{m+1,0}^{dec} = \mathbf{s}_{m,N_m}^{enc}$. The LSTM decoder generates each word based on the concatenated representation and the previous generated word (note the first word is a start token):

$$y_{m+1,n} = \text{decoder}(\text{LSTM}((y_{m+1,n-1}, \mathbf{s}_{m+1,n-1}^{dec}))).$$

The decoder shares the same lookup table weights as the encoder. The Seq2Seq model, consisting of the encoder and the decoder, is trained end-to-end to minimize the negative log-likelihood cost. During testing, the decoder gets a start token and the representation from the encoder, and then generates each word at each time step until it encounters a question mark token, $\text{QGen} := p(y_{m+1,n} | (\mathbf{q}, a)_{1:m}, I)$. The output is a complete question. After several question-answer rounds, the QGen outputs an end-of-dialogue token, and stops asking questions. The overall structure of the QGen model is illustrated in Fig. 2.

Guesser: The goal of the Guesser model is to find out which object the Oracle model is referring to, given the com-

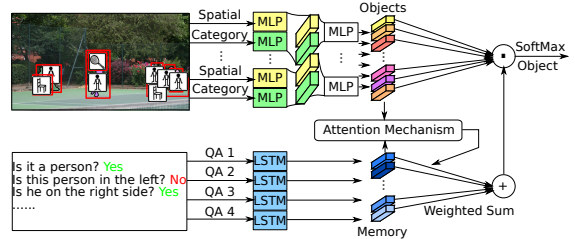


Fig. 3: Guesser model

plete history of the dialogue and a list of objects in the image, $p(o^* | (\mathbf{q}, a)_{1:M}, x_{spatial}^O, c^O)$. The Guesser model has access to the spatial, $x_{spatial}^O$, and category information, c^O , of the objects in the list. The task of the Guesser model is challenging because it needs to understand the dialogue and to focus on the important content, and then guess the object. To accomplish this task, we decided to integrate the Memory [17] and Attention [18] modules into the Guesser architecture used in the previous work [8]. First, we use an LSTM header to process the varying lengths of question-answer pairs in parallel into multiple fixed-size vectors. Here, each QA-pair has been encoded into some facts, $\text{Fact}_m = \text{LSTM}((\mathbf{q}, a)_m)$, and stored into a memory base. Later, we use the sum of the spatial and category embeddings of all objects as a key, $\text{Key}_1 = \text{MLP}(x_{spatial}^O, c^O)$, to query the memory and calculate an attention mask, $\text{Attention}_1(\text{Fact}_m) = \text{Fact}_m \odot \text{key}_1$, over each fact. Next, we use the sum of attended facts and the first key to calculate the second key. Further, we use the second key to query the memory base again to have a more accurate attention. These are the so called “two-hops” of attention in the literature [17]. Finally, we compare the attended facts with each object embedding in the list using a dot product. The most similar object to these facts is the prediction, $\text{Guesser} := p(o^* | (\mathbf{q}, a)_{1:M}, x_{spatial}^O, c^O)$. The intention of using the attention module here is to find out the most relevant descriptions or facts concerning the candidate objects. We train the whole Guesser network end-to-end using the negative log-likelihood criterion. A more graphical description of the Guesser model is shown in Fig. 3.

4.2. Reinforcement Learning

Now, we post-train the QGen and the Guesser model with reinforcement learning. We keep the Oracle model fixed. In each game episode, when the models find the correct object, $r = 1$, otherwise, $r = 0$.

Next, we can assign credits for each action of the QGen and the Guesser models. In the case of the QGen model, we spread the reward uniformly over the sequence of actions in the episode. The baseline function, b , used here is the running average of the game success rate. Consider that the Guesser model has only one action in each episode, i.e., taking the guess. If the Guesser finds the correct object, then it gets an

immediate reward and the Guesser’s parameters are updated using the REINFORCE rule without baseline. The QGen is trained using the following four methods.

REINFORCE: The baseline method used here is REINFORCE [15]. During training, in the forward pass the words are sampled with $\tau = 1$, $y_{m+1,n} \sim f(\text{QGen}(\mathbf{x} \mid \mathbf{w}))$. In the backward pass, the weights are updated using REINFORCE, $\mathbf{w} = \mathbf{w} + \alpha(r - b)\nabla_{\mathbf{w}}\ln f(y_{m+1,n} \mid \text{QGen}(\mathbf{x} \mid \mathbf{w}))$.

Single-TPG: We use temperature $\tau_{global} = 1.5$ during training to encourage exploration, mathematically, $y_{m+1,n}^{\tau_{global}} \sim f^{\tau_{global}}(\text{QGen}(\mathbf{x} \mid \mathbf{w}))$. In the backward pass, the weights are updated using $\mathbf{w} = \mathbf{w} + \alpha(r - b)\nabla_{\mathbf{w}}\ln f(y_{m+1,n}^{\tau_{global}} \mid \text{QGen}(\mathbf{x} \mid \mathbf{w}))$.

Parallel-TPG: For Parallel-TPG, we use two temperatures $\tau_1 = 1.0$ and $\tau_2 = 1.5$ to encourage the exploration. The words are sampled in the forward pass using $y_{m+1,n}^{\tau_1}, y_{m+1,n}^{\tau_2} \sim f^{\tau_1, \tau_2}(\text{QGen}(\mathbf{x} \mid \mathbf{w}))$. In the backward pass, the weights are updated using $\mathbf{w} = \mathbf{w} + \sum_{k=1}^2 \alpha(r - b)\nabla_{\mathbf{w}}\ln f(y_{m+1,n}^{\tau_k} \mid \text{QGen}(\mathbf{x} \mid \mathbf{w}))$.

Dynamic-TPG: The last method we evaluated is Dynamic-TPG. We use a heuristic function to calculate the temperature for each word at each time step: $\tau_{m+1,n}^h = \tau_{min} + (\tau_{max} - \tau_{min}) \frac{tf-idf(y_{m+1,n}^* - tf-idf_{min})}{tf-idf_{max} - tf-idf_{min}}$, where we set $\tau_{min} = 0.5$, $\tau_{max} = 1.5$, and set $tf-idf_{min} = 0$, $tf-idf_{max} = 8$. After the calculation of $\tau_{m+1,n}^h$, we substitute the value into the formula at each time step and sample the next word using $y_{m+1,n}^{\tau_{m+1,n}^h} \sim f^{\tau_{m+1,n}^h}(\text{QGen}(\mathbf{x} \mid \mathbf{w}))$. In the backward pass, the weights are updated using $\mathbf{w} = \mathbf{w} + \alpha(r - b)\nabla_{\mathbf{w}}\ln f(y_{m+1,n}^{\tau_{m+1,n}^h} \mid \text{QGen}(\mathbf{x} \mid \mathbf{w}))$. For all four methods, we use greedy search in evaluation.

5. EXPERIMENT

We first train all the networks in a supervised fashion, and then optimize the QGen and the Guesser model using reinforcement learning. Our implementation ¹ uses Torch [29].

5.1. Pretraining

We train all three models using 0.5 dropout [30] during training, using the ADAM optimizer [31]. We use a learning rate of 0.0001 for the Oracle model and the Guesser model, and a learning rate of 0.001 for QGen. All the models are trained with at most 30 epochs and early stopped within five epochs without improvement on the validation set. We report the performance on the test set which consists of images not used in training. We report the game success rate as the performance metric for all three models, which equals to the number of succeeded games divided by the total number of all games. Compared to previous works [6, 8, 32], after supervised training, our models obtain a game success rate of 48.77%, that

#	Method	Accuracy
1	Strub et al., 2017 [8]	52.30%
2	Strub and de Vries, 2017 [32]	60.30%
3	Our Torch reimplementation of (# 2)	62.61%
4	(# 3) + new QGen (Seq2Seq)	63.47%
5	(# 4) + new Guesser (Memory Nets)	68.32%
6	(# 5) + new Guesser (REINFORCE)	69.66%
7	(# 6) + Single-TPG	69.76%
8	(# 6) + Parallel-TPG	73.86%
9	(# 6) + Dynamic-TPG	74.31%

Table 1: Performance comparison and ablation tests

is 4% higher than state-of-the-art methods [32], which has 44.6% accuracy.

5.2. Reinforcement Learning

We first initialize all models with pre-trained parameters from supervised learning and then post-train the QGen using either REINFORCE or TPG for 80 epochs. We update the parameters using stochastic gradient descent (SGD) with a learning rate of 0.001 and a batch size of 64. In each epoch, we sample each image in the training set once and randomly pick one of the objects as a target. We track the running average of the game success rate and use it directly as the baseline, b , in REINFORCE. We limit the maximum number of questions to 8 and the maximum number of words to 12. Simultaneously, we train the Guesser model using REINFORCE without baseline and using SGD with a learning rate of 0.0001. The performance comparison is shown in Tab. 1.

Ablation Study: From Tab. 1 (# 2 & 3), we see that our reimplementation using Torch [29] achieves a comparable performance compared to the original TensorFlow implementation [32]. We use our reimplementation as the baseline.

Upon the baseline, the new QGen model with Seq2Seq structure improves the performance by about 1%, see Tab. 1 (# 3 & 4). With the Seq2Seq structure, our QGen model is trained in *question level*. This means that the model first learns to query meaningfully, step by step. Eventually, it learns to conduct a meaningful dialog. Compared to directly learning to manage a strategic conversation, this bottom-up training procedure helps the model absorb knowledge, because it breaks large tasks down into smaller, more manageable pieces. This makes the learning for QGen much easier.

The next improvement is because of our new Guesser model, which uses Memory Network with two-hops attention [17]. The memory and attention mechanisms bring an improvement of 4.85%, as shown in Tab. 1 (# 4 & 5). Furthermore, we train the new Guesser model additionally via REINFORCE (# 6). In this way, the Guesser and the QGen learn to cooperate with each other and improve the performance by another 1.34%, as shown in Tab. 1 (# 5 & 6).

¹<https://github.com/ruiyhaogit/GuessWhat-TemperedPolicyGradient>



Image	Policy Gradient		Tempered Policy Gradient	
	Is it in left?	No	Is it a person?	No
	Is it in front?	No	Is it a vehicle ?	Yes
	Is it in right?	Yes	Is it a truck ?	Yes
	Is it in middle?	Yes	Is it in front of photo?	No
	Is it person?	No	In the left half?	No
	Is it ball?	No	In the middle of photo?	Yes
	Is it bat?	No	Is it to the right photo?	Yes
	Is it car ?	Yes	Is it in the middle of photo?	Yes
	Status:	Failure	Status:	Success
		Is it in left ?	No	Is it a giraffe?
Is it in front?		Yes	In front of photo?	Yes
Is it in right?		No	In the left half ?	Yes
Is it in middle?		Yes	Is it in the middle of photo?	Yes
Is it person?		No	Is it to the left of photo?	Yes
Is it giraffe?		Yes	Is it to the right photo?	No
Is in middle?		Yes	In the left in photo?	No
Is in middle?		Yes	In the middle of photo?	Yes
Status:		Failure	Status:	Success

Table 2: Some samples generated by our improved models using REINFORCE (left column: “Policy Gradient”) and Dynamic-TPG (right column: “Tempered Policy Gradient”). The green bounding boxes highlight the target objects; the red boxes highlight the wrong guesses.

Here, we take a closer look at the improvement brought by TPGs. From Tab. 1, we see that compared to the REINFORCE-trained models (#6), Single-TPG (#7) with $\tau_{global} = 1.5$ achieves a comparable performance. With two different temperatures $\tau_1 = 1.0$ and $\tau_2 = 1.5$, Parallel-TPG (#8) achieves an improvement of approximately 4%. Parallel-TPG requires more computational resources. Compared to Parallel-TPG, Dynamic-TPG only uses the same computational power as REINFORCE does and still gives a larger improvement by using a dynamic temperature, $\tau_t^h \in [0.5, 1.5]$. After comparison, we can see that the best model is Dynamic-TPG (#9), which gives a 4.65% improvement upon new models (#6).

TPG Dialogue Samples: The generated dialogue samples in Tab. 2 can give some interesting insights. First of all, the sentences generated from TPG-trained models are on average longer and use slightly more complex structures, such as “Is it in the middle of photo?” instead of a simple form “Is it in middle?”. Secondly, TPGs enable the models to explore better and comprehend more words. For example, in the first task (upper half of Tab. 2), both models ask about the category. The REINFORCE-trained model can only ask with the single word “car” to query about the vehicle category. In contrast, the TPG-trained model can first ask a *more general* category with the word “vehicle” and follows up querying with a *more specific* category “trucks”. These two words “vehicle” and “trucks” give much more information than the single word “car”, and help the Guesser model identify the truck among many cars. Lastly, similar to the category case, the models trained with TPG can first ask a *larger* spatial range

of the object and follow up with a *smaller* range. In the second task (lower half of Tab. 2), we see that the TPG-trained model first asks “In the left half?”, which refers to all the three giraffes in the left half, and the answer is “Yes”. Then it asks “Is it to the left of photo?”, which refers to the second left giraffe, and the answer is “Yes”. Eventually the QGen asks “In the left in photo?”, which refers to the most left giraffe, and the answer is “No”. These specific questions about locations are not learned using REINFORCE. The REINFORCE-trained model can only ask a similar question with the word “left”. In this task, there are many giraffes in the left part of the image. The top-down spatial questions help the Guesser model find the correct giraffe. To summarize, the TPG-trained models use longer and more informative sentences than the REINFORCE-trained models.

6. CONCLUSION

Our paper makes two contributions. Firstly, by extending existing models with Seq2Seq and Memory Networks we could improve the performance of a goal-oriented dialogue system by 7%. Secondly, we introduced TPG, a novel class of temperature-based policy gradient approaches. TPGs boosted the performance of the goal-oriented dialogue systems by another 4.7%. Among the three TPGs, Dynamic-TPG gave the best performance, which helped the agent comprehend more words, and produce more meaningful questions. TPG is a generic strategy to encourage word exploration on top of policy gradients and can be applied to any dialog agents.

7. REFERENCES

- [1] Rui Zhao and Volker Tresp, “Improving goal-oriented visual dialog agents via advanced recurrent nets with tempered policy gradient,” *arXiv preprint arXiv:1807.00737*, 2018.
- [2] Richard S Sutton and Andrew G Barto, *Reinforcement learning: An introduction*, MIT press, 1998.
- [3] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al., “Mastering the game of go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
- [4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al., “Human-level control through deep reinforcement learning,” *Nature*, 2015.
- [5] Antoine Bordes and Jason Weston, “Learning end-to-end goal-oriented dialog,” in *International Conference on Learning Representations (ICLR)*, 2017.
- [6] Harm de Vries, Florian Strub, Sarath Chandar, Olivier Pietquin, Hugo Larochelle, and Aaron C. Courville, “Guesswhat?! visual object discovery through multi-modal dialogue,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [7] Abhishek Das, Satwik Kottur, José M.F. Moura, Stefan Lee, and Dhruv Batra, “Learning cooperative visual dialog agents with deep reinforcement learning,” in *International Conference on Computer Vision (ICCV)*, 2017.
- [8] Florian Strub, Harm de Vries, Jérémie Mary, Bilal Piot, Aaron C. Courville, and Olivier Pietquin, “End-to-end optimization of goal-driven and visually grounded dialogue systems,” in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2017.
- [9] Zachary Lipton, Xiujun Li, Jianfeng Gao, Lihong Li, Faisal Ahmed, and Li Deng, “Bbq-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems,” *arXiv preprint arXiv:1711.05715*, 2017.
- [10] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky, “Deep reinforcement learning for dialogue generation,” *arXiv preprint arXiv:1606.01541*, 2016.
- [11] Jason D Williams and Geoffrey Zweig, “End-to-end lstm-based dialog control optimized with supervised and reinforcement learning,” *arXiv preprint arXiv:1606.01269*, 2016.
- [12] Mike Lewis, Denis Yarats, Yann N Dauphin, Devi Parikh, and Dhruv Batra, “Deal or no deal? end-to-end learning for negotiation dialogues,” *arXiv preprint arXiv:1706.05125*, 2017.
- [13] Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng, “End-to-end reinforcement learning of dialogue agents for information access,” *arXiv preprint arXiv:1609.00777*, 2016.
- [14] Rui Zhao and Volker Tresp, “Efficient dialog policy learning via positive memory retention,” in *IEEE Spoken Language Technology (SLT) (forthcoming)*, 2018.
- [15] Ronald J Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” *Machine learning*, 1992.
- [16] Ilya Sutskever, Oriol Vinyals, and Quoc V Le, “Sequence to sequence learning with neural networks,” in *Advances in neural information processing systems*, 2014, pp. 3104–3112.
- [17] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al., “End-to-end memory networks,” in *Advances in neural information processing systems*, 2015, pp. 2440–2448.
- [18] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, “Neural machine translation by jointly learning to align and translate,” *arXiv preprint arXiv:1409.0473*, 2014.
- [19] David Carmel and Shaul Markovitch, “Exploration strategies for model-based learning in multi-agent systems: Exploration strategies,” *Autonomous Agents and Multi-agent systems*, vol. 2, no. 2, pp. 141–172, 1999.
- [20] Ofir Nachum, Mohammad Norouzi, and Dale Schuurmans, “Improving policy gradient by exploring underappreciated rewards,” *arXiv preprint arXiv:1611.09321*, 2016.
- [21] Yang Liu, Prajit Ramachandran, Qiang Liu, and Jian Peng, “Stein variational policy gradient,” *arXiv preprint arXiv:1704.02399*, 2017.
- [22] Sebastian B Thrun, “Efficient exploration in reinforcement learning,” 1992.
- [23] Andrej Karpathy and Li Fei-Fei, “Deep visual-semantic alignments for generating image descriptions,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3128–3137.
- [24] Christian P Robert, *Monte carlo methods*, Wiley Online Library, 2004.
- [25] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman, *Mining of massive datasets*, Cambridge university press, 2014.

- [26] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick, “Microsoft coco: Common objects in context,” in *European conference on computer vision*. Springer, 2014, pp. 740–755.
- [27] Sepp Hochreiter and Jürgen Schmidhuber, “Long short-term memory,” *Neural computation*, 1997.
- [28] Karen Simonyan and Andrew Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [29] R. Collobert, K. Kavukcuoglu, and C. Farabet, “Torch7: A matlab-like environment for machine learning,” in *BigLearn, NIPS Workshop*, 2011.
- [30] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.
- [31] Diederik Kingma and Jimmy Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [32] Florian Strub and Harm de Vries, “Guesswhat?! models,” <https://github.com/GuessWhatGame/guesswhat/>, 2017.

Chapter 3

Efficient Dialog Policy Learning via Positive Memory Retention

EFFICIENT DIALOG POLICY LEARNING VIA POSITIVE MEMORY RETENTION

Rui Zhao, Volker Tresp

Ludwig Maximilian University, Oettingenstr. 67, 80538 Munich, Germany

Siemens AG, Corporate Technology, Otto-Hahn-Ring 6, 81739 Munich, Germany

ABSTRACT

This paper is concerned with the training of recurrent neural networks as goal-oriented dialog agents using reinforcement learning. Training such agents with policy gradients typically requires a large amount of samples. However, the collection of the required data in form of conversations between chat-bots and human agents is time-consuming and expensive. To mitigate this problem, we describe an efficient policy gradient method using positive memory retention, which significantly increases the sample-efficiency. We show that our method is 10 times more sample-efficient than policy gradients in extensive experiments on a new synthetic number guessing game. Moreover, in a real-world visual object discovery game, the proposed method is twice as sample-efficient as policy gradients and shows state-of-the-art performance.

Index Terms— Goal-Oriented Dialog System, Deep Reinforcement Learning, Recurrent Neural Network

1. INTRODUCTION

In recent years, advances in Deep Learning (DL) and Reinforcement Learning (RL) have led to tremendous progress across many areas of natural language processing (NLP) and gameplay [1, 2, 3, 4, 5]. This progress, in turn, generated an emerging research area, the learning of goal-oriented dialogs [6]. This research involves agents that conduct a multi-turn dialogue to achieve some task-specific goal, such as locating a specific object in a group of objects [7], inferring which image the user is thinking about [8], and providing customer services and restaurant reservations [6]. All these tasks require that the agent possesses the ability to conduct a multi-round dialog and to track the inter-dependence of each question-answer pair. Eventually, the agent learns an optimal policy through trial-and-error. The reward signal of each trail is delayed, and is only available at the end of the dialog. Also the reward signal is very sparse compared with a vocabulary size that often exceeds several thousands. Due to these challenges, in practice, policy gradient methods [9] perform more favorably than Q-learning methods [10].

Consider a simple goal-oriented dialog example from our synthetic dataset in Figure 1. We initialize three roles in this number guessing game, i.e. a *questioner*, an *answerer*, and a

#	Question	Answer
1	Is it 2 in the image?	No
2	Is it in a yellow background?	No
3	Is it 9 in the image?	Yes
4	Is it in a white background?	Yes
5	Is it a stroke style digit?	Yes
6	Is it a digit in blue?	No
	Guess: row 1 column 3	✓

Fig. 1: MNIST GuessNumber dataset example: Each sample consists of an image (left), a set of sequential questions with answers (right), and a target digit. The goal of this game is to find out the target digit by a multi-round question-answering.

guesser. The questioner and the guesser try to infer which number the answerer is thinking about. First, the questioner asks questions about the target digit given the image, such as the color of the digit, the background color, the style of the digit, and also the number itself. Then the answerer responds with a yes/no answer. The questioner needs to reason based on the history dialog and keeps querying with meaningful questions. At the end, when the maximum number of questions is reached, the guesser analyzes the whole conversation along with the image, and takes a guess. If the guess is correct, then the task is completed successfully, and the questioner gets a positive reward signal. Otherwise the task is counted as a failure, and the questioner gets a non-positive reward signal.

The training of chat-bots using on-policy policy gradient methods requires numerous training samples. When the samples are generated through human-machine-interaction, e.g. by using the Amazon Mechanical Turk or in real-world applications, the collection of the data is time-consuming and expensive [11]. Hence, sample-efficiency receives increasingly more attention in dialog policy learning. We improve sample-efficiency using a novel on-off-policy policy gradient method relying on a biologically inspired mechanism [12], termed positive memory retention. This mechanism employs a bounded importance weight proposal on past positive trajectories, i.e. the behavior policy [13], to train the target policy network. The retention stops automatically when no further improvement occurs in a predefined number of iterations. The

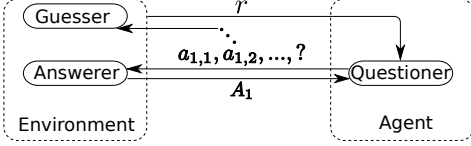


Fig. 2: GuessNumber Gameplay: The gameplay of the guessing game involves three plays, a questioner, an answerer, and a guesser. In our settings, we first pretrain all three models, then post-train only the questioner using RL. For each gameplay, the questioner first asks a question word by word, $a_{1,1}, a_{1,2}, \dots, ?$, then the answerer responds with an answer, A_1 . This question-answering repeats for a predefined number of rounds. Finally, the guesser reads in the dialog and makes a guess. If the guess is correct, then the questioner receives a reward $r = 1$, otherwise, $r = 0$.

bounded importance weight proposal tackles the problem of high variance in importance sampling. To reduce variance even further, we use recent behavior policies to update the probability in the memory buffer. An early-stopping mechanism within each epoch provides a trade-off between the sample-efficiency and the computational cost.

Contributions: (1) We introduce positive memory retention for efficient dialog policy learning, which uses bounded importance sampling, probability updating, and adaptively adjusts the retention times via early stopping within epochs. (2) We perform a comprehensive study about the performance of our method for goal-oriented dialog tasks using a new synthetic number guessing game and verify the high sample-efficiency of our algorithm. (3) The proposed model is also tested on a real-world benchmark GuessWhat?! game [14] and shows state-of-the-art performance, and an increased sample-efficiency by a factor of two.

2. BACKGROUND

This section introduces recurrent language models, Markov decision process, policy gradient, and importance sampling.

Recurrent Language Models: The goal of a recurrent neural network (RNN) based language model in NLP is to produce an output sequence $y = [a_1, a_2, \dots, a_T]$ given a context x as input [15]. Here $a_i \in \mathcal{A}$ where \mathcal{A} is the word vocabulary. For each step, the recurrent unit processes the previous word along with the context, and outputs a new word. At each time step t , the state s_t is the context input x and the words $y_{t-1} = [a_1, \dots, a_{t-1}]$ produced by the RNN so far, i.e. $s_t = (x, y_{t-1})$. We sample the next word a_t from this probability distribution $\pi(\cdot|s_t)$, then update our state $s_{t+1} = (x, y_t)$ where $y_t = [y_{t-1}, a_t]$, and repeat in a similar fashion.

Markov Decision Process and Policy Gradient: We formalize a simplified Markov decision process (MDP) to our setting. In the MDP, an agent takes an action a in a state s and transitions to a new state s' . A trajectory τ refers to a sequence of transitions until the agent enters a terminal state where it receives a reward from the environment.

In our guessing games, a trajectory τ is $(x, a_{1,1}, a_{1,2}, \dots, A_1, a_{2,1}, a_{2,2}, \dots, A_2, \dots, G, r)$, where x is the context (image); a_i is the word sequence in question i ; $?$ is the question mark that only occurs at the end of each question; A_i is the answer to question i ; G is the output of the guesser; r is the reward for being correct or incorrect. See also Figure 2.

Formally, the simplified MDP is a triple of $(\mathcal{S}, \mathcal{A}, R)$ where \mathcal{S} , \mathcal{A} , and R represent a set of states, actions, and rewards, respectively. A policy π is a function that chooses an action at a given state, e.g. $\pi : \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$, where $\pi(a|s)$ refers to the probability of executing action a at the state s . When we sample an action $a_t \sim \pi(\cdot|s_t)$, we transition into state $(x, [y_{t-1}, a_t])$. We overload notation and let $R(\tau) = \sum_{t=1}^T R(s_t)$ be the accumulated reward of a trajectory τ . We define that $R(\tau) = 1$, if the task is a success; $R(\tau) = 0$, if the task is a failure. The policy network π is a recurrent language model, parametrized by a vector $\theta \in \mathbb{R}^n$, i.e. π_θ . The expected return of a policy π_θ is:

$$J(\theta) = \mathbb{E}[R(\tau)|\pi_\theta].$$

Our goal is to learn θ to maximize the expectation of the return $J(\theta)$. The objective function can be optimized with an on-policy policy gradient method, known as REINFORCE [9]. The gradient is calculated as:

$$\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}[(R(\tau) - b) \log \pi_\theta(a_t|s_t)]$$

where b is an optional baseline function used to reduce the variance of the gradient estimate [9].

Importance Sampling: Importance sampling (IS) is a general technique to estimate an integral $\int f(x)p(x)dx$ of a function $f(x)$, with distribution $p(x)$ [16]. IS samples from an appropriate proposal distribution $q(x)$, and then uses the samples to estimate the integral:

$$I = \mathbb{E}[f] = \int f(x) \frac{p(x)}{q(x)} q(x) dx \approx \frac{1}{N} \sum_{n=1}^N \omega_n f(x^{(n)}) = \hat{I},$$

where $\omega_n = p(x^{(n)})/q(x^{(n)})$ are the importance weights. The goal is to minimize the variance of the estimate \hat{I} , which is proportional to $\text{var}_q[f(x)\omega(x)] = \mathbb{E}_q[f^2(x)\omega^2(x)] - I^2$. Since the last term is independent of q , we can ignore it. Using Jensen's inequality, we have the following lower bound:

$$\mathbb{E}_q[f^2(x)\omega^2(x)] \geq (\mathbb{E}_q[|f(x)\omega(x)|])^2 = \left(\int |f(x)|p(x)dx \right)^2$$

The lower bound is obtained when using the optimal importance distribution: $q^*(x) = |f(x)|p(x)/\int |f(x')|p(x')dx'$. Interestingly, to estimate an integral, it is more efficient to sample from $q(x) \propto |f(x)|p(x)$ than to sample from $p(x)$. Of course, if $f(x)$ is unknown, it is best to sample from $q(x) = p(x)$ [17].

3. POSITIVE MEMORY RETENTION

This section contains our main contribution, the positive memory retention method. The reuse of past positive trajectories in policy learning is enabled by several contributions: first, sample efficiency is improved by concentration of past positive trajectories; secondly, the sampling bias is corrected by importance sampling in policy gradients; thirdly, the stability is ensured by introducing bounds on the important weights; fourthly, the variance is reduced by probability updating of the proposal distribution; finally, the stability is improved by policy search via early stopping.

Positive Memory Matters: In human memory retention, focusing on *rewarded* events has been discovered to be a preferred strategy in the post-learning phase happening in the hippocampus area of the brain [12]. We believe that this fact also intuitively applies to RL since non-rewarded trajectories do not contribute directly to the estimated gradient to increase the expected return, $\nabla_{\theta} J(\theta)$, since $R(\tau)$ is zero.

In a more general case, consider the policy updating with a baseline function, e.g. $0 < b < 1$. The gradient of non-rewarded trajectories is opposite to the direction of the gradient of the current policy, $\nabla_{\theta} \log \pi_{\theta}(a_t | s_t)$, because $R(\tau) - b < 0$. This means that the weights are changed in a way to depress the current policy π_{θ} , which is not necessarily equivalent to maximizing the expected return. However, it might be helpful, in a way to encourage exploration.

The *high efficiency* of positive memory retention can also be derived from importance sampling. Consider that the expectation we want to estimate is the expected return $\mathbb{E}[R(\tau)]$, so $R(\tau)$ assumes the role of $f(x)$ in Section 2. The proposal distribution q should thus be of the form $q(\tau) \propto R(\tau)p(\tau)$, thus we only need to consider successful memory trajectories.

Policy Gradient with Importance Sampling: However, memory trajectories cannot be directly applied to policy gradient methods. The main reason is that the training requires trajectories from the target policy $p(\tau | \pi_{\theta})$ with the current parameter vector θ , whereas the memory trajectories were generated by $q(\tau | \pi_{\theta'}) = p(\tau | \pi_{\theta'})$ with a different parameter vector θ' . We again can use the concept of IS and obtain

$$\hat{J}(\theta) = \frac{1}{n} \sum_{i=1}^n \frac{p(\tau^{(i)} | \pi_{\theta})}{q(\tau^{(i)} | \pi_{\theta'})} R(\tau^{(i)}), \text{ with } \tau^{(i)} \sim q$$

where n is the number of trajectories used to estimate the expected return $J(\theta)$ [18]. In the equation above, we assume $q(\tau) = 0 \Rightarrow p(\tau) = 0$. This is readily true, since each action is sampled from the defined action space \mathcal{A} . The importance weights are evaluated using:

$$\omega(\tau^{(i)}) = \frac{p(\tau^{(i)} | \pi_{\theta})}{q(\tau^{(i)} | \pi_{\theta'})} = \frac{\prod_{t=1}^T \pi_{\theta}(a_t | s_t)}{\prod_{t=1}^T \pi_{\theta'}(a_t | s_t)}$$

where $\prod_{t=1}^T \pi_{\theta}(a_t | s_t)$ needs to be calculated from the target policy, and $\prod_{t=1}^T \pi_{\theta'}(a_t | s_t)$ has already been calculated from

the behavior policy. Finally, the importance weighted policy gradient is:

$$\nabla_{\theta} \hat{J}(\theta) = \nabla_{\theta} \mathbb{E}_q [\omega(\tau)(R(\tau) - b) \log \pi_{\theta}(a_t | s_t)]. \quad (1)$$

Bounded Importance Weight Proposal: This estimator is unbiased, but it suffers from very high variances because it involves a product of a series of unbounded importance weights. To prevent the importance weight from “exploding”, the goal is now to select only samples that are *not far* from the target policy.

To evaluate the distance, we use a symmetric version of the KL-divergence, i.e. the Jensen-Shannon divergence [19]:

$$JS(p, q) = 0.5 \mathbb{KL}(p \parallel 0.5(p + q)) + 0.5 \mathbb{KL}(q \parallel 0.5(p + q)).$$

We now derive a formulation of the JS-divergence, as a distance metric, which is related to the importance weight ω :

$$\begin{aligned} JS(p, q) &\approx 0.5 \sum_{k=1}^K p_k \log \frac{2p_k}{p_k + q_k} + 0.5 \sum_{k=1}^K q_k \log \frac{2q_k}{p_k + q_k} \\ &= 0.5 \sum_{k=1}^K p_k \log \frac{2}{1 + \omega_k} + 0.5 \sum_{k=1}^K q_k \log \frac{2}{\frac{1}{\omega_k} + 1} \end{aligned}$$

We can see that the distance between the proposal distribution q and the optimal solution p depends on both ω_k and $1/\omega_k$. To limit the variance of the importance sampling, we limit the importance weight as $\omega_k \leq \omega_{max}$ and its inverse as $1/\omega_k \leq \omega_{max}$. Subsequently, we define a trust region of importance weights, $\omega_k \in [1/\omega_{max}, \omega_{max}]$ and only use trajectories whose importance weights fall within this range.

Essentially, we use the importance weight ω as a value to select high quality trajectories, filtering out those that deviate far from the current policy. In this way, we shape the proposal distribution into a safe one. The bound ω_{max} of the distribution can be selected by observing the learning curve during training.

Probability Updating: Another way to reduce the variance is to *adapt* the proposal distribution, $q(x)$, to make it as close as possible to $p(x)$. After updating the target policy with Equation 1, we use the current target policy as a behavior policy for retention in the future. In this way, the memory buffer is being continuously updated and the proposal distribution is also updated.

Policy Search via Early Stopping: In order to make the best use of the memory, the learning process is verified using a group of validation samples. During the training process, the model remembers the positive trajectories within the current epoch for later retention. During the retention phase, the model first goes through the memory buffer, and updates the model using Equation 1 with the bounded importance weight proposal. After each iteration, the model verifies the learned policy on a validation set. If the policy becomes better than the previous policy, then it is saved. If the policy has not been

Algorithm 1 Positive Memory Retention (PMR)

Require: pretrained RNN language model π_θ

- 1: **for** iteration in range(max iterations) **do**
- 2: **for** $t = 1$ to T **do**
- 3: $(a_t, p_t) = \pi_\theta(x, y_{t-1}), y_t = (y_{t-1}, a_t)$
- 4: $r = R(y) \leftarrow$ Environment
- 5: **for** $t = T$ to 1 **do**
- 6: $\theta = \theta + \alpha(r - b)\nabla_\theta \log \pi_\theta(a_t|(x, y_{t-1}))$
- 7: **if** $r > 0$ **then**
- 8: memory $\leftarrow \tau = (x, y, p, r)$
- 9: validating(π_θ), $\theta' = \theta$
- 10: **for** trajectory τ in memory **do** # memory retention
- 11: $x, y, p, r \leftarrow \tau$
- 12: **for** $t = 1$ to T **do**
- 13: $q_t = p_t, (a_t, p_t) = \pi_\theta(x, y_{t-1})$
- 14: memory $\leftarrow p_t$ # probability updating
- 15: $\omega = \prod_{t=1}^T p_t / \prod_{t=1}^T q_t$
- 16: **if** $\omega \in [1/\omega_{max}, \omega_{max}]$ **then**
- 17: **for** $t = T$ to 1 **do**
- 18: $\theta = \theta + \alpha\omega(r - b)\nabla_\theta \log \pi_\theta(a_t|(x, y_{t-1}))$
- 19: validating(π_θ)
- 20: **if** no improvement for n_{max} iterations **then**
- 21: $\theta = \theta'$
- 22: **if** improvement **then**
- 23: $\theta' = \theta$

improved for a limited number of iterations n_{max} , then memory retention is stopped and the training of the model with REINFORCE continues. This mechanism helps the model make the best use of the past training samples and makes the learning more stable.

Complete Training Algorithm: We summarize the complete method in Algorithm 1. Note that, before training the language model with RL, we pretrained the model in a supervised fashion for a kickstart policy.

4. EXPERIMENTS

We conduct two sets of experiments to verify the proposed method. To highlight the methods' ability to boost sample-efficiency, we first create and experiment with a synthetic dataset¹. We then show that the method also works well on a real-word benchmark, GuessWhat?! [14].

4.1. MNIST GuessNumber Dataset

Experiment setting: We create a synthetic dataset, named MNIST GuessNumber, which is designed for quick testing and analysis of RL methods in the task of visual-grounded goal-oriented dialog systems. The creation of the dataset

is inspired by [20]. Each image in MNIST GuessNumber contains a 3×3 grid of MNIST digits and each MNIST digit in the grid has four randomly sampled attributes, i.e. color = {red, blue, green, purple, brown}, bgcolor = {cyan, yellow, white, silver, salmon}, number = $\{x | 0 \leq x \leq 9\}$ and style = {flat, stroke}, as illustrated in Figure 1.

Given the generated image from MNIST GuessNumber, we automatically generate questions and answers about a set of the digits in the grid that focus on one of the four attributes. During question generation, the target subset for a question is selected based on the previous target subset referred by the previous QA, as shown in Figure 1. For answer generation, we generate a yes/no answer based on whether the questioned attribute matches with the target digit. The QA is repeated until there is only one digit in the target subset. We generated 30 K, 10 K, and 10 K images for training, validating, and testing, respectively, and one successful game for each unique image. In each grid image, there are nine cells. Each cell contains four attributes, including the color, bgcolor, number, and the style.

Model details and pretraining: There are three roles in this number guessing game, a questioner, an answerer, and a guesser. The word and the image embeddings are trained end-to-end using lookup table layers. The questioner model that we used is a long short-term memory (LSTM) [21] of 256 units conditioned on a given image. The answerer model takes in the question along with the target digit and outputs a yes/no answer. The answer model is based on an LSTM with 64 units. The last one is the guesser model. The guesser uses an LSTM with 64 units to process the whole dialog, and compares it with each digit using a dot product on their respective latent representations. The prediction is the most similar digit. We train all three models for 30 epochs using the maximum likelihood criterion. The pretrained models obtain a game success rate of 63.09% on the test split with a maximum of four rounds of QA.

Reinforced training: After pretraining, we keep the answerer and the guesser fixed and train the questioner model with RL. Given the unique images in the training set, for each game, the answerer randomly picks a digit in the image as the target and lets the questioner ask. The baseline method is the REINFORCE. For positive memory retention, we set the parameter $\omega_{max} = 10$, and the early stopping threshold $n_{max} = 2$. The ω_{max} is selected on the validation set. An extensive evaluation of the impact of ω_{max} is shown in Table 1 lower part. When we use $\omega_{max} = 10$, we observe that about 65% of the positive trajectories are used for weight updating. So, ω_{max} can be considered as a trade-off between sample reuse ratio and the variance introduced by importance sampling. The early stopping threshold n_{max} is a trade-off between sample-efficiency and computational power. Our implementation² uses Torch7 [22].

Results: From Figure 3 left, we can see that at the 10th

¹<https://github.com/ruizhaogit/MNIST-GuessNumber>

²<https://github.com/ruizhaogit/PositiveMemoryRetention>

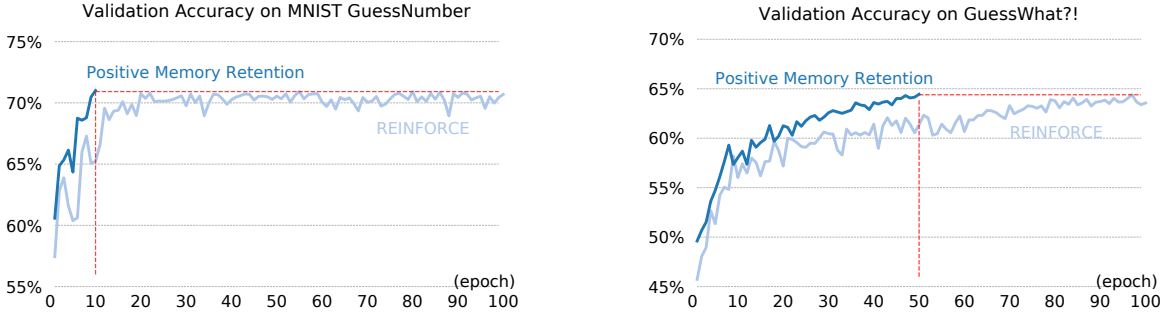


Fig. 3: Experimental results: The left figure shows that at the 10th epoch, the model trained with positive memory retention obtains about 70% accuracy on the validation set, which is equivalent to the same model trained with REINFORCE for 100 epochs, which obtains 69.92% accuracy. The right figure shows that the model trained with REINFORCE for 100 epochs, obtains 63.39% accuracy on the validation split. At the 50th epoch, the same model trained with our proposed method has a better result, 63.44%.

epoch, the model trained with positive memory retention, obtains about 70% accuracy on the validation set, which is comparable with the same model trained with REINFORCE for 100 epochs with an accuracy of 69.92%. After training, we evaluated the best model on the test set. REINFORCE and positive memory retention obtain 69.86% and 70.27% accuracy on the test split, respectively. However, the memory positive attention only needs one-tenth of the training samples. The experiment on MNIST GuessNumber shows that the sample-efficiency has been improved by a factor of 10.

Ablation tests: A summary of the ablation tests is shown in Table 1. We can see that the bounded importance weight proposal is critical for policy gradient with importance sampling. Without this component, the model *diverges* quickly, shown in Table 1 (#3-4). The other proposed innovations all improve model performance as well, as shown in Table 1 (#5-9). One can also see that the choice of the bound parameter ω_{max} has a major influence on the performance.

4.2. GuessWhat?! Game

Experiment settings: In the GuessWhat?! dataset [14] the dialogs are collected using Amazon Mechanical Turk with respect to MS-COCO [23] images. Each game is composed of an image, a target object in the image, the spatial information of the objects, the category of the objects, and the QA-dialogs. Unlike the MNIST GuessNumber, the questions in the training set are in free form text. The answers are still in the yes/no form. This dataset is more challenging due to its large vocabulary size (5 K), and long dialog sequences. Due to the large action space and the extremely delayed reward signals, the importance sampling estimates have very large variances. In our experiment, when we first attempted to retain with all past trajectories, and without the weight bound, the model *diverged* quickly, as in the MNIST GuessNumber experiments, see Table 1 (#3). Our proposed method reduces the variance and makes the sample reuse possible for real-word sceneries.

Model details and pretraining: Each game in the Guess-

What?! contains three roles, a questioner, an answerer, and a guesser. As our aim is to compare the sample-efficiency of our proposed model with other strong baselines, we use the same model structure as was used in [7]. The questioner model is a one layer LSTM with 512 units and conditioned on the VGG16-CNN-FC8 [24] features of the image. The answerer model deploys an LSTM with 512 units to process the question along with spatial and categorical information of the target object. The guesser model uses an LSTM to process the whole dialog and can consider all the spatial and categorical information of the objects in the image. The guesser compares the similarities between the dialog representation and each object representation with a dot product, and then takes the guess. All these three models are pretrained with MLE for 30 epochs for a kickstart policy. We reproduced the paper’s experimental results using Torch7 [22], and obtained 41.41% accuracy on the test split after supervised training.

Reinforced training: With the pretrained models, we keep the answerer and the guesser fixed and train the questioner model. We train the model for 100 epochs, using REINFORCE with a learning rate α of 0.0001 and a running average as the baseline b . Our reimplemention using Torch obtains 62.61% accuracy on the test split, about 2% higher than their result of 60.3% from the original implementation [25], due to some technical differences. We use our reimplemention as the REINFORCE baseline, in Figure 3, to eliminate the influence of these technical differences for a fair comparison. For positive memory retention, Algorithm 1, we use weight bound $\omega_{max} = 5$, so that $\omega \in [1/5, 5]$, to stabilize the training. We use the early stopping threshold in each epoch as $n_{max} = 2$. We observe that with $\omega_{max} = 5$, about 85% of the trajectories in the memory contribute to the model weight updates. This high ratio of reuse is also due to the probability updating mechanism, which bridges the gap between the behavior policies and the target policy.

Results: From Figure 3 right, we can see that the model trained with REINFORCE obtains 63.39% accuracy on the validation split after training for 100 epochs. However, at the

Table 1: Ablation tests on MNIST GuessNumber: Notations: RF denotes the REINFORCE; IS is importance sampling; PM means using positive memory only, otherwise all memory; UB denotes the upper bound ω_{max} ; LB represents the lower bound $1/\omega_{max}$; PB is the probability updating trick; ES means the early stopping within epochs, if use ES, then the early stopping threshold is 2, otherwise train for 3 iterations; (%) is the accuracy on the test split using the best model selected via validation split during 10 epochs of training. The **upper part** above the middle horizontal line shows the ablation test of different components in the positive memory retention. Here, $\omega_{max} = 10$, and (#1) is the performance of the kickstart policy after supervised training. Note that (#3) and (#4) *diverges* quickly, which means that the testing accuracies are lower than 20.0% after 10 epochs of training. UB makes the stable training of RF+IS possible, shown in (#5). PM, LB, PB, ES, contribute 0.76%, 1.18%, 0.73%, and 1.54%, respectively. The **lower part** below the middle horizontal line shows the extensive evaluation regarding to the upper bound parameter ω_{max} .

#	RF	IS	PM	UB	LB	PB	ES	(%)
1	-	-	-	-	-	-	-	63.09
2	✓	-	-	-	-	-	-	65.40
3	✓	✓	-	-	-	-	-	< 20.
4	✓	✓	✓	-	-	-	-	< 20.
5	✓	✓	-	✓	-	-	-	66.06
6	✓	✓	-	✓	✓	-	-	67.24
7	✓	✓	✓	✓	✓	-	-	68.00
8	✓	✓	✓	✓	✓	✓	-	68.73
9	✓	✓	✓	✓	✓	✓	✓	70.27
10	✓	✓	✓	1	✓	✓	✓	66.24
11	✓	✓	✓	5	✓	✓	✓	65.69
12	✓	✓	✓	10	✓	✓	✓	70.27
13	✓	✓	✓	20	✓	✓	✓	69.38
14	✓	✓	✓	30	✓	✓	✓	69.09
15	✓	✓	✓	100	✓	✓	✓	65.39

50th epoch, the same model trained with positive memory retention reaches 63.44% validation accuracy. After training, we evaluated the best model on the test set. REINFORCE and positive memory retention obtain 62.61% and 63.17% accuracy on the test split, respectively. We can see that the proposed method provides state-of-the-art performance with double sample-efficiency on the GuessWhat?! dataset.

5. RELATED WORK

Goal-Oriented Dialogs: Recently, researchers started exploring intensively deep RL for goal-oriented dialogs [6, 7, 8, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40], focusing on learning to achieve a goal via dialog. Bordes and Weston [6] pointed out that the recent success in chit-chat dialogs may not carry over to goal-oriented settings. Strub et al. [7] and Das et al. [8] conduct the task-oriented conversation over image guessing games. In [7], the dialog aims at object

discovery through a series of yes/no QAs. Policy gradient is used to improve performances of dialog agents in terms of task completion rate. Das et al. [8] use policy gradient to train two chat-bots to play image guessing games and show that they establish their own communication style. Both works use on-policy policy gradient methods. Sample-efficient on-off-policy learning methods, as developed in this paper, have not yet been explored in the field of goal-oriented dialog.

Sample-Efficient RL: While the goal-oriented dialog using RL is a recent research direction, control tasks via RL have been studied extensively and importance sampling based actor-critic methods have been known to be beneficial for sample-efficiency [18, 13, 41, 42, 43, 44]. However, the control tasks are inherently different from dialog tasks in the aspect of action space. For example, in Atari games, the agents normally have less than 20 actions to explore; in contrast, the action space, i.e. the vocabulary, contains thousands of words in dialogs. Moreover, the reward of a dialog is only available at the end, which is much more sparse and delayed than in Atari games. In these games, there are intermediate rewards prior to the game ending. The long trajectories in dialog tasks make the often observed problem of exploding importance weights even more extreme. Even if an explosion does not occur, the variance of the importance sampling increases. To overcome these challenges, new solutions must be introduced.

Memory Retention: The use of positive memory retention is inspired by recent neuroscience research, which concludes that the brain prioritizes those high-reward memories, which might be the most important for obtaining future rewards [12]. Tresp et al. [45] argue that the brain’s memory functions might inspire technical solutions requiring memory traces. Biologically-inspired experience replay [46], was used to stabilize the training process in RL and thereby was quite successful. These papers used Q-learning, which is an off-policy method that is able to use the past trajectories directly. However, on-policy policy gradients cannot reuse past samples directly [18, 13]. The main contribution of this paper is that we show that our extensions permit an efficient reuse of past samples in on-policy policy gradients methods. These extensions also work well in dialog settings, which are challenging due to the sparse reward and the large action space.

6. CONCLUSION

We proposed a novel positive memory retention mechanism for improving sample-efficiency in dialog policy learning, using past positive trajectories and low-variance importance sampling estimates. The model reuses past positive samples as behavior policies, samples from a bounded importance weight proposal, and updates the target policy with an importance weight correction. We tested on both synthetic and real-word datasets and illustrated dramatically improved sample-efficiency. We demonstrate that policy gradient can successfully be trained using past trajectories in dialog tasks.

7. REFERENCES

- [1] Ilya Sutskever, Oriol Vinyals, and Quoc V Le, “Sequence to sequence learning with neural networks,” in *Advances in neural information processing systems*, 2014, pp. 3104–3112.
- [2] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al., “Mastering the game of go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
- [3] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [4] Mike Lewis, Denis Yarats, Yann N Dauphin, Devi Parikh, and Dhruv Batra, “Deal or no deal? end-to-end learning for negotiation dialogues,” *arXiv preprint arXiv:1706.05125*, 2017.
- [5] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan, “Show and tell: A neural image caption generator,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3156–3164.
- [6] Antoine Bordes and Jason Weston, “Learning end-to-end goal-oriented dialog,” in *International Conference on Learning Representations (ICLR)*, 2017.
- [7] Florian Strub, Harm de Vries, Jérémie Mary, Bilal Piot, Aaron C. Courville, and Olivier Pietquin, “End-to-end optimization of goal-driven and visually grounded dialogue systems,” in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2017.
- [8] Abhishek Das, Satwik Kottur, José M.F. Moura, Stefan Lee, and Dhruv Batra, “Learning cooperative visual dialog agents with deep reinforcement learning,” in *International Conference on Computer Vision (ICCV)*, 2017.
- [9] Ronald J Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” *Machine learning*, 1992.
- [10] Richard S Sutton and Andrew G Barto, *Reinforcement learning: An introduction (Second Edition)*, MIT press, 2018.
- [11] Prithvijit Chattopadhyay, Deshraj Yadav, Viraj Prabhu, Arjun Chandrasekaran, Abhishek Das, Stefan Lee, Dhruv Batra, and Devi Parikh, “Evaluating visual conversational agents via cooperative human-ai games,” in *Conference on Human Computation and Crowdsourcing (HCOMP)*, 2017.
- [12] Matthias J Gruber, Maureen Ritchey, Shao-Fang Wang, Manoj K Doss, and Charan Ranganath, “Post-learning hippocampal dynamics promote preferential retention of rewarding events,” *Neuron*, 2016.
- [13] Thomas Degris, Martha White, and Richard S Sutton, “Off-policy actor-critic,” in *International Conference on Machine Learning (ICML)*, 2012.
- [14] Harm de Vries, Florian Strub, Sarath Chandar, Olivier Pietquin, Hugo Larochelle, and Aaron C. Courville, “Guesswhat?! visual object discovery through multimodal dialogue,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [15] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur, “Recurrent neural network based language model,” in *Interspeech*, 2010.
- [16] Art B. Owen, *Monte Carlo theory, methods and examples*, 2013.
- [17] Kevin P Murphy, *Machine learning: a probabilistic perspective*, MIT press, 2012.
- [18] Tang Jie and Pieter Abbeel, “On a connection between importance sampling and the likelihood ratio policy gradient,” in *Advances in Neural Information Processing Systems (NIPS)*, 2010.
- [19] Jianhua Lin, “Divergence measures based on the shannon entropy,” *IEEE Transactions on Information theory*, 1991.
- [20] Paul Hongsuck Seo, Andreas Lehrmann, Bohyung Han, and Leonid Sigal, “Visual reference resolution using attention memory for visual dialog,” in *Advances in Neural Information Processing Systems*, 2017.
- [21] Sepp Hochreiter and Jürgen Schmidhuber, “Long short-term memory,” *Neural computation*, 1997.
- [22] R. Collobert, K. Kavukcuoglu, and C. Farabet, “Torch7: A matlab-like environment for machine learning,” in *BigLearn, NIPS Workshop*, 2011.
- [23] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick, “Microsoft coco: Common objects in context,” in *European conference on computer vision*. Springer, 2014, pp. 740–755.
- [24] Karen Simonyan and Andrew Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [25] Florian Strub and Harm de Vries, “Guesswhat?! models,” <https://github.com/GuessWhatGame/guesswhat/>, 2017.

- [26] Jason D Williams and Geoffrey Zweig, “End-to-end lstm-based dialog control optimized with supervised and reinforcement learning,” *arXiv preprint arXiv:1606.01269*, 2016.
- [27] Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng, “End-to-end reinforcement learning of dialogue agents for information access,” *arXiv preprint arXiv:1609.00777*, 2016.
- [28] Zachary C Lipton, Jianfeng Gao, Lihong Li, Xiujun Li, Faisal Ahmed, and Li Deng, “Efficient exploration for dialogue policy learning with bbq networks & replay buffer spiking,” *arXiv preprint arXiv:1608.05081*, 2016.
- [29] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky, “Deep reinforcement learning for dialogue generation,” *arXiv preprint arXiv:1606.01541*, 2016.
- [30] Xuijun Li, Yun-Nung Chen, Lihong Li, and Jianfeng Gao, “End-to-end task-completion neural dialogue systems,” *arXiv preprint arXiv:1703.01008*, 2017.
- [31] Alexander Rudnicky and Wei Xu, “An agenda-based dialog management architecture for spoken language systems,” in *IEEE Automatic Speech Recognition and Understanding Workshop*, 1999, vol. 13.
- [32] Satinder P Singh, Michael J Kearns, Diane J Litman, and Marilyn A Walker, “Reinforcement learning for spoken dialogue systems,” in *Advances in Neural Information Processing Systems*, 2000, pp. 956–962.
- [33] Oliver Lemon, Xingkun Liu, Daniel Shapiro, and Carl Tollander, “Hierarchical reinforcement learning of dialogue policies in a development environment for dialogue systems: Reall-dude,” in *BRANDIAL’06, Proceedings of the 10th Workshop on the Semantics and Pragmatics of Dialogue*, 2006, pp. 185–186.
- [34] Tiancheng Zhao and Maxine Eskenazi, “Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning,” *arXiv preprint arXiv:1606.02560*, 2016.
- [35] Rui Zhao and Volker Tresp, “Improving goal-oriented visual dialog agents via advanced recurrent nets with tempered policy gradient,” *arXiv preprint arXiv:1807.00737*, 2018.
- [36] Rui Zhao and Volker Tresp, “Learning goal-oriented visual dialog via tempered policy gradient,” in *IEEE Spoken Language Technology (SLT) (forthcoming)*, 2018.
- [37] Kavosh Asadi and Jason D Williams, “Sample-efficient deep reinforcement learning for dialog control,” *arXiv preprint arXiv:1612.06000*, 2016.
- [38] Gellért Weisz, Paweł Budzianowski, Pei-Hao Su, and Milica Gašić, “Sample efficient deep reinforcement learning for dialogue systems with large action spaces,” *arXiv preprint arXiv:1802.03753*, 2018.
- [39] Pei-Hao Su, Paweł Budzianowski, Stefan Ultes, Milica Gasic, and Steve Young, “Sample-efficient actor-critic reinforcement learning with supervised data for dialogue management,” *arXiv preprint arXiv:1707.00130*, 2017.
- [40] Lu Chen, Xiang Zhou, Cheng Chang, Runzhe Yang, and Kai Yu, “Agent-aware dropout dqn for safe and efficient on-line dialogue policy learning,” in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 2454–2464.
- [41] Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc Bellemare, “Safe and efficient off-policy reinforcement learning,” in *Advances in Neural Information Processing Systems*, 2016, pp. 1054–1062.
- [42] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas, “Sample efficient actor-critic with experience replay,” in *International Conference on Learning Representations (ICLR)*, 2017.
- [43] Audrunas Gruslys, Mohammad Gheshlaghi Azar, Marc G Bellemare, and Remi Munos, “The reactor: A sample-efficient actor-critic architecture,” *arXiv preprint arXiv:1704.04651*, 2017.
- [44] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al., “Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures,” *arXiv preprint arXiv:1802.01561*, 2018.
- [45] Volker Tresp, Cristóbal Esteban, Yinchong Yang, Stephan Baier, and Denis Krompaß, “Learning with memory embeddings,” in *Advances in neural information processing systems (NIPS Workshop)*, 2015.
- [46] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al., “Human-level control through deep reinforcement learning,” *Nature*, 2015.

Chapter 4

Energy-Based Hindsight Experience Prioritization

Energy-Based Hindsight Experience Prioritization

Rui Zhao and Volker Tresp

Ludwig Maximilian University, Munich, Germany
Siemens AG, Corporate Technology, Munich, Germany
{ruizhao,volker.tresp}@siemens.com

Abstract: In Hindsight Experience Replay (HER), a reinforcement learning agent is trained by treating whatever it has achieved as virtual goals. However, in previous work, the experience was replayed at random, without considering which episode might be the most valuable for learning. In this paper, we develop an energy-based framework for prioritizing hindsight experience in robotic manipulation tasks. Our approach is inspired by the work-energy principle in physics. We define a trajectory energy function as the sum of the transition energy of the target object over the trajectory. We hypothesize that replaying episodes that have high trajectory energy is more effective for reinforcement learning in robotics. To verify our hypothesis, we designed a framework for hindsight experience prioritization based on the trajectory energy of goal states. The trajectory energy function takes the potential, kinetic, and rotational energy into consideration. We evaluate our Energy-Based Prioritization (EBP) approach on four challenging robotic manipulation tasks in simulation. Our empirical results show that our proposed method surpasses state-of-the-art approaches in terms of both performance and sample-efficiency on all four tasks, without increasing computational time. A video showing experimental results is available at <https://youtu.be/jtsF2tTeUGQ>.

Keywords: Prioritized Replay, Hindsight Experience, Energy (Physics)

1 Introduction

Reinforcement learning techniques [1] combined with deep neural networks [2] led to great successes in various domains, such as playing video games [3], challenging the World Go Champion [4], and learning autonomously to accomplish robotic tasks [5, 6, 7, 8, 9].

In robotic tasks, autonomous agents are expected to achieve multiple goals in different scenarios. Standard approaches are based on goal-conditioned policies that allow agents to learn different policies concurrently [10, 11, 12, 13, 14]. Alternatively, the agent exploits what alternative goals it has achieved, learns from these achieved states, and further attempts to achieve the real goal. This kind of goal-conditioned curriculum learning has recently been introduced as hindsight experience replay (HER) [9]. HER lets an agent learn from undesired outcomes and tackles the problem of sparse rewards in Reinforcement Learning (RL).

However, HER samples episodes uniformly from the memory buffer for replay. Subsequently, in the selected episode, the virtual goals are sampled randomly at a future timestep with respect to a randomly chosen state. The replay process does not consider which episodes or states might be the most valuable for learning [15]. It would be more efficient in training to prioritize the more important and valuable episodes. The challenge now is how to judge which episodes are more valuable for learning. We define the trajectory energy as the sum of the transition energy of the object over all timesteps of the trajectory, see more detail in Section 3.2. Our hypothesis is that the trajectory energy is an effective metric for indicating which episode is more difficult to achieve. This is readily true in most robotic manipulation tasks. Imagine a robot arm with an end effector, attempting to pick up an object on the floor and place it on a shelf. The achieved goal state is the position of the object at each timestep. In an unsuccessful scenario, the robot arm attempts to reach the object but fails, leaving the object on the floor. The total energy of the object, including the

potential energy and the kinetic energy, does not change during the episode because of Newton’s laws of motion or, equivalently, the work-energy principle. Therefore, the trajectory energy of the object remains zero. In a somewhat better scenario, the robot arm picks up the object successfully, but accidentally drops the object before it reaches the shelf. In this case, the trajectory energy of the object rises because the robot arm does work on the object. This case can be explained by the work-energy principle in physics. In a successful scenario, the robot arm picks up the object and transfers the object to the goal position on the shelf. Here, the trajectory energy is the highest among all three scenarios; the robot arm does the most work on the object and fulfills the task. Obviously, the successful episode is the most valuable for replay. The first scenario is the least important episode because the object state is barely correlated with the trajectory of the robot arm. In this robotic task example, we can see that the trajectory energy indeed indicates the importance of episodes.

In almost all robotic tasks, goal states can be expressed with physics quantities. In these cases, the energy-based prioritization method is applicable. Based on the position and the orientation of the object, we can calculate the linear and the angular velocity, as well as the kinetic energy and the rotational energy. Based on the height of the object, we can calculate the potential energy. We calculate the energy increases from state to state as the transition energy, see Equation (1). We sum the transition energy over time to have the trajectory energy, see Equation (2). Using the trajectory energy function, in our approach, we prioritize the episodes with higher trajectory energy to speed up the training. In order to verify our hypothesis, we implemented and tested our prioritization framework in four simulated robotic tasks. These tasks include pick-and-place with a robot arm and manipulating a block, an egg, and a pen with a Dexterous robot hand. The tasks are simulated with the MuJoCo physics engine [16] and runs in the OpenAI Gym environment [17, 15].

In this paper we propose to use the trajectory energy function of achieved goal states as a metric to evaluate which episodes are more difficult to achieve. Subsequently, we introduce Energy-Based Prioritization (EBP) with hindsight experience replay. EBP prioritizes the trajectories with higher energy during training to improve sample-efficiency. The proposed technique is applicable to any robotic manipulation task, whenever multi-goal off-policy RL algorithms apply. The core idea of EBP is to prioritize the explored episode, which is relatively difficult to achieve. This can be considered as a form of curriculum learning, which selects difficult yet achievable episodes for replay.

2 Background

In this section, we introduce the preliminaries, such as the used reinforcement learning approaches and the work-energy principle in physics.

2.1 Markov Decision Process

We consider an agent interacting with an environment. We assume the environment is fully observable, including a set of state \mathcal{S} , a set of action \mathcal{A} , a distribution of initial states $p(s_0)$, transition probabilities $p(s_{t+1}|s_t, a_t)$, a reward function $r: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, and also a discount factor $\gamma \in [0, 1]$. These components formulate a Markov decision process represented as a tuple, $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$. A policy π maps a state to an action, $\pi: \mathcal{S} \rightarrow \mathcal{A}$.

At the beginning of each episode, an initial state s_0 is sampled from the distribution $p(s_0)$. Then, at each timestep t , an agent performs an action a_t at the current state s_t , which follows a policy $a_t = \pi(s_t)$. Afterwards, a reward $r_t = r(s_t, a_t)$ is produced by the environment and the next state s_{t+1} is sampled from the distribution $p(\cdot|s_t, a_t)$. The reward might be discounted by a factor γ at each timestep. The goal of the agent is to maximize the accumulated reward, i.e. the return, $R_t = \sum_{i=t}^{\infty} \gamma^{i-t} r_i$, over all episodes, which is equivalent to maximizing the expected return, $\mathbb{E}_{s_0}[R_0|s_0]$.

2.2 Deep Deterministic Policy Gradient

The objective $\mathbb{E}_{s_0}[R_0|s_0]$ can be maximized using temporal difference learning, policy gradients, or the combination of both, i.e. the actor-critic methods [1]. For continuous control tasks, Deep Deterministic Policy Gradient (DDPG) shows promising performance, which is essentially an off-policy actor-critic method [18]. DDPG has an actor network, $\pi: \mathcal{S} \rightarrow \mathcal{A}$, that learns the policy directly. It also has a critic network, $Q: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, that learns the action-value function, i.e. Q-function Q^π . During training, the actor network uses a behavior policy to explore the environment, which is the

target policy plus some noise, $\pi_b = \pi(s) + \mathcal{N}(0, 1)$. The critic is trained using temporal difference learning with the actions produced by the actor: $y_t = r_t + \gamma Q(s_{t+1}, \pi(s_{t+1}))$. The actor is trained using policy gradients by descending on the gradients of the loss function, $\mathcal{L}_a = -\mathbb{E}_s[Q(s, \pi(s))]$, where s is sampled from the replay buffer. For stability reasons, the target y_t for the actor is usually calculated using a separate network, i.e. an averaged version of the previous Q-function networks [3, 18, 19]. The parameters of the actor and critic are updated using backpropagation.

2.3 Hindsight Experience Replay

For multi-goal continuous control tasks, DDPG can be extended with Universal Value Function Approximators (UVFA) [10]. UVFA essentially generalizes the Q-function to multiple goal states $g \in \mathcal{G}$. For the critic network, the Q-value depends not only on the state-action pairs, but also depends on the goals: $Q^\pi(s_t, a_t, g) = \mathbb{E}[R_t | s_t, a_t, g]$.

For robotic tasks, if the goal is challenging and the reward is sparse, then the agent could perform badly for a long time before learning anything. Hindsight Experience Replay (HER) encourages the agent to learn something instead of nothing. During exploration, the agent samples some trajectories conditioned on the real goal g . The main idea of HER is that during replay, the selected transitions are substituted with achieved goals g' instead of the real goals. In this way, the agent could get a sufficient amount of reward signal to begin learning. Andrychowicz et al. [9] show that HER makes training possible in challenging robotic environments. However, the episodes are uniformly sampled in the replay buffer, and subsequently, the virtual goals are sampled from the episodes. More sophisticated replay strategies are requested for improving sample-efficiency [15].

2.4 Work-Energy Principle

Prior to the energy-based hindsight experience prioritization method, we illustrate here the work-energy principle using robotic manipulation examples. In physics, a force is said to do work if, when acting, there is a displacement of the point of application in the direction of the force [20]. For instance, a robot arm picks up an object from the floor, and places it on the shelf. The work done on the object is equal to the weight of the object multiplied by the vertical distance to the floor. As a result, the potential energy of the object becomes higher.

The work-energy principle states that the work done by all forces acting on a particle equals the change in the kinetic energy of the particle [21]. That is, the work W done by a force on an object (simplified as a particle) equals the change in the object’s kinetic energy E_k [22], $W = \Delta E_k = \frac{1}{2}mv_2^2 - \frac{1}{2}mv_1^2$, where v_1 and v_2 are the speeds of the object before and after the work is done, respectively, and m is its mass. As the robot arm is moving the object towards the shelf, the work is being done by the robot on the object. Consequently, the kinetic energy of the object increases.

3 Method

In this section, we formally describe our approach, including the motivation, the derivation of the trajectory energy function, the energy-based prioritization framework, and a comparison with prioritized experience replay [23].

3.1 Motivation

Consider a robotic manipulation task. We observe that in order to complete the tasks, the robot needs to apply force and do work on the object. Typically, the more difficult a task is, the more work from the robot is required. Consequently, the energy of the object is changed by the robot. Thus, our hypothesis is that, in robotic manipulation tasks, the trajectory energy of the object indicates the difficulty level of the tasks.

From the perspective of curriculum learning, we want to assign the right level of curriculum to the agent. The curriculum should not be too difficult to achieve, also not too simple to learn. We use the trajectory energy to evaluate the difficulty level of the curriculum, and then prioritize the difficult but still achievable tasks for the agent. In this way, the agent might learn with higher sample-efficiency. In robotic tasks, training samples are expensive to acquire, making sample-efficiency in learning important.

3.2 Trajectory Energy Function

In this section, we introduce the trajectory energy function formally.

A complete trajectory \mathcal{T} in an episode is represented as a tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$. A trajectory contains a series of continuous states s_t , where t is the timestep $t \in \{0, 1, \dots, T\}$. The interval between each timestep, Δt , corresponds to the sampling frequency in the system, such as $\Delta t = 0.04$ in our experiments. Each state $s_t \in \mathcal{S}$ also includes the state of the achieved goal, meaning the goal state is a subset of the normal state. Here, we overwrite the notation s_t as the achieved goal state, i.e. the state of the object. A trajectory energy function, E_{traj} , only depends on the goal states, s_0, s_1, \dots, s_T , and represents the total energy of a trajectory.

Each state s_t is described as a state vector. In robotic manipulation tasks, we use a state vector $s_t = [x_t, y_t, z_t, a_t, b_t, c_t, d_t]$ to describe the state of the object. In each state s_t , x , y , and z specify the object position in the Cartesian coordinate system; a , b , c , and d of a quaternion, $q = a + bi + cj + dk$, describe the orientation of the object. The total trajectory energy consists of three parts, namely the potential energy, the kinetic energy, and the rotational energy.

Potential Energy: The potential energy of the object E_p at the state s_t is calculated using: $E_p(s_t) = mgz_t$, where m denotes the mass of the object, and $g \approx 9.81 \text{ m/s}^2$ represents the gravity of earth.

Kinetic Energy: To calculate the kinetic energy, we need the velocity of the object. The velocity along the x -axis can be calculated using $v_{x,t} \approx (x_t - x_{t-1})/\Delta t$. Similarly, the velocities along the y -axis and the z -axis are calculated as $v_{y,t} \approx (y_t - y_{t-1})/\Delta t$ and $v_{z,t} \approx (z_t - z_{t-1})/\Delta t$, respectively. The kinetic energy at s_t can be approximated as:

$$E_k(s_t) = \frac{1}{2}mv_{x,t}^2 + \frac{1}{2}mv_{y,t}^2 + \frac{1}{2}mv_{z,t}^2 \approx \frac{m((x_t - x_{t-1})^2 + (y_t - y_{t-1})^2 + (z_t - z_{t-1})^2)}{2\Delta t^2}.$$

Rotational Energy: For the rotational energy function, we have the quaternion in the simulation [15], $q = a + bi + cj + dk$, representing the object orientation. First, we need to convert the quaternion representation to Euler angles, (ϕ, θ, ψ) , where ϕ , θ , and ψ represents the rotations around the x , y , and z -axes, respectively. The Euler angles are obtained from quaternion using [24]:

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \arctan \frac{2(ab+cd)}{1-2(b^2+c^2)} \\ \arcsin(2(ac-db)) \\ \arctan \frac{2(ad+bc)}{1-2(c^2+d^2)} \end{bmatrix} = \begin{bmatrix} \text{atan2}(2(ab+cd), 1-2(b^2+c^2)) \\ \text{asin}(2(ac-db)) \\ \text{atan2}(2(ad+bc), 1-2(c^2+d^2)) \end{bmatrix}.$$

Note that to obtain full orientations we use atan2 in the implementation instead of the regular atan function because atan2 allows calculating the arc-tangent of all four quadrants. Atan only allows calculating of quadrants one and four. The rotational energy in physics is defined as: $E_r = \frac{1}{2}I\omega^2$, where I is the moment of inertia around the axis of rotation; ω is the angular velocity and E_r is the rotational energy, also termed as angular kinetic energy. The angular velocity around the x -axis is: $\omega_{x,t} \approx (\phi_t - \phi_{t-1})/\Delta t$. Similarly, for the y and z -axes $\omega_{y,t} \approx (\theta_t - \theta_{t-1})/\Delta t$ and $\omega_{z,t} \approx (\psi_t - \psi_{t-1})/\Delta t$. We approximate the rotational energy as:

$$E_r(s_t) = \frac{1}{2}I_x\omega_{x,t}^2 + \frac{1}{2}I_y\omega_{y,t}^2 + \frac{1}{2}I_z\omega_{z,t}^2 \approx \frac{I_x(\phi_t - \phi_{t-1})^2 + I_y(\theta_t - \theta_{t-1})^2 + I_z(\psi_t - \psi_{t-1})^2}{2\Delta t^2}.$$

Total Energy: The total energy is defined as: $E(s_t) = E_p(s_t) + E_k(s_t) + E_r(s_t)$. The total energy includes the potential energy, the kinetic energy, and the rotation energy. Since for prioritizing different trajectories, we are only interested in the relative differences of the trajectory energy, these constant variables, including m , I_x , I_y , and I_z , can be set as a constant, such as $m = I_x = I_y = I_z = 1$ used in our experiments.

Transition Energy: We define the transition energy as the total energy increase from the previous state s_{t-1} to the current state s_t , mathematically:

$$E_{tran}(s_{t-1}, s_t) = \text{clip}(E(s_t) - E(s_{t-1}), 0, E_{tran}^{max}) \quad (1)$$

where $t \geq 1$ and E_{tran}^{max} is the predefined maximal transition energy value. The clip function limits the transition energy value in an interval of $[0, E_{tran}^{max}]$. Here, we are only interested in the positive transition energy because the energy increase of the object is only due to the work done by the

robot. The robot does work on the object, consequently, the total energy of the object increases. In practice, to mitigate the influence of some particular large transition energy, we find it useful to clip the transition energy with a threshold value E_{tran}^{max} . This trick makes the training stable. The threshold value can either be tuned as a hyperparameter or estimated using the energy functions.

Trajectory Energy: Given the definition of the transition energy, we define the trajectory energy as the sum of the transition energy over all the timesteps in the trajectory, mathematically:

$$E_{traj}(\mathcal{T}) = E_{traj}(s_0, s_1, \dots, s_T) = \sum_{t=1}^T E_{tran}(s_{t-1}, s_t) \quad (2)$$

3.3 Energy-Based Prioritization

In this section, we describe the Energy-Based Prioritization (EBP) framework. In a nutshell, we first calculate the trajectory energy, then prioritize the trajectories with higher energy for replay.

At the beginning of each episode, the agent uses random policies to start to explore the environment. The sampled trajectories are stored in a replay buffer. When the agent acquires a new trajectory, the agent calculates the energy by using the trajectory energy function, Equation (2), and stores the energy value along with the trajectory in the replay buffer for later prioritization.

During sampling from the replay buffer, the agent uses the trajectory energy values directly as the probability for sampling. This means that the high energy trajectories have higher priorities to be replayed. Mathematically, the probability of a trajectory \mathcal{T}_i to be replayed after the prioritization is:

$$p(\mathcal{T}_i) = \frac{E_{traj}(\mathcal{T}_i)}{\sum_{n=1}^N E_{traj}(\mathcal{T}_n)} \quad (3)$$

where N is the total number of trajectories in the buffer.

Complete Algorithm: We summarize the complete training algorithm in Algorithm 1.

Algorithm 1 HER with Energy-Based Prioritization (EBP)

Given:

- an off-policy RL algorithm \mathbb{A} ▷ e.g. DQN, DDPG
- a reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \rightarrow \mathbb{R}$. ▷ e.g. $r(s, a, g) = -1$ if fail, 0 if success

Initialize neural networks of \mathbb{A} and replay buffer R

for episode = 1, N **do**

Sample a goal g and an initial state s_0 .

for $t = 0, T - 1$ **do**

Sample an action a_t using the behavioral policy from \mathbb{A} :

$$a_t \leftarrow \pi_b(s_t \| g) \quad \triangleright \| \text{ denotes concatenation}$$

Execute the action a_t and observe a new state s_{t+1}

end for

Calculate trajectory energy $E_{traj}(s_0, s_1, \dots, s_T)$ via Equation (1) and (2) ▷ trajectory energy

Calculate priority $p(\mathcal{T})$ based on Equation (3)

for $t = 0, T - 1$ **do**

$$r_t := r(s_t, a_t, g)$$

Store the transition $(s_t \| g, a_t, r_t, s_{t+1} \| g, p, E_{traj})$ in R

Sample trajectory \mathcal{T} for replay based on priority $p(\mathcal{T})$ ▷ prioritization

Sample transitions (s_t, a_t, s_{t+1}) from \mathcal{T}

Sample virtual goals $g' \in \{s_{t+1}, \dots, s_{T-1}\}$ at a future timestep in \mathcal{T}

$$r'_t := r(s_t, a_t, g') \quad \triangleright \text{ recalculate reward (HER)}$$

Store the transition $(s_t \| g', a_t, r'_t, s_{t+1} \| g', p, E_{traj})$ in R

end for

for $t = 1, M$ **do**

Sample a minibatch B from the replay buffer R

Perform one step of optimization using \mathbb{A} and minibatch B

end for

end for

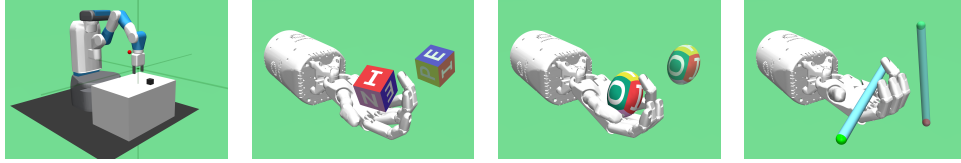


Figure 1: Robot arm Fetch and Shadow Dexterous hand environment: FetchPickAndPlace, HandManipulateBlock, HandManipulateEgg, and HandManipulatePen.

3.4 Comparison with Prioritized Experience Replay

To the best of our knowledge, the most similar method to EBP is Prioritized Experience Replay (PER) [23]. To combine PER with HER, we calculate the TD-error of each transition based on the randomly selected achieved goals. Then we prioritize the transitions with higher TD-errors for replay. It is known that PER can become very expensive in computational time. The reason is that PER uses TD-errors for prioritization. After each update of the model, the agent needs to update the priorities of the transitions in the replay buffer with the new TD-errors, and then ranks them based on the priorities and samples the trajectories for replay. In our experiments, see Section 4, we use the efficient implementation based on the "sum-tree" data structure, which can be relatively efficiently updated and sampled from [23].

Compared to PER, EBP is much faster in computational time because it only calculates the trajectory energy once, when a new trajectory becomes available. Due to this reason, EBP is much more efficient than PER in computational time and can easily be combined with any multi-goal RL methods, such as HER. In the experiments Section 4, we first compare the performance improvement of EBP and PER. Afterwards, we compare the time-complexity of PER and EBP. We show that EBP improves performance without additional computational time. However, PER consumes much more computational time with less improvement. Furthermore, the motivations of PER and EBP are different. The former uses TD-errors, while the latter is based on the energy in physics.

4 Experiments

In this section, we first introduce the robot simulation environment used for evaluation. Then, we investigate the following questions:

- Does incorporating energy-based prioritization bring benefits to hindsight experience replay?
- Does energy-based prioritization improve the sample-efficiency in robotic manipulation tasks?
- How does the trajectory energy relate to the TD-errors of the trajectory during training?

Our code is available at this link¹.

Environments: The environment we used throughout our experiments is the robotic simulations provided by OpenAI Gym [17, 15], using the MuJoCo physics engine [16].

The robotic environment is based on currently existing robotic hardware and is designed as a standard benchmark for Multi-goal RL. The robot agent is required to complete several tasks with different goals in each scenario. There are two kinds of robot agents in the environment. One is a 7-DOF Fetch robotic arm with a two-finger gripper as an end-effector. The other is a 24-DOF Shadow Dexterous robotic hand. We use four challenging tasks for evaluation, including pick & place, and hand manipulation of block, egg, or pen, see Figure 1.

The states in the simulation consist of positions, orientations, linear and angular velocities of all robot joints and of an object. Goals represent desired position and orientations of the object. There is a tolerant range around the desired positions and orientations. In all environments, we consider sparse rewards. If the object is not in the tolerant range of the goal, the agent receives reward signal -1 for each transition; otherwise the reward signal is 0.

Performance: To test the performance difference between vanilla HER, HER with PER, and HER with EBP, we run the experiment in all four challenging object manipulation robotic environments.

¹<https://github.com/ruizhaogit/EnergyBasedPrioritization>

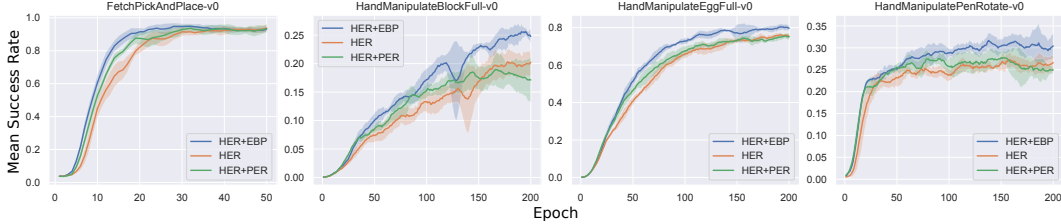


Figure 2: Mean test success rate with standard deviation range for all four environments

Table 1: Final Mean Success Rate for all four environments

	Pick & Place	Block	Egg	Pen
Vanilla HER	93.78%	20.32%	76.19%	27.28%
HER + PER	93.66%	18.95%	75.46%	27.74%
HER + EBP	94.84%	25.63%	80.42%	31.69%

We compare the mean success rates. Each experiment is carried out across 5 random seeds and the shaded area represents the standard deviation. The learning curve with respect to training epochs is shown in Figure 2. For all experiments, we use 19 CPUs. For the robotic arm environment, we train the model for 50 epochs with $E_{tran}^{max} = 0.5$. For the robotic hand environment, we train the agent for 200 epochs with $E_{tran}^{max} = 2.5$. After training, we use the best learned policy as the final policy, and test it in the environment. The testing results are the final mean success rates. A comparison of the final performances is shown in Table 1.

From Figure 2, we can see that HER with EBP converges faster than do both vanilla HER and HER with PER in all four tasks. The agent trained with EBP also shows a better performance, at the end of the training time. This is attractive, since HER with EBP consumes nearly the same computational time as vanilla HER, as shown in Table 2. However, we see that HER with PER consumes about 10 times the training time as vanilla HER or HER with EBP does in the robot hand environments.

From Table 1, we can see that HER cooperating with EBP gives a better performance in all four tasks. The improvement varies from 1 percentage point to 5.3 percentage points compared to HER. The average improvement over the four tasks is 3.75 percentage points. We can see that EBP is a simple yet effective method, *without increasing computational time*, but still, improves current state-of-the-art methods.

Sample-Efficiency: To compare the sample-efficiency of vanilla HER and HER with EBP, we compare the number of training samples needed for a certain mean test success rate. The comparison is shown in Figure 3.

From Figure 3, in the `FetchPickAndPlace-v0` environment, we can see that for the same 93.8% mean test success rate, HER needs 93,100 samples for training, while HER with EBP only needs 48,000 samples. In this case, HER with EBP is nearly twice (1.94) as sample-efficient as vanilla HER. Similarly, in the other three environments, EBP improves sample-efficiency by factors of 1.49, 1.69, and 2.72, respectively. In conclusion, for all four testing environments, EBP is able to improve sample-efficiency by an average factor of two (1.96) over vanilla HER’s sample-efficiency.

Insights: We also investigate the correlation between the trajectory energy and the TD-errors of the trajectory. The Pearson correlation coefficient, i.e. Pearson’s r [25], between the energy and the TD-errors of the trajectory is shown in Figure 4. The value of Pearson’s r is between 1 and -1, where 1 is total positive linear correlation, 0 is no linear correlation, -1 is total negative linear correlation. In Figure 4, we can see that the trajectory energy is correlated with the TD-errors of the trajectory with

Table 2: Training time (hours:minutes:seconds) in all four environments (single run)

	Pick & Place	Block	Egg	Pen
Vanilla HER	01:32:40	08:28:19	07:19:59	07:33:29
HER + PER	03:07:45	80:43:27	79:51:55	81:10:38
HER + EBP	01:29:57	07:28:29	07:28:25	07:35:48

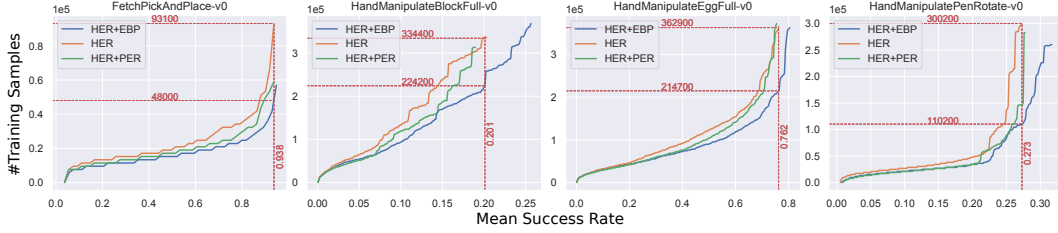


Figure 3: Number of training samples needed with respect to mean test success rate for all four environments (the lower the better)

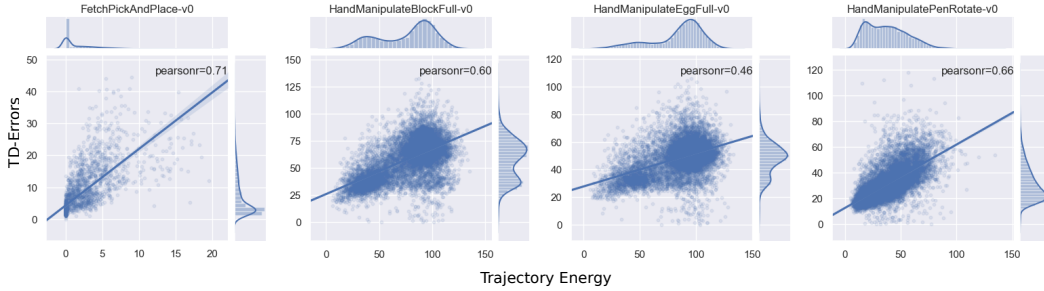


Figure 4: Pearson correlation between the trajectory energy and TD-errors in the middle of training

an average Pearson’s r of 0.6. This proves that high energy trajectories are relatively more valuable for learning. Therefore, it is helpful to prioritize high energy trajectories during training.

5 Related Work

Experience replay was proposed by Lin [26] in 1992 and became popular due to the success of DQN [3] in 2015. In the same year, prioritized experience replay was introduced by Schaul et al. [23] as an improvement of the experience replay in DQN. It prioritized the transitions with higher TD-error in the replay buffer to speed up training. This idea is complementary to our method.

In 2015, Schaul et al. [10] proposed universal function approximators, generalizing not just over states but also over goals. There are also many other research works about multi-task RL [27, 28, 29, 30, 31, 32, 33]. Hindsight experience replay [9] is a kind of goal-conditioned RL that substitutes any achieved goals as real goals to encourage the agent to learn something instead of nothing.

Our method can be considered as a form of curriculum learning [34, 35, 36, 37, 38, 39]. The essence of our method is to assign priority to the achieved trajectories with higher energy, which are relatively difficult to achieve. In RL, curriculum generation can be traced back to 2004. Schmidhuber [40] used a program search to construct an asymptotically optimal algorithm to approach the problem. Recently, Florensa et al. [41] trained the agent reversely, from the start states near the goal states, gradually to the states far from the goals. Our method bares a similar motivation, but is orthogonal to these previous works and can be combined with them.

6 Conclusion

In conclusion, we proposed a simple yet effective energy-based approach to prioritize hindsight experience. Energy-Based Prioritization shows promising experimental results in all four challenging robotic manipulation tasks. This method can be combined with any off-policy RL methods. We integrated physics knowledge about energy into the modern reinforcement learning paradigm, and improved sample-efficiency by a factor of two and the final performance by about four percentage points on top of state-of-the-art methods, without increasing computational time.

References

- [1] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.
- [2] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.
- [3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.
- [4] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.
- [5] A. Y. Ng, A. Coates, M. Diehl, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang. Autonomous inverted helicopter flight via reinforcement learning. In *Experimental Robotics IX*, page 363–372. Springer, 2006.
- [6] J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697, 2008.
- [7] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.
- [8] Y. Chebotar, M. Kalakrishnan, A. Yahya, A. Li, S. Schaal, and S. Levine. Path integral guided policy search. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 3381–3388. IEEE, 2017.
- [9] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, page 5048–5058, 2017.
- [10] T. Schaul, D. Horgan, K. Gregor, and D. Silver. Universal value function approximators. In *International Conference on Machine Learning*, pages 1312–1320, 2015.
- [11] J. Schmidhuber. Powerplay: Training an increasingly general problem solver by continually searching for the simplest still unsolvable problem. *Frontiers in psychology*, 4:313, 2013.
- [12] M. P. Deisenroth, P. Englert, J. Peters, and D. Fox. Multi-task policy search for robotics. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 3876–3881. IEEE, 2014.
- [13] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 3357–3364. IEEE, 2017.
- [14] D. Held, X. Geng, C. Florensa, and P. Abbeel. Automatic goal generation for reinforcement learning agents. *arXiv preprint arXiv:1705.06366*, 2017.
- [15] M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.
- [16] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.
- [17] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.
- [18] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

- [19] B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.
- [20] P. A. Tipler and G. Mosca. *Physics for scientists and engineers*. Macmillan, 2007.
- [21] J. L. Meriam and L. G. Kraige. *Engineering mechanics: dynamics*, volume 2. John Wiley & Sons, 2012.
- [22] H. D. Young, R. A. Freedman, and A. L. Ford. *Sears and Zemansky’s university physics*, volume 1. Pearson education, 2006.
- [23] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.
- [24] J.-L. Blanco. A tutorial on se (3) transformation parameterizations and on-manifold optimization. *University of Malaga, Tech. Rep*, 3, 2010.
- [25] J. Benesty, J. Chen, Y. Huang, and I. Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009.
- [26] L.-J. Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.
- [27] J. Schmidhuber and R. Huber. *Learning to generate focus trajectories for attentive vision*. Institut für Informatik, 1990.
- [28] R. Caruana. Multitask learning. In *Learning to learn*, page 95–133. Springer, 1998.
- [29] B. Da Silva, G. Konidaris, and A. Barto. Learning parameterized skills. *arXiv preprint arXiv:1206.6398*, 2012.
- [30] J. Kober, A. Wilhelm, E. Oztop, and J. Peters. Reinforcement learning to adjust parametrized motor primitives to new situations. *Autonomous Robots*, 33(4):361–379, 2012.
- [31] L. Pinto and A. Gupta. Learning to push by grasping: Using multiple tasks for effective learning. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2161–2168. IEEE, 2017.
- [32] D. Foster and P. Dayan. Structure in the space of value functions. *Machine Learning*, 49(2-3): 325–346, 2002.
- [33] R. S. Sutton, J. Modayil, M. Delp, T. Degris, P. M. Pilarski, A. White, and D. Precup. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 761–768. International Foundation for Autonomous Agents and Multiagent Systems, 2011.
- [34] J. L. Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99, 1993.
- [35] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.
- [36] W. Zaremba and I. Sutskever. Learning to execute. *arXiv preprint arXiv:1410.4615*, 2014.
- [37] A. Graves, M. G. Bellemare, J. Menick, R. Munos, and K. Kavukcuoglu. Automated curriculum learning for neural networks. *arXiv preprint arXiv:1704.03003*, 2017.
- [38] S. Sukhbaatar, Z. Lin, I. Kostrikov, G. Synnaeve, A. Szlam, and R. Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv preprint arXiv:1703.05407*, 2017.
- [39] R. K. Srivastava, B. R. Steunebrink, and J. Schmidhuber. First experiments with powerplay. *Neural Networks*, 41:130–136, 2013.
- [40] J. Schmidhuber. Optimal ordered problem solver. *Machine Learning*, 54(3):211–254, 2004.
- [41] C. Florensa, D. Held, M. Wulfmeier, and P. Abbeel. Reverse curriculum generation for reinforcement learning. *arXiv preprint arXiv:1707.05300*, 2017.

Chapter 5

Curiosity-Driven Experience

Prioritization via Density

Estimation

Curiosity-Driven Experience Prioritization via Density Estimation

Rui Zhao and Volker Tresp
Ludwig Maximilian University, Munich, Germany
Siemens AG, Corporate Technology, Munich, Germany
{ruizhao, volker.tresp}@siemens.com

Abstract

In Reinforcement Learning (RL), an agent explores the environment and collects trajectories into the memory buffer for later learning. However, the collected trajectories can easily be imbalanced with respect to the achieved goal states. The problem of learning from imbalanced data is a well-known problem in supervised learning, but has not yet been thoroughly researched in RL. To address this problem, we propose a novel Curiosity-Driven Prioritization (CDP) framework to encourage the agent to over-sample those trajectories that have rare achieved goal states. The CDP framework mimics the human learning process and focuses more on relatively uncommon events. We evaluate our methods using the robotic environment provided by OpenAI Gym. The environment contains six robot manipulation tasks. In our experiments, we combined CDP with Deep Deterministic Policy Gradient (DDPG) with or without Hindsight Experience Replay (HER). The experimental results show that CDP improves both performance and sample-efficiency of reinforcement learning agents, compared to state-of-the-art methods.

1 Introduction

Reinforcement Learning (RL) [49] combined with Deep Learning (DL) [17, 24, 18, 56] led to great successes in various tasks, such as playing video games [29], challenging the World Go Champion [46], conducting goal-oriented dialogues [5, 54, 55, 52], and learning autonomously to accomplish different robotic tasks [32, 36, 25, 9, 1].

One of the biggest challenges in RL is to make the agent learn sample-efficiently in applications with sparse rewards. Recent RL algorithms, such as Deep Deterministic Policy Gradient (DDPG) [26], enable the agent to learn continuous control, such as manipulation and locomotion. Furthermore, to make the agent learn faster in the sparse reward settings, Andrychowicz et al. [1] introduced Hindsight Experience Replay (HER) that encourages the agent to learn from whatever goal states it has achieved. The combination use of DDPG and HER lets the agent learn to accomplish more complex robot manipulation tasks. However, there is still a huge gap between the learning efficiency of humans and RL agents. In most cases, an RL agent needs millions of samples before it becomes good at the tasks, while humans only need a few samples [29].

One ability of humans is to learn with curiosity. Imagine a boy learning to play basketball and he attempting to shoot the ball into the hoop. After a day of training, he replayed the memory about the moves he practiced. During his recall, he realized that he missed most of his attempts. However, a few made contact with the hoop. These near successful attempts are more interesting to learn from. He will put more focus on learning from these. This kind of curiosity-driven learning might make the learning process more efficient.

Similar curiosity mechanisms could be beneficial for RL agents. We are interested in the RL tasks, in which the goals can be expressed in states. In this case, the agent can analyze the achieved goals and find out which states have been achieved most of the time and which are rare. Based on the analysis, the agent is able to prioritize the trajectories, of which the achieved goal states are novel. For example, the goal states could be the position and the orientation of the target object. We want to encourage the agent to balance the training samples in the memory buffer. The reason is that the policy of the agent could be biased and focuses on a certain group of achieved goal states. This causes the samples to be imbalanced in the memory buffer, which we refer to as memory imbalance.

To overcome the class imbalance issue in supervised learning, such as training deep convolutional neural networks with biased datasets, researchers utilized over-sampling and under-sampling techniques [14, 7, 16]. For instance, the number of one image class is significantly higher than another class. They over-sampled the training images in the smaller class to balance the training set and ultimately to improve the classification accuracy. This idea could be combined with experience replay in RL. We investigate into this research direction and propose a novel curiosity-based prioritization framework for RL agents.

In this paper, we introduce a framework called Curiosity-Driven Prioritization (CDP) which allows the agent to realize a curiosity-driven learning ability similar to humans. This approach can be combined with any off-policy RL algorithm. It is applicable whenever the achieved goals can be described with state vectors. The pivotal idea of CDP is to first estimate the density of each achieved goal and then prioritize the trajectories with lower density to balance the samples that the agent learns from. To evaluate CDP, we combine CDP with DDPG and DDPG+HER and test them in the robot manipulation environments.

2 Background

In this section, we introduce the preliminaries, such as the reinforcement learning approaches and the density estimation algorithm we used in the experiments.

2.1 Markov Decision Process

We consider an agent interacting with an environment. We assume the environment is fully observable, including a set of state \mathcal{S} , a set of action \mathcal{A} , a distribution of initial states $p(s_0)$, transition probabilities $p(s_{t+1}|s_t, a_t)$, a reward function $r: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, and also a discount factor $\gamma \in [0, 1]$. These components formulate a Markov decision process represented as a tuple, $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$. A policy π maps a state to an action, $\pi: \mathcal{S} \rightarrow \mathcal{A}$.

At the beginning of each episode, an initial state s_0 is sampled from the distribution $p(s_0)$. Then, at each timestep t , an agent performs an action a_t at the current state s_t , which follows a policy $a_t = \pi(s_t)$. A reward $r_t = r(s_t, a_t)$ is produced by the environment and the next state s_{t+1} is sampled from the distribution $p(\cdot|s_t, a_t)$. The reward might be discounted by a factor γ at each timestep. The goal of the agent is to maximize the accumulated reward, i.e. the return, $R_t = \sum_{i=t}^{\infty} \gamma^{i-t} r_i$, over all episodes, which is equivalent to maximizing the expected return, $\mathbb{E}_{s_0}[R_0|s_0]$.

2.2 Deep Deterministic Policy Gradient

The objective $\mathbb{E}_{s_0}[R_0|s_0]$ can be maximized using temporal difference learning, policy gradients, or the combination of both, i.e. the actor-critic methods [49]. For continuous control tasks, Deep Deterministic Policy Gradient (DDPG) shows promising performance, which is essentially an off-policy actor-critic method [26]. DDPG has an actor network, $\pi: \mathcal{S} \rightarrow \mathcal{A}$, that learns the policy directly. It also has a critic network, $Q: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, that learns the action-value function, i.e. Q-function Q^π . During training, the actor network uses a behavior policy to explore the environment, which is the target policy plus some noise, $\pi_b = \pi(s) + \mathcal{N}(0, 1)$. The critic is trained using temporal difference learning with the actions produced by the actor: $y_t = r_t + \gamma Q(s_{t+1}, \pi(s_{t+1}))$. The actor is trained using policy gradients by descending on the gradients of the loss function, $\mathcal{L}_a = -\mathbb{E}_s[Q(s, \pi(s))]$, where s is sampled from the replay buffer. For stability reasons, the target y_t for the actor is usually calculated using a separate network, i.e. an averaged version of the previous Q-function networks [29, 26, 39]. The parameters of the actor and critic are updated using backpropagation.

2.3 Hindsight Experience Replay

For multi-goal continuous control tasks, DDPG can be extended with Universal Value Function Approximators (UVFA) [41]. UVFA essentially generalizes the Q-function to multiple goal states $g \in \mathcal{G}$. For the critic network, the Q-value depends not only on the state-action pairs, but also depends on the goals: $Q^\pi(s_t, a_t, g) = \mathbb{E}[R_t | s_t, a_t, g]$.

For robotic tasks, if the goal is challenging and the reward is sparse, then the agent could perform badly for a long time before learning anything. Hindsight Experience Replay (HER) encourages the agent to learn from whatever goal states that it has achieved. During exploration, the agent samples some trajectories conditioned on the real goal g . The main idea of HER is that during replay, the selected transitions are substituted with achieved goals g' instead of the real goals. In this way, the agent could get a sufficient amount of reward signal to begin learning. Andrychowicz et al. [1] show that HER makes training possible in challenging robotic environments. However, the episodes are uniformly sampled in the replay buffer, and subsequently, the virtual goals are sampled from the episodes. More sophisticated replay strategies are requested for improving sample-efficiency [38].

2.4 Gaussian Mixture Model

For estimating the density ρ of the achieved goals in the memory buffer, we use a Gaussian mixture model, which can be trained reasonably fast for RL agents. Gaussian Mixture Model (GMM) [13, 31] is a probabilistic model that assumes all the data points are generated from K Gaussian distributions with unknown parameters, mathematically: $\rho(\mathbf{x}) = \sum_{k=1}^K c_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$. Every Gaussian density $\mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is a component of the GMM and has its own mean $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma}_k$. The parameters c_k are the mixing coefficients. The parameter of GMM is estimated using Expectation-Maximization (EM) algorithm [12]. EM fits the model iteratively on the training data from the agent’s memory buffer.

In our experiments, we use Variational Gaussian Mixture Model (V-GMM) [4], a variation of GMM. V-GMM infers an approximate posterior distribution over the parameters of a GMM. The prior for the weight distribution we used is the Dirichlet distribution. This variational inference version of GMM has a natural tendency to set some mixing coefficients c_k close to zero. This enables the model to choose a suitable number of effective components automatically.

3 Method

In this section, we formally describe our approach, including the motivation, the curiosity-driven prioritization framework, and a comparison with prioritized experience replay [42].

3.1 Motivation

The motivation of incorporating curiosity mechanisms into RL agents is motivated by the human brain. Recent neuroscience research [19] has shown that curiosity can enhance learning. They discovered that when curiosity motivated learning was activated, there was increased activity in the hippocampus, a brain region that is important for human memory. To learn a new skill, such as playing basketball, people practice repeatedly in a trial-and-error fashion. During memory replay, people are more curious about the episodes that are relatively different and focus more on those. This curiosity mechanism has been shown to speed up learning.

Secondly, the inspiration of how to design the curiosity mechanism for RL agents comes from the supervised learning community, in particular the class imbalance dataset problem. Real-world datasets commonly show the particularity to have certain classes to be under-represented compared to other classes. When presented with complex imbalanced datasets, standard learning algorithms, including neural networks, fail to properly represent the distributive characteristics of the data and thus provide unfavorable accuracies across the different classes of the data [20, 16]. One of the effective methods to handle this problem is to over-sample the samples in the under-represented class. Therefore, we prioritize the under-represented trajectories with respect to the achieved goals in the agent’s memory buffer to improve the performance.

3.2 Curiosity-Driven Prioritization

In this section, we formally describe the Curiosity-Driven Prioritization (CDP) framework. In a nutshell, we first estimate the density of each trajectory according to its achieved goal states, then prioritize the trajectories with lower density for replay.

3.2.1 Collecting Experience

At the beginning of each episode, the agent uses partially random policies, such as ϵ -greedy, to start to explore the environment and stores the sampled trajectories into a memory buffer for later replay.

A complete trajectory \mathcal{T} in an episode is represented as a tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$. A trajectory contains a series of continuous states s_t , where t is the timestep $t \in \{0, 1, \dots, T\}$. Each state $s_t \in \mathcal{S}$ also includes the state of the achieved goal, meaning the goal state is a subset of the normal state. Here, we overwrite the notation s_t as the achieved goal state, i.e. the state of the object. The density of a trajectory, ρ , only depends on the goal states, s_0, s_1, \dots, s_T .

Each state s_t is described as a state vector. For example, in robotic manipulation tasks, we use a state vector $s_t = [x_t, y_t, z_t, a_t, b_t, c_t, d_t]$ to describe the state of the object, i.e. the achieved goals. In each state s_t , x , y , and z specify the object position in the Cartesian coordinate system; a , b , c , and d of a quaternion, $q = a + bi + cj + dk$, describe the orientation of the object.

3.2.2 Density Estimation

After the agent collected a number of trajectories, we can fit the density model. The density model we use here is the Variational Gaussian Mixture Model (V-GMM) as introduced in Section 2.4. The V-GMM fits on the data in the memory buffer every epoch and refreshes the density for each trajectory in the buffer. During each epoch, when the new trajectory comes in, the density model predicts the density ρ based on the achieved goals of the trajectory as:

$$\rho = \text{V-GMM}(\mathcal{T}) = \sum_{k=1}^K c_k \mathcal{N}(\mathcal{T} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad (1)$$

where $\mathcal{T} = (s_0 \| s_1 \| \dots \| s_T)$ and each trajectory \mathcal{T} has the same length. The symbol $\|$ denotes concatenation. We normalize the trajectory densities using

$$\rho_i = \frac{\rho_i}{\sum_{n=1}^N \rho_n} \quad (2)$$

where N is the number of trajectories in the memory buffer. Now the density ρ is between zero and one, i.e. $0 \leq \rho \leq 1$. After calculating the trajectory density, the agent stores the density value along with the trajectory in the memory buffer for later prioritization.

3.2.3 Prioritization

During replay, the agent puts more focus on the under-represented achieved states and prioritizes the according trajectories. These under-represented achieved goal states have lower trajectory density. We defined the complementary trajectory density as:

$$\bar{\rho} \propto 1 - \rho. \quad (3)$$

When the agent replays the samples, it first ranks all the trajectories with respect to their complementary density values $\bar{\rho}$, and then uses the ranking number (starting from zero) directly as the probability for sampling. This means that the low-density trajectories have high ranking numbers, and equivalently, have higher priorities to be replayed. Here we use the ranking instead of the density directly. The reason is that the rank-based variant is more robust because it is not affected by outliers nor by density magnitudes. Furthermore, its heavy-tail property also guarantees that samples will be diverse [42]. Mathematically, the probability of a trajectory to be replayed after the prioritization is:

$$p(\mathcal{T}_i) = \frac{\text{rank}(\bar{\rho}(\mathcal{T}_i))}{\sum_{n=1}^N \text{rank}(\bar{\rho}(\mathcal{T}_n))} \quad (4)$$

where N is the total number of trajectories in the buffer, and $\text{rank}(\cdot) \in \{0, 1, \dots, N - 1\}$.

3.2.4 Complete Algorithm

We summarize the complete training algorithm in Algorithm 1.

Algorithm 1 Curiosity-Driven Prioritization (CDP)

Given:

- an off-policy RL algorithm \mathbb{A} ▷ e.g. DDPG, DDPG+HER
- a reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \rightarrow \mathbb{R}$. ▷ e.g. $r(s, a, g) = -1$ (fail), 0 (success)

Initialize neural networks of \mathbb{A} , density model V-GMM, and replay buffer R

for epoch = 1, M **do**

for episode = 1, N **do**

 Sample a goal g and an initial state s_0 .

 Sample a trajectory $\mathcal{T} = (s_t \| g, a_t, r_t, s_{t+1} \| g)_{t=0}^T$ using π_b from \mathbb{A}

 Calculate the densities ρ and $\bar{\rho}$ using Equation (1), (2) and (3) ▷ estimate density

 Calculate the priority $p(\mathcal{T})$ using Equation (4)

 Store transitions $(s_t \| g, a_t, r_t, s_{t+1} \| g, p, \bar{\rho})_{t=0}^T$ in R

 Sample trajectory \mathcal{T} from R based on the priority, $p(\mathcal{T})$ ▷ prioritization

 Sample transitions (s_t, a_t, s_{t+1}) from \mathcal{T}

 Sample virtual goals $g' \in \{s_{t+1}, \dots, s_{T-1}\}$ at a future timestep in \mathcal{T}

$r'_t := r(s_t, a_t, g')$ ▷ recalculate reward (HER)

 Store the transition $(s_t \| g', a_t, r'_t, s_{t+1} \| g', p, \bar{\rho})$ in R

 Perform one step of optimization using \mathbb{A}

end for

 Train the density model using the collected trajectories in R ▷ fit density model

 Update the density in R using the trained model

▷ refresh density

end for

3.3 Comparison with Prioritized Experience Replay

To the best of our knowledge, the most similar method to CDP is Prioritized Experience Replay (PER) [42]. To combine PER with HER, we calculate the TD-error of each transition based on the randomly selected achieved goals. Then we prioritize the transitions with higher TD-errors for replay. It is known that PER can become very expensive in computational time. The reason is that PER uses TD-errors for prioritization. After each update of the model, the agent needs to update the priorities of the transitions in the replay buffer with the new TD-errors. The agent then ranks them based on the priorities and samples the trajectories for replay. In our experiments, see Section 4, we use the efficient implementation based on the "sum-tree" data structure, which can be relatively efficiently updated and sampled from [42].

Compared to PER, CDP is much faster in computational time because it only updates the trajectory density once per epoch. Due to this reason, CDP is much more efficient than PER in computational time and can be easily combined with any multi-goal RL methods, such as DDPG and HER. In the experiments, Section 4, we first compare the performance improvement of CDP and PER. Afterwards, we compare the time-complexity of PER and CDP. We show that CDP improves performance with much less computational time than PER. Furthermore, the motivations of PER and CDP are different. The former uses TD-errors, while the latter is based on the density of the trajectories.

4 Experiments

In this section, we first introduce the robot simulation environment used for evaluation. Then, we investigate the following questions:

- Does incorporating CDP bring benefits to DDPG or DDPG+HER?
- Does CDP improve the sample-efficiency in robotic manipulation tasks?
- How does the density $\bar{\rho}$ relate to the TD-errors of the trajectory during training?

Environments: The environment we used throughout our experiments is the robotic simulations provided by OpenAI Gym [6, 38], using the MuJoCo physics engine [51].

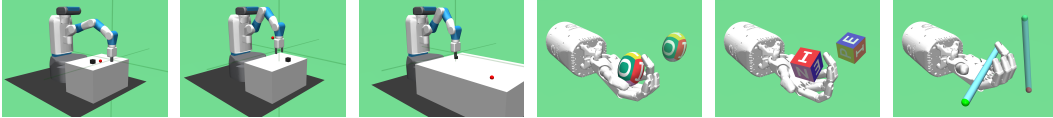


Figure 1: Robot arm Fetch and Shadow Dexterous hand environment: FetchPush, FetchPickAndPlace, FetchSlide, HandManipulateEgg, HandManipulateBlock, and HandManipulatePen.

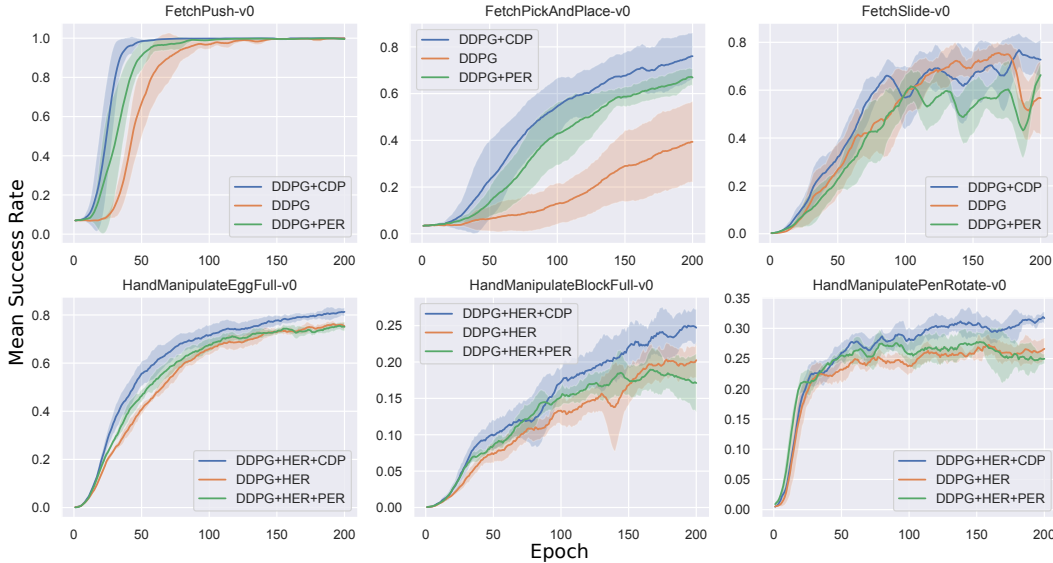


Figure 2: Mean test success rate with standard deviation in all six robot environments

The robotic environment is based on currently existing robotic hardware and is designed as a standard benchmark for Multi-goal RL. The robot agent is required to complete several tasks with different goals in each scenario. There are two kinds of robot agents in the environment. One is a 7-DOF Fetch robotic arm with a two-finger gripper as an end-effector. The other is a 24-DOF Shadow Dexterous robotic hand. We use six challenging tasks for evaluation, including push, slide, pick & place with the robot arm, and hand manipulation of the block, egg, and pen, see Figure 1.

States: The states in the simulation consist of positions, orientations, linear and angular velocities of all robot joints and of an object.

Goals: The real goals are desired positions and orientations of the object.

Rewards: In all environments, we consider sparse rewards. There is a tolerant range between the desired goal states and the achieved goal states. If the object is not in the tolerant range of the real goal, the agent receives a reward signal -1 for each transition; otherwise, the reward signal is 0.

Performance: To test the performance difference among DDPG, DDPG+PER, and DDPG+CDP, we run the experiment in the three robot arm environments. We use the DDPG as the baseline here because the robot arm environment is relatively simple. In the more challenging robot hand environments, we use DDPG+HER as the baseline method and test the performance among DDPG+HER, DDPG+HER+PER, and DDPG+HER+CDP.

We compare the mean success rates. Each experiment is carried out across 5 random seeds and the shaded area represents the standard deviation. The learning curve with respect to training epochs is shown in Figure 2. For all experiments, we use 19 CPUs and train the agent for 200 epochs. After training, we use the best-learned policy as the final policy and test it in the environment. The testing results are the final mean success rates. A comparison of the final performances along with the training time is shown in Table 1.

Table 1: Final mean success rate (%) and the training time (hour) for all six environments

Method	Push		Pick & Place		Slide	
	success	time	success	time	success	time
DDPG	99.90%	5.52h	39.34%	5.61h	75.67%	5.47h
DDPG+PER	99.94%	30.66h	67.19%	25.73h	66.33%	25.85h
DDPG+CDP	99.96%	6.76h	76.02%	6.92h	76.77%	6.66h

Method	Egg		Block		Pen	
	success	time	success	time	success	time
DDPG+HER	76.19%	7.33h	20.32%	8.47h	27.28%	7.55h
DDPG+HER+PER	75.46%	79.86h	18.95%	80.72h	27.74%	81.17h
DDPG+HER+CDP	81.30%	17.00h	25.00%	19.88h	31.88%	25.36h

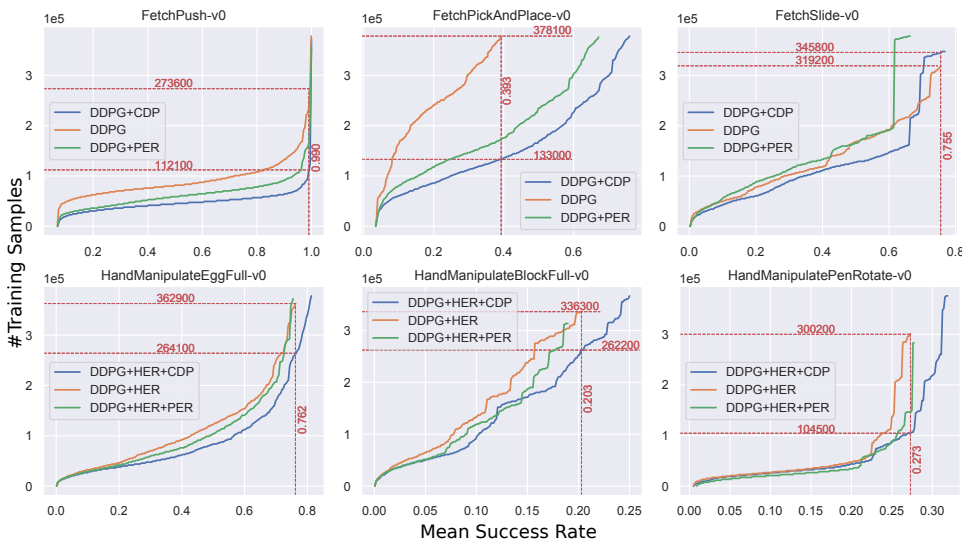


Figure 3: Number of training samples needed with respect to mean test success rate for all six environments (the lower the better)

From Figure 2, we can see that CDP converges faster in all six tasks than both the baseline and PER. The agent trained with CDP also shows a better performance at the end of the training, as shown in Table 1. In Table 1, we can see that the training time of CDP lies in between the baseline and PER. To be more specific, CDP consumes much less computational time than PER does. For example in the robot arm environments, on average DDPG+CDP consumes about 1.2 times the training time of DDPG. In comparison, DDPG+PER consumes about 5 times the training time as DDPG does. In this case, CDP is 4 times faster than PER.

Table 1 shows that baseline methods with CDP give a better performance in all six tasks. The improvement goes up to 39.34 percentage points compared to the baseline methods. The average improvement over the six tasks is 9.15 percentage points. We can see that CDP is a simple yet effective method, improves state-of-the-art methods.

Sample-Efficiency: To compare the sample-efficiency of the baseline and CDP, we compare the number of training samples needed for a certain mean test success rate. The comparison is shown in Figure 3. From Figure 3, in the `FetchPush-v0` environment, we can see that for the same 99% mean test success rate, the baseline DDPG needs 273,600 samples for training, while DDPG+CDP only needs 112,100 samples. In this case, DDPG+CDP is more than twice (2.44) as sample-efficient as DDPG. Similarly, in the other five environments, CDP improves sample-efficiency by factors of 2.84, 0.92, 1.37, 1.28 and 2.87, respectively. In conclusion, for all six environments, CDP is able to improve sample-efficiency by an average factor of two (1.95) over the baseline’s sample-efficiency.

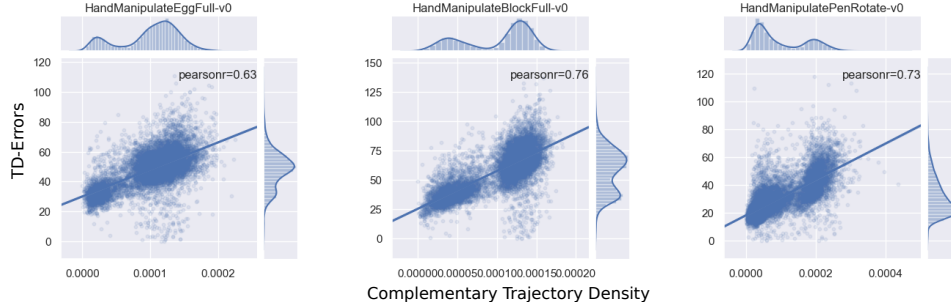


Figure 4: Pearson correlation between the density $\bar{\rho}$ and TD-errors in the middle of training

Insights: We also investigate the correlation between the complementary trajectory density $\bar{\rho}$ and the TD-errors of the trajectory. The Pearson correlation coefficient, i.e. Pearson’s r [3], between the density $\bar{\rho}$ and the TD-errors of the trajectory is shown in Figure 4. The value of Pearson’s r is between 1 and -1, where 1 is total positive linear correlation, 0 is no linear correlation, -1 is total negative linear correlation. In Figure 4, we can see that the complementary trajectory density is correlated with the TD-errors of the trajectory with an average Pearson’s r of 0.7. This proves that the relatively rare trajectories in the memory buffer are more valuable for learning. Therefore, it is helpful to prioritize the trajectories with lower density during training.

5 Related Work

Experience replay was proposed by Lin [27] and became popular due to the success of DQN [29]. In the same year, prioritized experience replay was introduced by Schaul et al. [42] as an improvement of the experience replay in DQN. It prioritized the transitions with higher TD-error in the replay buffer to speed up training. Schaul et al. [41] also proposed universal function approximators, generalizing not just over states but also over goals. There are also many other research works about multi-task RL [45, 8, 11, 23, 37, 15, 50]. Hindsight experience replay [1] is a kind of goal-conditioned RL that substitutes any achieved goals as real goals to encourage the agent to learn something instead of nothing.

Curiosity-driven exploration is a well-studied topic in reinforcement learning [33, 34, 43, 44, 48]. Pathak et al. [35] encourage the agent to explore states with high prediction error. The agents are also encouraged to explore "novel" or uncertain states [2, 28, 40, 22, 30, 10, 47].

However, we integrate curiosity into prioritization and tackle the problem of data imbalance [16] in the memory buffer of RL agents. A recent work [53] introduced a form of re-sampling for RL agents based on trajectory energy functions. The idea of our method is complementary and can be combined. The motivation of our method is from the curiosity mechanism in the human brain [19]. The essence of our method is to assign priority to the achieved trajectories with lower density, which are relatively more valuable to learn from. In supervised learning, similar tricks are used to mitigate the class imbalance challenge, such as over-sampling the data in the under-represented class [21, 20].

6 Conclusion

In conclusion, we proposed a simple yet effective curiosity-driven approach to prioritize agent’s experience based on the trajectory density. Curiosity-Driven Prioritization shows promising experimental results in all six challenging robotic manipulation tasks. This method can be combined with any off-policy RL methods, such as DDPG and DDPG+HER. We integrated the curiosity mechanism via density estimation into the modern RL paradigm and improved sample-efficiency by a factor of two and the final performance by nine percentage points on top of state-of-the-art methods.

References

- [1] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience

- replay. In *Advances in Neural Information Processing Systems*, page 5048–5058, 2017.
- [2] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016.
- [3] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009.
- [4] David M Blei, Michael I Jordan, et al. Variational inference for dirichlet process mixtures. *Bayesian analysis*, 1(1):121–143, 2006.
- [5] Antoine Bordes, Y-Lan Boureau, and Jason Weston. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*, 2016.
- [6] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.
- [7] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259, 2018.
- [8] Rich Caruana. Multitask learning. In *Learning to learn*, page 95–133. Springer, 1998.
- [9] Yevgen Chebotar, Mrinal Kalakrishnan, Ali Yahya, Adrian Li, Stefan Schaal, and Sergey Levine. Path integral guided policy search. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 3381–3388. IEEE, 2017.
- [10] Nuttapon Chentanez, Andrew G Barto, and Satinder P Singh. Intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pages 1281–1288, 2005.
- [11] Bruno Da Silva, George Konidaris, and Andrew Barto. Learning parameterized skills. *arXiv preprint arXiv:1206.6398*, 2012.
- [12] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.
- [13] Richard O Duda and Peter E Hart. Pattern classification and scene analysis. *A Wiley-Interscience Publication, New York: Wiley, 1973, 1973*.
- [14] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multi-scale, deformable part model. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.
- [15] David Foster and Peter Dayan. Structure in the space of value functions. *Machine Learning*, 49(2-3):325–346, 2002.
- [16] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484, 2012.
- [17] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.
- [18] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.
- [19] Matthias J Gruber, Bernard D Gelman, and Charan Ranganath. States of curiosity modulate hippocampus-dependent learning via the dopaminergic circuit. *Neuron*, 84(2):486–496, 2014.
- [20] Haibo He and Eduardo A Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge & Data Engineering*, (9):1263–1284, 2008.

- [21] Geoffrey E Hinton. To recognize shapes, first learn to generate images. *Progress in brain research*, 165:535–547, 2007.
- [22] Rein Houthoofd, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117, 2016.
- [23] Jens Kober, Andreas Wilhelm, Erhan Oztop, and Jan Peters. Reinforcement learning to adjust parametrized motor primitives to new situations. *Autonomous Robots*, 33(4):361–379, 2012.
- [24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [25] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.
- [26] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.
- [27] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.
- [28] Manuel Lopes, Tobias Lang, Marc Toussaint, and Pierre-Yves Oudeyer. Exploration in model-based reinforcement learning by empirically estimating learning progress. In *Advances in Neural Information Processing Systems*, pages 206–214, 2012.
- [29] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.
- [30] Shakir Mohamed and Danilo Jimenez Rezende. Variational information maximisation for intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pages 2125–2133, 2015.
- [31] Kevin P Murphy. *Machine learning: A probabilistic perspective*. adaptive computation and machine learning, 2012.
- [32] Andrew Y Ng, Adam Coates, Mark Diel, Varun Ganapathi, Jamie Schulte, Ben Tse, Eric Berger, and Eric Liang. Autonomous inverted helicopter flight via reinforcement learning. In *Experimental Robotics IX*, page 363–372. Springer, 2006.
- [33] Pierre-Yves Oudeyer and Frederic Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurobotics*, 1:6, 2009.
- [34] Pierre-Yves Oudeyer, Frdric Kaplan, and Verena V Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(2): 265–286, 2007.
- [35] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning (ICML)*, volume 2017, 2017.
- [36] Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697, 2008.
- [37] Lerrel Pinto and Abhinav Gupta. Learning to push by grasping: Using multiple tasks for effective learning. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2161–2168. IEEE, 2017.
- [38] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

- [39] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.
- [40] Pascal Poupart, Nikos Vlassis, Jesse Hoey, and Kevin Regan. An analytic solution to discrete bayesian reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 697–704. ACM, 2006.
- [41] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International Conference on Machine Learning*, pages 1312–1320, 2015.
- [42] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.
- [43] Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pages 222–227, 1991.
- [44] Jürgen Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, 2010.
- [45] Jürgen Schmidhuber and Rudolf Huber. *Learning to generate focus trajectories for attentive vision*. Institut für Informatik, 1990.
- [46] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.
- [47] Bradly C Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015.
- [48] Yi Sun, Faustino Gomez, and Jürgen Schmidhuber. Planning to be surprised: Optimal bayesian exploration in dynamic environments. In *International Conference on Artificial General Intelligence*, pages 41–51. Springer, 2011.
- [49] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.
- [50] Richard S Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M Pilarski, Adam White, and Doina Precup. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 761–768. International Foundation for Autonomous Agents and Multiagent Systems, 2011.
- [51] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.
- [52] Rui Zhao and Volker Tresp. Efficient dialog policy learning via positive memory retention. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 823–830. IEEE, 2018.
- [53] Rui Zhao and Volker Tresp. Energy-based hindsight experience prioritization. *arXiv preprint arXiv:1810.01363*, 2018.
- [54] Rui Zhao and Volker Tresp. Improving goal-oriented visual dialog agents via advanced recurrent nets with tempered policy gradient. *arXiv preprint arXiv:1807.00737*, 2018.
- [55] Rui Zhao and Volker Tresp. Learning goal-oriented visual dialog via tempered policy gradient. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 868–875. IEEE, 2018.
- [56] Rui Zhao, Haider Ali, and Patrick van der Smagt. Two-stream rnn/cnn for action recognition in 3d videos. *arXiv preprint arXiv:1703.09783*, 2017.

Chapter 6

Maximum Entropy-Regularized Multi-Goal Reinforcement Learning

Maximum Entropy-Regularized Multi-Goal Reinforcement Learning

Rui Zhao^{1,2} Xudong Sun¹ Volker Tresp^{1,2}

Abstract

In Multi-Goal Reinforcement Learning, an agent learns to achieve multiple goals with a goal-conditioned policy. During learning, the agent first collects the trajectories into a replay buffer, and later these trajectories are selected randomly for replay. However, the achieved goals in the replay buffer are often biased towards the behavior policies. From a Bayesian perspective, when there is no prior knowledge about the target goal distribution, the agent should learn uniformly from diverse achieved goals. Therefore, we first propose a novel multi-goal RL objective based on weighted entropy. This objective encourages the agent to maximize the expected return, as well as to achieve more diverse goals. Secondly, we developed a maximum entropy-based prioritization framework to optimize the proposed objective. For evaluation of this framework, we combine it with Deep Deterministic Policy Gradient, both with or without Hindsight Experience Replay. On a set of multi-goal robotic tasks of OpenAI Gym, we compare our method with other baselines and show promising improvements in both performance and sample-efficiency.

1. Introduction

Reinforcement Learning (RL) (Sutton & Barto, 1998) combined with Deep Learning (DL) (Goodfellow et al., 2016) has led to great successes in various tasks, such as playing video games (Mnih et al., 2015), challenging the World Go Champion (Silver et al., 2016), and learning autonomously to accomplish different robotic tasks (Ng et al., 2006; Peters & Schaal, 2008; Levine et al., 2016; Chebotar et al., 2017; Andrychowicz et al., 2017).

¹Faculty of Mathematics, Informatics and Statistics, Ludwig Maximilian University of Munich, Munich, Bavaria, Germany
²Siemens AG, Munich, Bavaria, Germany. Correspondence to: Rui Zhao <zhaorui.in.germany@gmail.com>.

One of the biggest challenges in RL is to make the agent learn efficiently in applications with sparse rewards. To tackle this challenge, Lillicrap et al. (2015) developed the Deep Deterministic Policy Gradient (DDPG), which enables the agent to learn continuous control, such as manipulation and locomotion. Schaul et al. (2015a) proposed Universal Value Function Approximators (UVFAs), which generalize not just over states, but also over goals, and extend value functions to multiple goals. Furthermore, to make the agent learn faster in sparse reward settings, Andrychowicz et al. (2017) introduced Hindsight Experience Replay (HER), which encourages the agent to learn from the goal-states it has achieved. The combined use of DDPG and HER allows the agent to learn to accomplish more complex robot manipulation tasks. However, there is still a huge gap between the learning efficiency of humans and RL agents. In most cases, an RL agent needs millions of samples before it is able to solve the tasks, while humans only need a few samples (Mnih et al., 2015).

In previous works, the concept of maximum entropy has been used to encourage exploration during training (Williams & Peng, 1991; Mnih et al., 2015; Wu & Tian, 2016). Recently, Haarnoja et al. (2017) introduced Soft-Q Learning, which learns a deep energy-based policy by evaluating the maximum entropy of actions for each state. Soft-Q Learning encourages the agent to learn all the policies that lead to the optimum (Levine, 2018). Furthermore, Soft Actor-Critic (Haarnoja et al., 2018c) demonstrated a better performance while showing compositional ability and robustness of the maximum entropy policy in locomotion (Haarnoja et al., 2018a) and robot manipulation tasks (Haarnoja et al., 2018b). The agent aims to maximize the expected reward while also maximizing the entropy to succeed at the task while acting as randomly as possible. Based on maximum entropy policies, Eysenbach et al. (2018) showed that the agent is able to develop diverse skills solely by maximizing an information theoretic objective without any reward function. For multi-goal and multi-task learning (Caruana, 1997), the diversity of training sets helps the agent transfer skills to unseen goals and tasks (Pan et al., 2010). The variability of training samples mitigates overfitting and helps the model to better generalize (Goodfellow

This paper is based on our 2018 NeurIPS Deep RL workshop paper (Zhao & Tresp, 2019).

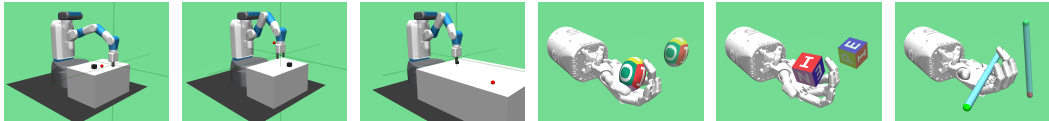


Figure 1. Robot arm Fetch and Shadow Dexterous hand environment: FetchPush, FetchPickAndPlace, FetchSlide, HandManipulateEgg, HandManipulateBlock, and HandManipulatePen.

et al., 2016). In our approach, we combine maximum entropy with multi-goal RL to help the agent to achieve unseen goals by learning uniformly from diverse achieved goals during training.

We observe that during experience replay the uniformly sampled trajectories are biased towards the behavior policies, with respect to the achieved goal-states. Consider training a robot arm to reach a certain point in a space. At the beginning, the agent samples trajectories using a random policy. The sampled trajectories are centered around the initial position of the robot arm. Therefore, the distribution of achieved goals, i.e., positions of the robot arm, is similar to a Gaussian distribution around the initial position, which is non-uniform. Sampling from such a distribution is biased towards the current policies. From a Bayesian point of view (Murphy, 2012), the agent should learn uniformly from these achieved goals, when there is no prior knowledge of the target goal distribution.

To correct this bias, we propose a new objective which combines maximum entropy and the multi-goal RL objective. This new objective uses entropy as a regularizer to encourage the agent to traverse diverse goal-states. Furthermore, we derive a safe lower bound for optimization. To optimize this surrogate objective, we implement maximum entropy-based prioritization as a simple yet effective solution.

2. Preliminary

2.1. Settings

Environments: We consider multi-goal reinforcement learning tasks, like the robotic simulation scenarios provided by OpenAI Gym (Plappert et al., 2018), where six challenging tasks are used for evaluation, including push, slide, pick & place with the robot arm, as well as hand manipulation of the block, egg, and pen, as shown in Figure 1. Accordingly, we define the following terminologies for this specific kind of multi-goal scenarios.

Goals: The goals g are the desired positions and the orientations of the object. Specifically, we use g^e , with e standing for environment, to denote the real goal which serves as the input from the environment, in order to distinguish it from the achieved goal used in Hindsight settings (Andrychowicz et al., 2017). Note that in this paper we consider the case where the goals can be represented by states, which leads

us to the concept of achieved goal-state g^s , with details explained below.

States, Goal-States and Achieved Goals: The state s consists of two sub-vectors, the achieved goal-state s^g , which represents the position and orientation of the object being manipulated, and the context state s^c , i.e. $s = (s^g \| s^c)$, where $\|$ denotes concatenation.

In our case, we define $g^s = s^g$ to represent an achieved goal that has the same dimension as the real goal g^e from the environment. The context state s^c contains the rest information about the state, including the linear and angular velocities of all robot joints and of the object. The real goals g^e can be substituted by the achieved goals g^s to facilitate learning. This goal relabeling technique was proposed by Andrychowicz et al. (2017) as Hindsight Experience Replay.

Achieved Goal Trajectory: A trajectory consisting solely of goal-states is represented as τ^g . We use τ^g to denote all the achieved goals in the trajectory τ , i.e., $\tau^g = (g_0^s, \dots, g_T^s)$.

Rewards: We consider sparse rewards r . There is a tolerated range between the desired goal-states and the achieved goal-states. If the object is not in the tolerated range of the real goal, the agent receives a reward signal -1 for each transition; otherwise, the agent receives a reward signal 0.

Goal-Conditioned Policy: In multi-goal settings, the agent receives the environmental goal g^e and the state input $s = (s^g \| s^c)$. We want to train a goal-conditioned policy to effectively generalize its behavior to different environmental goals g^e .

2.2. Reinforcement Learning

We consider an agent interacting with an environment. We assume the environment is fully observable, including a set of state \mathcal{S} , a set of action \mathcal{A} , a distribution of initial states $p(s_0)$, transition probabilities $p(s_{t+1} | s_t, a_t)$, a reward function $r: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, and a discount factor $\gamma \in [0, 1]$.

Deep Deterministic Policy Gradient: For continuous control tasks, the Deep Deterministic Policy Gradient (DDPG) shows promising performance, which is essentially an off-policy actor-critic method (Lillicrap et al., 2015).

Universal Value Function Approximators: For multi-

goal continuous control tasks, DDPG can be extended by Universal Value Function Approximators (UVFA) (Schaul et al., 2015a). UVFA essentially generalizes the Q-function to multiple goal-states, where the Q-value depends not only on the state-action pairs, but also on the goals.

Hindsight Experience Replay: For robotic tasks, if the goal is challenging and the reward is sparse, the agent could perform badly for a long time before learning anything. Hindsight Experience Replay (HER) encourages the agent to learn from whatever goal-states it has achieved. Andrychowicz et al. (2017) show that HER makes training possible in challenging robotic tasks via goal relabeling, i.e., randomly substituting real goals with achieved goals.

2.3. Weighted Entropy

Guiasu (1971) proposed weighted entropy, which is an extension of Shannon entropy. The definition of weighted entropy is given as

$$\mathcal{H}_p^w = - \sum_{k=1}^K w_k p_k \log p_k, \quad (1)$$

where w_k is the weight of the elementary event and p_k is the probability of the elementary event.

3. Method

In this section, we formally describe our method, including the mathematical derivation of the Maximum Entropy-Regularized Multi-Goal RL objective and the Maximum Entropy-based Prioritization framework.

3.1. Multi-Goal RL

In this paper, we consider multi-goal RL as goal-conditioned policy learning (Schaul et al., 2015a; Andrychowicz et al., 2017; Rauber et al., 2017; Plappert et al., 2018). We denote random variables by upper case letters and the values of random variables by corresponding lower case letters. For example, let $\text{Val}(X)$ denote the set of valid values to a random variable X , and let $p(x)$ denote the probability function of random variable X .

Consider that an agent receives a goal $g^e \in \text{Val}(G^e)$ at the beginning of the episode. The agent interacts with the environment for T timesteps. At each timestep t , the agent observes a state $s_t \in \text{Val}(S_t)$ and performs an action $a_t \in \text{Val}(A_t)$. The agent also receives a reward conditioned on the input goal $r(s_t, g^e) \in \mathbb{R}$.

We use $\tau = s_1, a_1, s_2, a_2, \dots, s_{T-1}, a_{T-1}, s_T$ to denote a trajectory, where $\tau \in \text{Val}(\mathcal{T})$. We assume that the probability $p(\tau | g^e, \theta)$ of trajectory τ , given goal g^e and a policy

parameterized by $\theta \in \text{Val}(\Theta)$, is given as

$$p(\tau | g^e, \theta) = p(s_1) \prod_{t=1}^{T-1} p(a_t | s_t, g^e, \theta) p(s_{t+1} | s_t, a_t).$$

The transition probability $p(s_{t+1} | s_t, a_t)$ states that the probability of a state transition given an action is independent of the goal, and we denote it with $S_{t+1} \perp\!\!\!\perp G^e | S_t, A_t$. For every τ, g^e , and θ , we also assume that $p(\tau | g^e, \theta)$ is non-zero. The expected return of a policy parameterized by θ is given as

$$\begin{aligned} \eta(\theta) &= \mathbb{E} \left[\sum_{t=1}^T r(S_t, G^e) | \theta \right] \\ &= \sum_{g^e} p(g^e) \sum_{\tau} p(\tau | g^e, \theta) \sum_{t=1}^T r(s_t, g^e). \end{aligned} \quad (2)$$

Off-policy RL methods use experience replay (Lin, 1992; Mnih et al., 2015) to leverage bias over variance and potentially improve sample-efficiency. In the off-policy case, the objective, Equation (2), is given as

$$\eta^{\mathcal{R}}(\theta) = \sum_{\tau, g^e} p_{\mathcal{R}}(\tau, g^e | \theta) \sum_{t=1}^T r(s_t, g^e), \quad (3)$$

where \mathcal{R} denotes the replay buffer. Normally, the trajectories τ are randomly sampled from the buffer. However, we observe that the trajectories in the replay buffer are often imbalanced with respect to the achieved goals τ^g . Thus, we propose Maximum Entropy-Regularized Multi-Goal RL to improve performance.

3.2. Maximum Entropy-Regularized Multi-Goal RL

In multi-goal RL, we want to encourage the agent to traverse diverse goal-state trajectories, and at the same time, maximize the expected return. This is like maximizing the empowerment (Mohamed & Rezende, 2015) of an agent attempting to achieve multiple goals. We propose the reward-weighted entropy objective for multi-goal RL, which is given as

$$\begin{aligned} \eta^{\mathcal{H}}(\theta) &= \mathcal{H}_p^w(\mathcal{T}^g) \\ &= \mathbb{E}_p \left[\log \frac{1}{p(\tau^g)} \sum_{t=1}^T r(S_t, G^e) | \theta \right]. \end{aligned} \quad (4)$$

For simplicity, we use $p(\tau^g)$ to represent $\sum_{g^e} p_{\mathcal{R}}(\tau^g, g^e | \theta)$, which is the occurrence probability of the goal-state trajectory τ^g . The expectation is calculated based on $p(\tau^g)$ as well, so the proposed objective is the weighted entropy (Guiasu, 1971; Kelbert et al., 2017) of τ^g , which we denote as $\mathcal{H}_p^w(\mathcal{T}^g)$, where the weight w is the accumulated reward $\sum_{t=1}^T r(s_t, g^e)$ in our case.

The objective function, Equation (4), has two interpretations. The first interpretation is to maximize the weighted expected return, where the rare trajectories have larger weights. Note that when all trajectories occur uniformly, this weighting mechanism has no effect. The second interpretation is to maximize a reward-weighted entropy, where the more rewarded trajectories have higher weights. This objective encourages the agent to learn how to achieve diverse goal-states, as well as to maximize the expected return.

In Equation (4), the weight, $\log(1/p(\tau^g))$, is unbounded, which makes the training of the universal function approximator unstable. Therefore, we propose a safe surrogate objective, $\eta^{\mathcal{L}}$, which is essentially a lower bound of the original objective.

3.3. Surrogate Objective

To construct the safe surrogate objective, we sample the trajectories from the replay buffer with a proposal distribution, $q(\tau^g) = \frac{1}{Z} p(\tau^g) (1 - p(\tau^g))$. $p(\tau^g)$ represents the distribution of the goal trajectories in the replay buffer. The surrogate objective is given in Theorem 1, which is proved to be a lower bound of the original objective, Equation (4).

Theorem 1. *The surrogate $\eta^{\mathcal{L}}(\theta)$ is a lower bound of the objective function $\eta^{\mathcal{H}}(\theta)$, i.e., $\eta^{\mathcal{L}}(\theta) < \eta^{\mathcal{H}}(\theta)$, where*

$$\begin{aligned} \eta^{\mathcal{H}}(\theta) &= \mathcal{H}_p^w(\mathcal{T}^g) \\ &= \mathbb{E}_p \left[\log \frac{1}{p(\tau^g)} \sum_{t=1}^T r(S_t, G^e) \mid \theta \right] \end{aligned} \quad (5)$$

$$\eta^{\mathcal{L}}(\theta) = Z \cdot \mathbb{E}_q \left[\sum_{t=1}^T r(S_t, G^e) \mid \theta \right] \quad (6)$$

$$q(\tau^g) = \frac{1}{Z} p(\tau^g) (1 - p(\tau^g)) \quad (7)$$

Z is the normalization factor for $q(\tau^g)$. $\mathcal{H}_p^w(\mathcal{T}^g)$ is the weighted entropy (Guaşu, 1971; Kelbert et al., 2017), where the weight is the accumulated reward $\sum_{t=1}^T r(S_t, G^e)$, in our case.

Proof. See Appendix. \square

3.4. Prioritized Sampling

To optimize the surrogate objective, Equation (6), we cast the optimization process into a prioritized sampling framework. At each iteration, we first construct the proposal distribution $q(\tau^g)$, which has a higher entropy than $p(\tau^g)$. This ensures that the agent learns from a more diverse goal-state distribution. In Theorem 2, we prove that the entropy

with respect to $q(\tau^g)$ is higher than the entropy with respect to $p(\tau^g)$.

Theorem 2. *Let the probability density function of goals in the replay buffer be*

$$p(\tau^g), \text{ where } p(\tau_i^g) \in (0, 1) \text{ and } \sum_{i=1}^N p(\tau_i^g) = 1. \quad (8)$$

Let the proposal probability density function be defined as

$$q(\tau_i^g) = \frac{1}{Z} p(\tau_i^g) (1 - p(\tau_i^g)), \text{ where } \sum_{i=1}^N q(\tau_i^g) = 1. \quad (9)$$

Then, the proposal goal distribution has an equal or higher entropy

$$\mathcal{H}_q(\mathcal{T}^g) - \mathcal{H}_p(\mathcal{T}^g) \geq 0. \quad (10)$$

Proof. See Appendix. \square

3.5. Estimation of Distribution

To optimize the surrogate objective with prioritized sampling, we need to know the probability distribution of a goal-state trajectory $p(\tau^g)$. We use a Latent Variable Model (LVM) (Murphy, 2012) to model the underlying distribution of $p(\tau^g)$, since LVM is suitable for modeling complex distributions.

Specifically, we use $p(\tau^g \mid z_k)$ to denote the latent-variable-conditioned goal-state trajectory distribution, which we assume to be Gaussians. z_k is the k -th latent variable, where $k \in \{1, \dots, K\}$ and K is the number of the latent variables. The resulting model is a Mixture of Gaussians (MoG), mathematically,

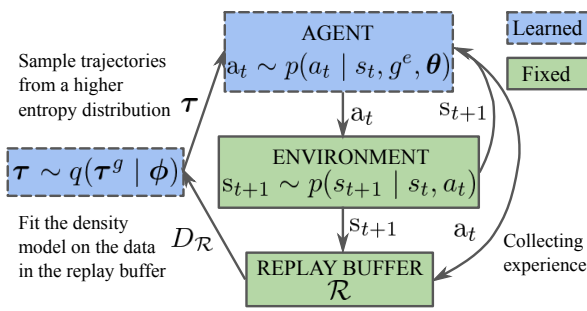
$$p(\tau^g \mid \phi) = \frac{1}{Z} \sum_{i=k}^K c_k \mathcal{N}(\tau^g \mid \mu_k, \Sigma_k), \quad (11)$$

where each Gaussian, $\mathcal{N}(\tau^g \mid \mu_k, \Sigma_k)$, has its own mean μ_k and covariance Σ_k , c_k represents the mixing coefficients, and Z is the partition function. The model parameter ϕ includes all mean μ_i , covariance Σ_i , and mixing coefficients c_k .

In prioritized sampling, we use the complementary predictive density of a goal-state trajectory τ^g as the priority, which is given as

$$\bar{p}(\tau^g \mid \phi) \propto 1 - p(\tau^g \mid \phi). \quad (12)$$

The complementary density describes the likelihood that a goal-state trajectory τ^g occurs in the replay buffer. A high complementary density corresponds to a rare occurrence of the goal trajectory. We want to over-sample these rare goal-state trajectories during replay to increase the entropy


Algorithm 1 Maximum Entropy-based Prioritization (MEP)

```

while not converged do
    Sample goal  $g^e \sim p(g^e)$  and initial state  $s_0 \sim p(s_0)$ 
    for steps_per_epoch do
        for steps_per_episode do
            Sample action  $a_t \sim p(a_t | s_t, g^e, \theta)$  from behavior policy.
            Step environment:  $s_{t+1} \sim p(s_{t+1} | s_t, a_t)$ .
            Update replay buffer  $\mathcal{R}$ .
            Construct prioritized sampling distribution:
             $q(\tau^g) \propto (1 - p(\tau^g | \phi))p(\tau^g)$  with higher  $\mathcal{H}_q(\mathcal{T}^g)$ .
            Sample trajectories  $\tau \sim q(\tau^g | \phi)$ 
            Update policy ( $\theta$ ) to max.  $\mathbb{E}_q [r(S, G)]$  via DDPG, HER.
            Update density model ( $\phi$ ).
    
```

Figure 2. **MEP Algorithm:** We update the density model to construct a higher entropy distribution of achieved goals and update the agent with the more diversified training distribution.

of the training distribution. Therefore, we use the complementary density to construct the proposal distribution as a joint distribution

$$\begin{aligned}
 q(\tau^g) &\propto \bar{p}(\tau^g | \phi)p(\tau^g) \\
 &\propto (1 - p(\tau^g | \phi))p(\tau^g) \\
 &\approx p(\tau^g) - p(\tau^g)^2.
 \end{aligned} \tag{13}$$

3.6. Maximum Entropy-Based Prioritization

With prioritized sampling, the agent learns to maximize the return of a more diverse goal distribution. When the agent replays the samples, it first ranks all the trajectories with respect to their proposal distribution $q(\tau^g)$, and then uses the ranking number directly as the probability for sampling. This means that rare goals have high ranking numbers and, equivalently, have higher priorities to be replayed. Here, we use the ranking instead of the density. The reason is that the rank-based variant is more robust since it is neither affected by outliers nor by density magnitudes. Furthermore, its heavy-tail property also guarantees that samples will be diverse (Schaul et al., 2015b). Mathematically, the probability of a trajectory to be replayed after the prioritization is:

$$q(\tau_i^g) = \frac{\text{rank}(q(\tau_i^g))}{\sum_{n=1}^N \text{rank}(q(\tau_n^g))}, \tag{14}$$

where N is the total number of trajectories in the replay buffer and $\text{rank}(\cdot)$ is the ranking function.

We summarize the complete training algorithm in Algorithm 1 and in Figure 2. In short, we propose Maximum Entropy-Regularized Multi-Goal RL (Section 3.2) to enable RL agents to learn more efficiently in multi-goal tasks (Section 3.1). We integrate a goal entropy term into the normal expected return objective. To maximize the objective, Equation (4), we derive a surrogate objective in Theorem 1, i.e., a lower bound of the original objective. We use prioritized

sampling based on a higher entropy proposal distribution at each iteration and utilize off-policy RL methods to maximize the expected return. This framework is implemented as Maximum Entropy-based Prioritization (MEP).

4. Experiments

We test the proposed method on a variety of simulated robotic tasks, see Section 2.1, and compare it to strong baselines, including DDPG and HER. To the best of our knowledge, the most similar method to MEP is Prioritized Experience Replay (PER) (Schaul et al., 2015b). In the experiments, we first compare the performance improvement of MEP and PER. Afterwards, we compare the time-complexity of the two methods. We show that MEP improves performance with much less computational time than PER. Furthermore, the motivations of PER and MEP are different. The former uses TD-errors, while the latter is based on an entropy-regularized objective function.

In this section, we investigate the following questions:

1. Does incorporating goal entropy via MEP bring benefits to off-policy RL algorithms, such as DDPG or DDPG+HER?
2. Does MEP improve sample-efficiency of state-of-the-art RL approaches in robotic manipulation tasks?
3. How does MEP influence the entropy of the achieved goal distribution during training?

Our code is available online at <https://github.com/ruizhaogit/mep.git>. The implementation uses OpenAI Baselines (Dhariwal et al., 2017) with a backend of TensorFlow (Abadi et al., 2016).

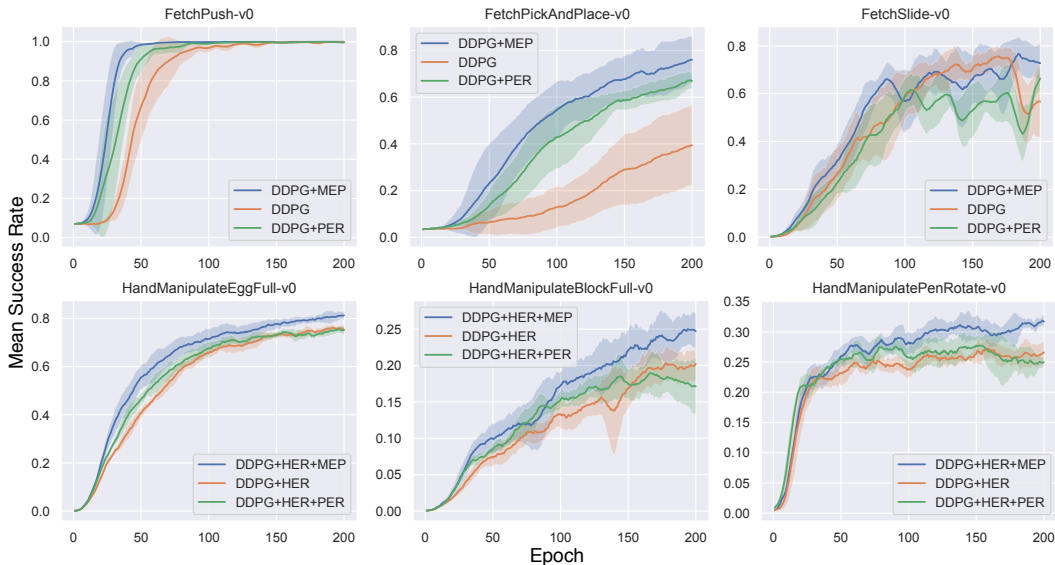


Figure 3. Mean success rate with standard deviation in all six robot environments

Table 1. Mean success rate (%) and training time (hour) for all six environments

Method	Push		Pick & Place		Slide	
	success	time	success	time	success	time
DDPG	99.90%	5.52h	39.34%	5.61h	75.67%	5.47h
DDPG+PER	99.94%	30.66h	67.19%	25.73h	66.33%	25.85h
DDPG+MEP	99.96%	6.76h	76.02%	6.92h	76.77%	6.66h
Method	Egg		Block		Pen	
	success	time	success	time	success	time
DDPG+HER	76.19%	7.33h	20.32%	8.47h	27.28%	7.55h
DDPG+HER+PER	75.46%	79.86h	18.95%	80.72h	27.74%	81.17h
DDPG+HER+MEP	81.30%	17.00h	25.00%	19.88h	31.88%	25.36h

4.1. Performance

To test the performance difference among methods including DDPG, DDPG+PER, and DDPG+MEP, we run the experiment in the three robot arm environments. We use the DDPG as the baseline here because the robot arm environment is relatively simple. In the more challenging robot hand environments, we use DDPG+HER as the baseline method and test the performance among DDPG+HER, DDPG+HER+PER, and DDPG+HER+MEP. To combine PER with HER, we calculate the TD-error of each transition based on the randomly selected achieved goals. Then we prioritize the transitions with higher TD-errors for replay.

Now, we compare the mean success rates. Each experiment is carried out with 5 random seeds and the shaded area represents the standard deviation. The learning curve with respect to training epochs is shown in Figure 3. For all experiments,

we use 19 CPUs and train the agent for 200 epochs. After training, we use the best-learned policy for evaluation and test it in the environment. The testing results are the mean success rates. A comparison of the performances along with the training time is shown in Table 1.

From Figure 3, we can see that MEP converges faster in all six tasks than both the baseline and PER. The agent trained with MEP also shows a better performance at the end of the training, as shown in Table 1. In Table 1, we can also see that the training time of MEP lies in between the baseline and PER. It is known that PER can become very time-consuming (Schaul et al., 2015b), especially when the memory size N is very large. The reason is that PER uses TD-errors for prioritization. After each update of the model, the agent needs to update the priorities of the transitions in the replay buffer, which is $O(\log N)$. In our experiments, we use the efficient implementation based on the “sum-tree” data

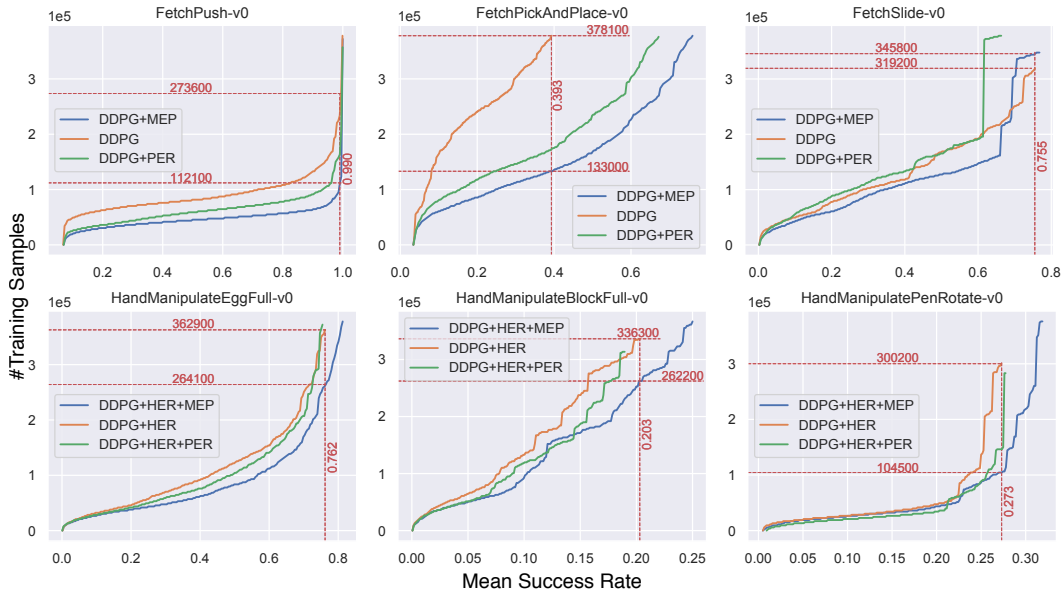


Figure 4. Number of training samples needed with respect to mean success rate for all six environments (the lower the better)

structure, which can be relatively efficiently updated and sampled from (Schaul et al., 2015b). To be more specific, MEP consumes much less computational time than PER. For example in the robot arm environments, on average, DDPG+MEP consumes about 1.2 times the training time of DDPG. In comparison, DDPG+PER consumes about 5 times the training time as DDPG. In this case, MEP is 4 times faster than PER. MEP is faster because it only updates the trajectory density once per epoch and can easily be combined with any multi-goal RL methods, such as DDPG and HER.

Table 1 shows that baseline methods with MEP result in better performance in all six tasks. The improvement increases by up to 39.34 percentage points compared to the baseline methods. The average improvement over the six tasks is 9.15 percentage points. We can see that MEP is a simple, yet effective method and it improves state-of-the-art methods.

4.2. Sample-Efficiency

To compare sample-efficiency of the baseline and MEP, we compare the number of training samples needed for a certain mean success rate. The comparison is shown in Figure 4. From Figure 4, in the *FetchPush-v0* environment, we can see that for the same 99% mean success rate, the baseline DDPG needs 273,600 samples for training, while DDPG+MEP only needs 112,100 samples. In this case, DDPG+MEP is more than twice (2.44) as sample-efficient as DDPG. Similarly, in the other five environments, MEP improves sample-efficiency by factors around one to three. In conclusion, for all six environments, MEP is able to

improve sample-efficiency by an average factor of two (1.95) over the baseline’s sample-efficiency.

4.3. Goal Entropy

To verify that the overall MEP procedure works as expected, we calculated the entropy value of the achieved goal distribution $\mathcal{H}_p(\mathcal{T}^g)$ with respect to the epoch of training. The experimental results are averaged over 5 different random seeds. Figure 5 shows the mean entropy values with its standard deviation in three different environments. From Figure 5, we can see that the implemented MEP algorithm indeed increases the entropy of the goal distribution. This affirms the consistency of the stated theory with the implemented MEP framework.

5. Related Work

Maximum entropy was used in RL by Williams & Peng (1991) as an additional term in the loss function to encourage exploration and avoid local minimums (Mnih et al., 2016; Wu & Tian, 2016; Nachum et al., 2016; Asadi & Littman, 2016). A similar idea has also been utilized in the deep learning community, where entropy loss was used as a regularization technique to penalize over-confident output distributions (Pereyra et al., 2017). In RL, the entropy loss adds more cost to actions that dominate quickly. A higher entropy loss favors more exploration (Mnih et al., 2016). Neu et al. (2017) gave a unified view on entropy-regularized Markov Decision Processes (MDP) and discussed the convergence properties of entropy-regularized RL, including TRPO (Schulman et al., 2015) and A3C (Mnih et al., 2016).

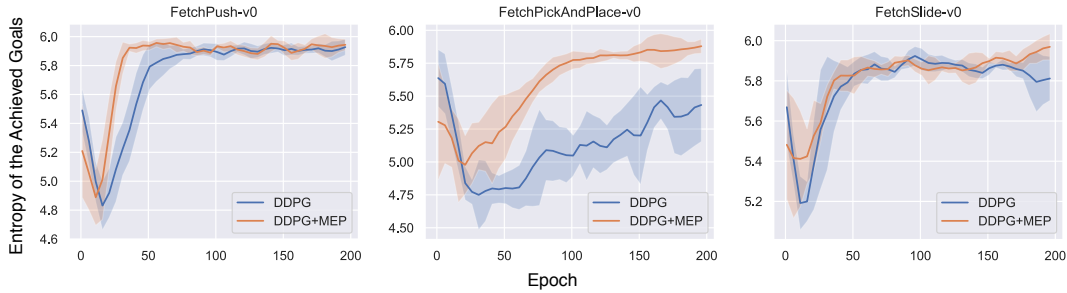


Figure 5. Entropy values of the achieved goal distribution $\mathcal{H}_p(\mathcal{T}^g)$ during training

More recently, Haarnoja et al. (2017) and Levine (2018) proposed deep energy-based policies with state conditioned entropy-based regularization, which is known as Soft-Q Learning. They showed that maximum entropy policies emerge as the solution when optimal control is cast as probabilistic inference. Concurrently, Schulman et al. (2017) showed the connection and the equivalence between Soft-Q Learning and policy gradients. Maximum entropy policies are shown to be robust and lead to better initializations for RL agents (Haarnoja et al., 2018a;b). Based on maximum entropy policies, Eysenbach et al. (2018) developed an information theoretic objective, which enables the agent to automatically discover different sets of skills.

Unlike aforementioned works (Williams & Peng, 1991; Mnih et al., 2016; Haarnoja et al., 2017), the information theoretic objective (Eysenbach et al., 2018) uses state, not actions, to calculate the entropy for distinguishing different skills. Our work is similar to this previous work (Eysenbach et al., 2018) in the sense that we also use the states, instead of actions, to calculate the entropy term and encourage the trained agent to cover a variety of goal-states. Our method generalizes to multi-goal and multi-task RL (Kaelbling, 1993; Sutton et al., 1999; Bakker & Schmidhuber, 2004; Sutton et al., 2011; Szepesvari et al., 2014; Schaul et al., 2015a; Pinto & Gupta, 2017; Plappert et al., 2018).

The entropy term that we used in the multi-goal RL objective is maximized over goal-states. We use maximum goal entropy as a regularization for multi-goal RL, which encourages the agent to learn uniformly with respect to goals instead of experienced transitions. This corrects the bias introduced by the agent’s behavior policies. For example, the more easily achievable goals are generally dominant in the replay buffer. The goal entropy-regularized objective allows the agent to learn to achieve the unknown real goals, as well as various virtual goals.

We implemented the maximum entropy regularization via prioritized sampling based on achieved goal-states. We believe that the most similar framework is prioritized experience replay (Schaul et al., 2015b). Prioritized experience replay was introduced by Schaul et al. (2015b) as an improve-

ment to the experience replay in DQN (Mnih et al., 2015). It prioritizes the transitions with higher TD-error in the replay buffer to speed up training. The prioritized experience replay is motivated by TD-errors. However, the motivation of our method comes from information theory—maximum entropy. Compared to prioritized experience replay, our method performs superior empirically and consumes much less computational time.

The intuition behind our method is to assign priority to those under-represented goals, which are relatively more valuable to learn from (see Appendix). Essentially, our method samples goals from an entropy-regularized distribution, rather than from a true replay buffer distribution, which is biased towards the behavior policies. Similar to recent work on goal sampling methods (Forestier et al., 2017; Péré et al., 2018; Florensa et al., 2018; Zhao & Tresp, 2018; Nair et al., 2018; Warde-Farley et al., 2018), our aim is to model a goal-conditioned MDP. In the future, we want to further explore the role of goal entropy in multi-goal RL.

6. Conclusion

This paper makes three contributions. First, we propose the idea of Maximum Entropy-Regularized Multi-Goal RL, which is essentially a reward-weighted entropy objective. Secondly, we derive a safe surrogate objective, i.e., a lower bound of the original objective, to achieve stable optimization. Thirdly, we implement a novel Maximum Entropy-based Prioritization framework for optimizing the surrogate objective. Overall, our approach encourages the agent to achieve a diverse set of goals while maximizing the expected return.

We evaluated our approach in multi-goal robotic simulations. The experimental results showed that our approach improves performance and sample-efficiency of the agent while keeping computational time under control. More precisely, the results showed that our method improves performance by 9 percentage points and sample-efficiency by a factor of two compared to state-of-the-art methods.

References

- Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pp. 265–283, 2016.
- Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, O. P., and Zaremba, W. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pp. 5048–5058, 2017.
- Asadi, K. and Littman, M. L. An alternative softmax operator for reinforcement learning. *arXiv preprint arXiv:1612.05628*, 2016.
- Bakker, B. and Schmidhuber, J. Hierarchical reinforcement learning based on subgoal discovery and subpolicy specialization. In *Proc. of the 8-th Conf. on Intelligent Autonomous Systems*, pp. 438–445, 2004.
- Caruana, R. Multitask learning. *Machine learning*, 28(1): 41–75, 1997.
- Chebotar, Y., Kalakrishnan, M., Yahya, A., Li, A., Schaal, S., and Levine, S. Path integral guided policy search. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pp. 3381–3388. IEEE, 2017.
- Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., Wu, Y., and Zhokhov, P. Openai baselines. <https://github.com/openai/baselines>, 2017.
- Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.
- Florensa, C., Held, D., Geng, X., and Abbeel, P. Automatic goal generation for reinforcement learning agents. In *International Conference on Machine Learning*, pp. 1514–1523, 2018.
- Forestier, S., Mollard, Y., and Oudeyer, P.-Y. Intrinsically motivated goal exploration processes with automatic curriculum learning. *arXiv preprint arXiv:1708.02190*, 2017.
- Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. *Deep learning*, volume 1. MIT press Cambridge, 2016.
- Guiaşu, S. Weighted entropy. *Reports on Mathematical Physics*, 2(3):165–179, 1971.
- Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement learning with deep energy-based policies. *arXiv preprint arXiv:1702.08165*, 2017.
- Haarnoja, T., Hartikainen, K., Abbeel, P., and Levine, S. Latent space policies for hierarchical reinforcement learning. *arXiv preprint arXiv:1804.02808*, 2018a.
- Haarnoja, T., Pong, V., Zhou, A., Dalal, M., Abbeel, P., and Levine, S. Composable deep reinforcement learning for robotic manipulation. *arXiv preprint arXiv:1803.06773*, 2018b.
- Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018c.
- Kaelbling, L. P. Hierarchical learning in stochastic domains: Preliminary results. In *Proceedings of the tenth international conference on machine learning*, volume 951, pp. 167–173, 1993.
- Kelbert, M., Stuhl, I., and Suhov, Y. Weighted entropy: basic inequalities. *Modern Stochastics: Theory and Applications*, 4(3):233–252, 2017. doi: 10.15559/17-VMSTA85. URL www.i-journals.org/vmsta.
- Levine, S. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.
- Levine, S., Finn, C., Darrell, T., and Abbeel, P. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.
- Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.
- Lin, L.-J. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Hiedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529, 2015.
- Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937, 2016.
- Mohamed, S. and Rezende, D. J. Variational information maximisation for intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pp. 2125–2133, 2015.

- Murphy, K. P. Machine learning: A probabilistic perspective. adaptive computation and machine learning, 2012.
- Nachum, O., Norouzi, M., and Schuurmans, D. Improving policy gradient by exploring under-appreciated rewards. *arXiv preprint arXiv:1611.09321*, 2016.
- Nair, A. V., Pong, V., Dalal, M., Bahl, S., Lin, S., and Levine, S. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*, pp. 9191–9200, 2018.
- Neu, G., Jonsson, A., and Gómez, V. A unified view of entropy-regularized markov decision processes. *arXiv preprint arXiv:1705.07798*, 2017.
- Ng, A. Y., Coates, A., Diel, M., Ganapathi, V., Schulte, J., Tse, B., Berger, E., and Liang, E. Autonomous inverted helicopter flight via reinforcement learning. In *Experimental Robotics IX*, pp. 363–372. Springer, 2006.
- Pan, S. J., Yang, Q., et al. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.
- Péré, A., Forestier, S., Sigaud, O., and Oudeyer, P.-Y. Un-supervised learning of goal spaces for intrinsically motivated goal exploration. *arXiv preprint arXiv:1803.00781*, 2018.
- Pereyra, G., Tucker, G., Chorowski, J., Kaiser, Ł., and Hinton, G. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.
- Peters, J. and Schaal, S. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4): 682–697, 2008.
- Pinto, L. and Gupta, A. Learning to push by grasping: Using multiple tasks for effective learning. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pp. 2161–2168. IEEE, 2017.
- Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P., et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.
- Rauber, P., Mutz, F., and Schmidhuber, J. Hindsight policy gradients. *arXiv preprint arXiv:1711.06006*, 2017.
- Schaul, T., Horgan, D., Gregor, K., and Silver, D. Universal value function approximators. In *International Conference on Machine Learning*, pp. 1312–1320, 2015a.
- Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015b.
- Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897, 2015.
- Schulman, J., Chen, X., and Abbeel, P. Equivalence between policy gradients and soft q-learning. *arXiv preprint arXiv:1704.06440*, 2017.
- 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. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.
- Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.
- Sutton, R. S., Precup, D., and Singh, S. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2): 181–211, 1999.
- Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., White, A., and Precup, D. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pp. 761–768. International Foundation for Autonomous Agents and Multiagent Systems, 2011.
- Szepesvari, C., Sutton, R. S., Modayil, J., Bhatnagar, S., et al. Universal option models. In *Advances in Neural Information Processing Systems*, pp. 990–998, 2014.
- Warde-Farley, D., Van de Wiele, T., Kulkarni, T., Ionescu, C., Hansen, S., and Mnih, V. Unsupervised control through non-parametric discriminative rewards. *arXiv preprint arXiv:1811.11359*, 2018.
- Williams, R. J. and Peng, J. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.
- Wu, Y. and Tian, Y. Training agent for first-person shooter game with actor-critic curriculum learning. 2016.
- Zhao, R. and Tresp, V. Energy-based hindsight experience prioritization. In *Proceedings of the 2nd Conference on Robot Learning*, pp. 113–122, 2018.
- Zhao, R. and Tresp, V. Curiosity-driven experience prioritization via density estimation. *arXiv preprint arXiv:1902.08039*, 2019.

Maximum Entropy-Regularized Multi-Goal Reinforcement Learning (Appendix)

Rui Zhao^{1,2} Xudong Sun¹ Volker Tresp^{1,2}

A. Proof of Theorem 1

Theorem 1. The surrogate $\eta^{\mathcal{L}}(\boldsymbol{\theta})$ is a lower bound of the objective function $\eta^{\mathcal{H}}(\boldsymbol{\theta})$, i.e., $\eta^{\mathcal{L}}(\boldsymbol{\theta}) < \eta^{\mathcal{H}}(\boldsymbol{\theta})$, where

$$\eta^{\mathcal{H}}(\boldsymbol{\theta}) = \mathcal{H}_p^w(\mathcal{T}^g) = \mathbb{E}_p \left[\log \frac{1}{p(\boldsymbol{\tau}^g)} \sum_{t=1}^T r(S_t, G^e) \mid \boldsymbol{\theta} \right] \quad (1)$$

$$\eta^{\mathcal{L}}(\boldsymbol{\theta}) = Z \cdot \mathbb{E}_q \left[\sum_{t=1}^T r(S_t, G^e) \mid \boldsymbol{\theta} \right] \quad (2)$$

$$q(\boldsymbol{\tau}^g) = \frac{1}{Z} p(\boldsymbol{\tau}^g) (1 - p(\boldsymbol{\tau}^g)) \quad (3)$$

Z is the normalization factor for $q(\boldsymbol{\tau}^g)$. $\mathcal{H}_p^w(\mathcal{T}^g)$ is the weighted entropy (Guiaşu, 1971; Kelbert et al., 2017), where the weight is the accumulated reward $\sum_{t=1}^T r(S_t, G^e)$ in our case.

Proof.

$$\eta^{\mathcal{L}}(\boldsymbol{\theta}) = Z \cdot \mathbb{E}_q \left[\sum_{t=1}^T r(S_t, G^e) \mid \boldsymbol{\theta} \right] \quad (4)$$

$$= \sum_{\boldsymbol{\tau}^g} Z \cdot q(\boldsymbol{\tau}^g) \sum_{t=1}^T r(s_t, g^e) \quad (5)$$

$$= \sum_{\boldsymbol{\tau}^g} \frac{Z}{Z} p(\boldsymbol{\tau}^g) (1 - p(\boldsymbol{\tau}^g)) \sum_{t=1}^T r(s_t, g^e) \quad (6)$$

$$< \sum_{\boldsymbol{\tau}^g} -p(\boldsymbol{\tau}^g) \log p(\boldsymbol{\tau}^g) \sum_{t=1}^T r(s_t, g^e) \quad (7)$$

$$= \mathbb{E}_p \left[\log \frac{1}{p(\boldsymbol{\tau}^g)} \sum_{t=1}^T r(S_t, G^e) \mid \boldsymbol{\theta} \right] \quad (8)$$

$$= \mathcal{H}_p^w(\mathcal{T}^g) \quad (9)$$

$$= \eta^{\mathcal{H}}(\boldsymbol{\theta}) \quad (10)$$

In the inequality, we use the property $\log x < x - 1$. □

¹Faculty of Mathematics, Informatics and Statistics, Ludwig Maximilian University of Munich, Munich, Bavaria, Germany ²Siemens AG, Munich, Bavaria, Germany. Correspondence to: Rui Zhao <zhaorui.in.germany@gmail.com>.

B. Proof of Theorem 2

Theorem 2. *Let the probability density function of goals in the replay buffer be*

$$p(\tau^g), \text{ where } p(\tau_i^g) \in (0, 1) \text{ and } \sum_{i=1}^N p(\tau_i^g) = 1. \quad (11)$$

Let the proposal probability density function be defined as

$$q(\tau_i^g) = \frac{1}{Z} p(\tau_i^g) (1 - p(\tau_i^g)), \text{ where } \sum_{i=1}^N q(\tau_i^g) = 1. \quad (12)$$

Then, the proposal goal distribution has an equal or higher entropy

$$\mathcal{H}_q(\mathcal{T}^g) - \mathcal{H}_p(\mathcal{T}^g) \geq 0. \quad (13)$$

Proof. For clarity, we define the notations in this proof as $p_i = p(\tau_i^g)$ and $q_i = q(\tau_i^g)$.

Note that the definition of Entropy is

$$\mathcal{H}_p = \sum_i -p_i \log(p_i), \quad (14)$$

where the i th summand is $p_i \log(p_i)$, which is a concave function. Since the goal distribution has a finite support I , we have the real-valued vector (p_1, \dots, p_N) and $(\frac{1}{Z}q_1, \dots, \frac{1}{Z}q_N)$.

We use Karamata's inequality (Kadelburg et al., 2005), which states that if the vector (p_1, \dots, p_N) majorizes $(\frac{1}{Z}q_1, \dots, \frac{1}{Z}q_N)$ then the summation of the concave transformation of the first vector is smaller than the concave transformation of the second vector.

In our case, the concave transformation is the weighted information at the i th position $-p_i \log(p_i)$, where the weight is the probability p_i (entropy is the expectation of information). Therefore, the proof of the theorem is also a proof of the majorizing property of p over q (Petrov).

We denote the proposal goal distribution as

$$q_i = f(p_i) = \frac{1}{Z} p_i (1 - p_i). \quad (15)$$

Note that in our case, the partition function Z is a constant.

Majorizing has three requirements (Marshall et al., 1979).

The first requirement is that both vectors must sum up to one. This requirement is already met because

$$\sum_i p_i = \sum_i q_i = 1. \quad (16)$$

The second requirement is that monotonicity exists. Without loss of generality, we assume the probabilities are sorted:

$$p_1 \geq p_2 \geq \dots \geq p_N \quad (17)$$

Thus, if $i > j$ then

$$f(p_i) - f(p_j) = \frac{1}{Z} p_i (1 - p_i) - \frac{1}{Z} p_j (1 - p_j) \quad (18)$$

$$= \frac{1}{Z} [(p_i - p_j) - (p_i + p_j)(p_i - p_j)] \quad (19)$$

$$= \frac{1}{Z} (p_i - p_j)(1 - p_i - p_j) \quad (20)$$

$$\geq 0. \quad (21)$$

which means that if the original goal probabilities are sorted, the transformed goal probabilities are also sorted,

$$f(p_1) \geq f(p_2) \geq \dots \geq f(p_N). \quad (22)$$

The third requirement is that for an arbitrary cutoff index k , there is

$$p_1 + \dots + p_k < q_1 + \dots + q_k. \quad (23)$$

To prove this, we have

$$p_1 + \dots + p_k = \frac{p_1 + \dots + p_k}{1} \quad (24)$$

$$= \frac{p_1 + \dots + p_k}{p_1 + \dots + p_N} \quad (25)$$

$$\geq f(p_1) + \dots + f(p_k) \quad (26)$$

$$= \frac{1}{Z} [p_1(1 - p_1) + \dots + p_k(1 - p_k)] \quad (27)$$

$$= \frac{1}{Z} [p_1 + \dots + p_k - (p_1^2 + \dots + p_k^2)] \quad (28)$$

Note that, we multiply $Z * 1$ to each side of

$$Z = p_1(1 - p_1) + \dots + p_N(1 - p_N). \quad (29)$$

Then we have

$$(p_1 + \dots + p_k)Z * 1 \geq p_1 + \dots + p_k - (p_1^2 + \dots + p_k^2) * 1. \quad (30)$$

Now, we substitute the expression of Z and then have

$$(p_1 + \dots + p_k)[p_1(1 - p_1) + \dots + p_N(1 - p_N)] \geq [p_1 + \dots + p_k - (p_1^2 + \dots + p_k^2)] * 1. \quad (31)$$

We express 1 as a series of terms $\sum_i p_i$, we have

$$(p_1 + \dots + p_k)[p_1(1 - p_1) + \dots + p_N(1 - p_N)] \geq [p_1 + \dots + p_k - (p_1^2 + \dots + p_k^2)] * [(p_1 + \dots + p_k) + (p_{k+1} + \dots + p_N)]. \quad (32)$$

We use the distributive law to the right side and have

$$\begin{aligned} & (p_1 + \dots + p_k)[p_1(1 - p_1) + \dots + p_N(1 - p_N)] \\ & \geq [p_1 + \dots + p_k] * [(p_1 + \dots + p_k) + (p_{k+1} + \dots + p_N)] - [(p_1^2 + \dots + p_k^2)] * [(p_1 + \dots + p_k) + (p_{k+1} + \dots + p_N)]. \end{aligned} \quad (33)$$

We move the first term on the right side to the left and use the distributive law then have

$$(p_1 + \dots + p_k)[-1 * (p_1^2 + \dots + p_N^2)] \geq -[(p_1^2 + \dots + p_k^2)] * [(p_1 + \dots + p_k) + (p_{k+1} + \dots + p_N)]. \quad (34)$$

We use the distributive law again on the right side and move the first term to the left and use the distributive law then have

$$(p_1 + \dots + p_k)[-1 * (p_{k+1}^2 + \dots + p_N^2)] \geq -[(p_1^2 + \dots + p_k^2)] * [(p_{k+1} + \dots + p_N)]. \quad (35)$$

We remove the minus sign then have

$$(p_1 + \dots + p_k)[(p_{k+1}^2 + \dots + p_N^2)] \leq [(p_1^2 + \dots + p_k^2)] * [(p_{k+1} + \dots + p_N)]. \quad (36)$$

To prove the inequality above, it suffices to show that the inequality holds true for each associated term of the multiplication on each side of the inequality.

Suppose that

$$i \leq k < j \quad (37)$$

then we have

$$p_i > p_j. \quad (38)$$

As mentioned above, the probabilities are sorted in descending order. We have

$$p_i p_j^2 - p_i^2 p_j = p_i p_j (p_j - p_i) < 0 \quad (39)$$

then

$$p_i p_j^2 < p_i^2 p_j. \quad (40)$$

Therefore, we have proved that the inequality holds true for an arbitrary associated term, which also applies when they are added up. \square

C. Insights

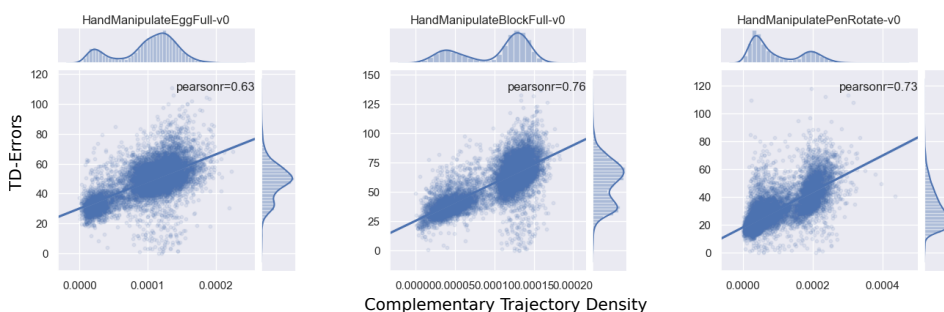


Figure 1. Pearson correlation between the complementary density $\bar{p}(\tau^g)$ and TD-errors in the middle of training

To further understand why maximum entropy in goal space facilitates learning, we look into the TD-errors during training. We investigate the correlation between the complementary predictive density $\bar{p}(\tau^g | \phi)$ and the TD-errors of the trajectory. The Pearson correlation coefficients, i.e., Pearson’s r (Benesty et al., 2009), between the density $\bar{p}(\tau^g | \phi)$ and the TD-errors of the trajectory are 0.63, 0.76, and 0.73, for the hand manipulation of egg, block, and pen tasks, respectively. The plot of the Pearson correlation is shown in Figure 1. The value of Pearson’s r is between 1 and -1, where 1 is total positive linear correlation, 0 is no linear correlation, and -1 is total negative linear correlation. We can see that the complementary predictive density is correlated with the TD-errors of the trajectory with an average Pearson’s r of 0.7. This proves that the agent learns faster from a more diverse goal distribution. Under-represented goals often have higher TD-errors, and thus are relatively more valuable to learn from. Therefore, it is helpful to maximize the goal entropy and prioritize the under-represented goals during training.

References

- Benesty, J., Chen, J., Huang, Y., and Cohen, I. Pearson correlation coefficient. In *Noise reduction in speech processing*, pp. 1–4. Springer, 2009.
- Guişu, S. Weighted entropy. *Reports on Mathematical Physics*, 2(3):165–179, 1971.
- Kadelburg, Z., Dukic, D., Lukic, M., and Matic, I. Inequalities of karamata, schur and muirhead, and some applications. *The Teaching of Mathematics*, 8(1):31–45, 2005.
- Kelbert, M., Stuhl, I., and Suhov, Y. Weighted entropy: basic inequalities. *Modern Stochastics: Theory and Applications*, 4(3):233–252, 2017. doi: 10.15559/17-VMSTA85. URL www.i-journals.org/vmsta.
- Marshall, A. W., Olkin, I., and Arnold, B. C. *Inequalities: theory of majorization and its applications*, volume 143. Springer, 1979.
- Petrov, F. Shannon entropy of $p(x)(1 - p(x))$ is no less than entropy of $p(x)$. MathOverflow. URL <https://mathoverflow.net/q/320726>. URL: <https://mathoverflow.net/q/320726> (version: 2019-01-12).

Chapter 7

Mutual Information-based State-Control for Intrinsically Motivated Reinforcement Learning

Mutual Information-based State-Control for Intrinsically Motivated Reinforcement Learning

Rui Zhao^{1,2,3} Volker Tresp^{2,3} Wei Xu¹

Abstract

In reinforcement learning, an agent learns to reach a set of goals by means of an external reward signal. In the natural world, intelligent organisms learn from internal drives, bypassing the need for external signals, which is beneficial for a wide range of tasks. Motivated by this observation, we propose to formulate an intrinsic objective as the mutual information between the goal states and the controllable states. This objective encourages the agent to take control of its environment. Subsequently, we derive a surrogate objective of the proposed reward function, which can be optimized efficiently. Lastly, we evaluate the developed framework in different robotic manipulation and navigation tasks and demonstrate the efficacy of our approach. A video showing experimental results is available at <https://youtu.be/CT4CKMWBZ0>.

1. Introduction

In psychology (Sansone & Harackiewicz, 2000), behavior is considered intrinsically motivated when it originates from an internal drive. An intrinsic motivation is essential to develop behaviors required for accomplishing a broad range of tasks rather than solving a specific problem guided by an external reward.

Intrinsically motivated reinforcement learning (Chentanez et al., 2005) equips an agent with various internal drives via intrinsic rewards, such as curiosity (Schmidhuber, 1991; Pathak et al., 2017; Burda et al., 2018), diversity (Gregor et al., 2016; Haarnoja et al., 2018; Eysenbach et al., 2019), and empowerment (Klyubin et al., 2005; Salge et al., 2014; Mohamed & Rezende, 2015), which allow the agent to de-

velop meaningful behaviors for solving a wide range of tasks. Mutual information is a core statistical quantity that has many applications in intrinsically motivated reinforcement learning. Still & Precup (2012) calculate the curiosity bonus based on the mutual information between the past and the future states within a time series. Mohamed & Rezende (2015) developed a scalable approach to calculate a common internal drive known as empowerment, which is defined as the channel capacity between the states and the actions. Eysenbach et al. (2019) use the mutual information between skills and states as an intrinsic reward to help the agent to discover a diverse set of skills. In multi-goal reinforcement learning (Schaul et al., 2015; Andrychowicz et al., 2017; Plappert et al., 2018), Warde-Farley et al. (2019) propose to utilize the mutual information between the high-dimensional observation and the goals as the reward signal to help the agent to learn goal-conditioned policies with visual inputs. To discover skills and learn the dynamics of these skills for model-based reinforcement learning, Sharma et al. (2020) recently designed an approach based on maximizing the mutual information between the next state and the current skill, conditioned on the current state.

In this paper, we investigate the idea that agent’s “preparedness” to control the states to reach any potential goal would be an effective intrinsic motivation for Reinforcement Learning (RL) agents. We formulate the “preparedness” of control as the mutual information between the goal states and agent’s controllable states. This internal drive extends agent’s controllability from controllable states to goal states and subsequently prepares the agent to reach any goal. It makes learning possible in the absence of hand-engineered reward functions or manually-specified goals. Furthermore, learning to “master” the environment potentially helps the agent to learn in sparse reward settings. We propose a new unsupervised reinforcement learning method called Mutual Information-based State-Control (MISC). During the learning process of the agent, a Mutual Information (MI) estimator is trained to evaluate the mutual information between the goal states and agent’s controllable states. Concurrently, the agent is rewarded for maximizing the MI estimation.

This paper contains the following five contributions. First, we introduce Mutual Information-based State-Control for in-

¹Horizon Robotics, Cupertino, California, United States

²Faculty of Mathematics, Informatics and Statistics, Ludwig Maximilian University of Munich, Munich, Bavaria, Germany ³Siemens AG, Munich, Bavaria, Germany. Correspondence to: Rui Zhao <zhaorui.in.germany@gmail.com>.

The work was done during an internship at Horizon Robotics. After the internship, the paper was revised. This paper is under review.

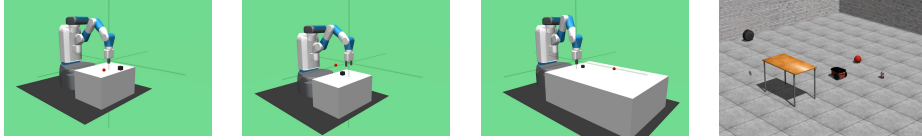


Figure 1. Fetch robot arm manipulation tasks provided by OpenAI Gym and a navigation task based on the Gazebo simulator: FetchPush, FetchPickAndPlace, FetchSlide, SocialBot-PlayGround.

trinsically motivated RL. Secondly, we derive a scalable MI surrogate objective for optimization. Thirdly, we evaluate the developed framework for the robotic tasks of manipulation and navigation and demonstrate the control behavior that agents learned purely via the intrinsic reward. Fourthly, incorporating the intrinsic reward with the task reward, we compare our approach with state-of-the-art methods. Last but not least, we observe that the learned MI estimator from one task can be transferred to a different task and still accelerate learning.

2. Preliminaries

2.1. Environments

We consider multi-goal reinforcement learning tasks, like the robotic simulation scenarios provided by OpenAI Gym (Plappert et al., 2018), where four tasks are used for evaluation, including push, slide, pick & place with the robot arm, and a newly designed navigation task with a mobile robot in Gazebo (Koenig & Howard, 2004), as shown in Figure 1. Accordingly, we define the following terminologies for these scenarios.

2.2. Goal States and Controllable States

The goals g in the manipulation tasks are the desired positions of the object. For the navigation task, the goal for the robot is to navigate to the ball. These goals are sampled from the environment. Note that in this paper we consider that the goals can be represented by states (Andrychowicz et al., 2017), which leads us to the concept of goal states s^g . The *goal state* s^g has the same dimension as the real goal from the environment but represents the achieved states of the object being manipulated or the target ball position in the navigation task. The *controllable state* s^c is the state that can be directly influenced by the agent (Borsa et al., 2019), such as the state of the robot and its end-effector. The goal states and the controllable states are mutually exclusive. The state split is under the designer’s control. Using states is common in reinforcement learning (Sutton & Barto, 2018).

2.3. Reinforcement Learning Settings

We consider an agent interacting with an environment. We assume the environment is fully observable, including a set

of state \mathcal{S} , a set of action \mathcal{A} , a distribution of initial states $p(s_0)$, transition probabilities $p(s_{t+1} | s_t, a_t)$, a reward function $r: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, and a discount factor $\gamma \in [0, 1]$.

2.4. Notations

In this paper, we use upper letters, such as S , to denote random variables and the corresponding lower case letter, such as s , to represent the values of random variables.

3. Method

We focus on agents learning to control goal states purely by using its observations and actions without supervision. Motivated by the idea that an agent capable of controlling the goal state s^g to obtain high mutual information with its controllable state s^c has been “prepared” to reach any future goal, we formulate the problem of learning without external supervision as one of learning a policy $\pi_\theta(a_t | s_t)$ with parameters θ to maximize intrinsic mutual information rewards, $r = I(S^g; S^c)$. In this section, we formally describe our method, mutual information-based state control.

3.1. Mutual Information Reward Function

Our framework simultaneously learns a policy and an intrinsic reward function by maximizing the mutual information between the goal states and the controllable states. Mathematically, the mutual information between the goal state random variable S^g and the controllable state random variable S^c is represented as Equation (1, 2):

$$I(S^g; S^c) = H(S^g) - H(S^g | S^c) \quad (1)$$

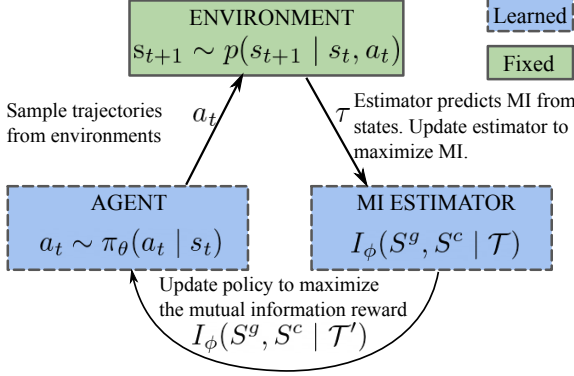
$$= D_{KL}(\mathbb{P}_{S^g S^c} || \mathbb{P}_{S^g} \otimes \mathbb{P}_{S^c}) \quad (2)$$

$$= \sup_{T: \Omega \rightarrow \mathbb{R}} \mathbb{E}_{\mathbb{P}_{S^g S^c}}[T] - \log(\mathbb{E}_{\mathbb{P}_{S^g} \otimes \mathbb{P}_{S^c}}[e^T]) \quad (3)$$

$$\geq \sup_{\phi \in \Phi} \mathbb{E}_{\mathbb{P}_{S^g S^c}}[T_\phi] - \log(\mathbb{E}_{\mathbb{P}_{S^g} \otimes \mathbb{P}_{S^c}}[e^{T_\phi}]) \quad (4)$$

$$= I_\Phi(S^g; S^c), \quad (5)$$

where $\mathbb{P}_{S^g S^c}$ is the joint probability distribution; $\mathbb{P}_{S^g} \otimes \mathbb{P}_{S^c}$ is the product of the marginal distributions \mathbb{P}_{S^g} and \mathbb{P}_{S^c} ; D_{KL} denotes the Kullback-Leibler (KL) divergence. Equation (1) tells us that the agent should maximize the entropy of goal states $H(S^g)$, and concurrently, should minimize the conditional entropy of goal states given the controllable states $H(S^g | S^c)$. When the conditional entropy


Algorithm 1 Mutual Information-based State Control (MISC)

while not converged do

 Sample an initial state $s_0 \sim p(s_0)$.

for $t \leftarrow 1$ **to** $steps_per_episode$ **do**

 Sample the action from policy $a_t \sim \pi_\theta(a_t | s_t)$.

 Step environments $s_{t+1} \sim p(s_{t+1} | s_t, a_t)$.

Update the replay buffer.

 Sample trajectories τ and transitions from the buffer.

 Set intrinsic MI reward $r = I_\phi(S^g; S^c | \mathcal{T}')$.

 Update agent's policy (θ) via DDPG or SAC.

 Update the MI estimator (ϕ) with SGD.

Figure 2. MISC Algorithm: We update the estimator to better predict the mutual information (MI), and update the agent to control goal states to have higher MI with the controllable states.

$H(S^g | S^c)$ is small, it becomes easy to predict the goal states based on the controllable states. For instance, we can see that the robot is controlling an object when it becomes easy to tell what the object state is, based on the robot state. Equation (2) gives us the mutual information in the KL divergence form.

Mutual information is notoriously difficult to compute in real-world settings (Hjelm et al., 2019). Motivated by Belghazi et al. (2018), we use a lower bound to approximate the mutual information quantity $I(S^g; S^c)$. First, we rewrite Equation (2), the KL formulation of the mutual information objective, using the Donsker-Varadhan representation (Donsker & Varadhan, 1975), to Equation (3). The input space Ω is a compact domain of \mathbb{R}^d , i.e., $\Omega \subset \mathbb{R}^d$, and the supremum is taken over all functions T such that the two expectations are finite. Secondly, we lower bound the mutual information in the Donsker-Varadhan representation with the compression lemma in the PAC-Bayes literature (Banerjee, 2006) and then derive Equation (4). The expectations in Equation (4) are estimated by using empirical samples from $\mathbb{P}_{S^g S^c}$ and $\mathbb{P}_{S^g} \otimes \mathbb{P}_{S^c}$. We can also sample the marginal distributions by shuffling the samples from the joint distribution along the axis (Belghazi et al., 2018). The derived mutual information reward function, $r = I_\phi(S^g; S^c)$, can be trained by gradient ascent. The statistics model T_ϕ is parameterized by a deep neural network with parameters $\phi \in \Phi$, which is capable of estimating the mutual information with arbitrary accuracy.

3.2. Efficient Learning State-Control

At the beginning of each episode, the agent takes actions a_t following a partially random policy, such as ϵ -greedy, to explore the environment and collects trajectories into a replay buffer. The trajectory τ contains a series of states, $\tau = \{s_1, s_2, \dots, s_{t^*}\}$, where t^* is the time horizon of the trajectory. Its random variable is denoted as \mathcal{T} . Each state

s_t consists of goal states s_t^g and controllable states s_t^c .

For training the mutual information estimator network, we first randomly sample the trajectory τ from the replay buffer. Then, the states s_t^c used for calculating the product of marginal distributions are sampled by shuffling the states s_t^c from the joint distribution along the temporal axis t within the trajectory, see Equation (6,7). Note that we calculate the mutual information by using the samples from the same trajectory. If the agent does not alter the goal states during the episode, then the mutual information between the goal states and the controllable states remains zero.

We use back-propagation to optimize the parameter (ϕ) to maximize the MI lower bound, see Equation (7). However, for evaluating the mutual information, this lower bound, Equation (7), is time-consuming to calculate because it needs to process on all the samples from the whole trajectory. To improve its scalability and efficiency, we derive a surrogate objective, Equation (11), which is computed much more efficiently. Each time, to calculate the MI reward for the transition $r = I_\phi(S^g; S^c | \mathcal{T}')$, the new objective only needs to calculate over a small fraction of the complete trajectory, τ' . The trajectory fraction, τ' , is defined as adjacent state pairs, $\tau' = \{s_t, s_{t+1}\}$, and \mathcal{T}' represents its corresponding random variable. The derivation of the new MI surrogate objective Equation (11) is shown as follows:

$$I_\phi(S^g; S^c | \mathcal{T}) \quad (6)$$

$$= \mathbb{E}_{\mathbb{P}_{S^g S^c | \mathcal{T}}} [T_\phi] - \log(\mathbb{E}_{\mathbb{P}_{S^g | \mathcal{T}} \otimes \mathbb{P}_{S^c | \mathcal{T}}} [e^{T_\phi}]) \quad (7)$$

$$\times \mathbb{E}_{\mathbb{P}_{S^g S^c | \mathcal{T}}} [T_\phi] - \mathbb{E}_{\mathbb{P}_{S^g | \mathcal{T}} \otimes \mathbb{P}_{S^c | \mathcal{T}}} [e^{T_\phi}] \quad (8)$$

$$= \mathbb{E}_{\mathbb{P}_{\mathcal{T}'}} [\mathbb{E}_{\mathbb{P}_{S^g S^c | \mathcal{T}'}} [T_\phi] - \mathbb{E}_{\mathbb{P}_{S^g | \mathcal{T}'} \otimes \mathbb{P}_{S^c | \mathcal{T}'}} [e^{T_\phi}]] \quad (9)$$

$$\times \mathbb{E}_{\mathbb{P}_{\mathcal{T}'}} [\mathbb{E}_{\mathbb{P}_{S^g S^c | \mathcal{T}'}} [T_\phi] - \log(\mathbb{E}_{\mathbb{P}_{S^g | \mathcal{T}'} \otimes \mathbb{P}_{S^c | \mathcal{T}'}} [e^{T_\phi}])] \quad (10)$$

$$= \mathbb{E}_{\mathbb{P}_{\mathcal{T}'}} [I_\phi(S^g; S^c | \mathcal{T}')], \quad (11)$$

where T_ϕ represents a neural network, whose inputs are state samples and the output is a scalar. For simplicity,

we use the symbol \times to denote a monotonically increasing relationship between two variables, for example, $\log(x) \times x$ means that as the value of x increases, the value of $\log(x)$ also increases and vice versa.

To decompose the lower bound Equation (7) into small parts, we make the following derivations, see Equation (8,9,10). Deriving from Equation (7) to Equation (8), we use the property that $\log(x) \times x$. Here, the new form, Equation (8), allows us to decompose the MI estimation into the expectation over MI estimations of each trajectory fractions, Equation (9). To be more specific, we move the implicit expectation over trajectory fractions in Equation (8) to the front, and then have Equation (9). The quantity inside the expectation over trajectory fractions is the MI estimation using only each trajectory fraction, see Equation (9). We use the property, $\log(x) \times x$, again to derive from Equation (9) to Equation (10).

The derived mutual information surrogate objective, Equation (11), brings us two important benefits. First, it enables us to estimate the MI reward for each transition with much less computational time because we only use the trajectory fraction, instead of the trajectory. This approximately reduces the complexity from $\mathcal{O}(t^*)$ to $\mathcal{O}(1)$ with respect to the trajectory length t^* . Secondly, this way of estimating MI also enables us to assign rewards more accurately at the transition level because now we use only the relevant state pair to calculate the transition reward.

Formally, we define the transition MI reward as the MI estimation of each trajectory fraction, $\tau' = \{s_t, s_{t+1}\}$, mathematically,

$$r_\phi(a_t, s_t) = \text{clip}[\alpha I_\phi(S^g; S^c | \mathcal{T}'), 0, 1],$$

where α is a scale hyper-parameter, which is tuned in conjunction with the learning rate, in case that the estimated MI value, $I_\phi(S^g; S^c | \mathcal{T}')$, is particularly small. The clipping function limits the range of the MI reward between 0 and 1.

3.3. Implementation

We combine MISC with both deep deterministic policy gradient (DDPG) (Lillicrap et al., 2016) and soft actor-critic (SAC) (Haarnoja et al., 2018) to learn a policy $\pi_\theta(a | s)$ that aims to control the goal states. In comparison to DDPG and SAC, the DDPG method improves the policy in a more “greedy” fashion, while the SAC approach is more conservative, in the sense that SAC incorporates an entropy regularizer $\mathcal{H}(A | S)$ that maximizes the policy’s entropy over actions.

3.4. Complete Algorithm

Overall, the agent is rewarded for controlling the goal states to have higher mutual information with its controllable

states, which is considered the “preparedness” to achieve any future goal. We summarize the complete training algorithm in Algorithm 1 and in Figure 2.

3.5. MISC Variants with Task Rewards

We propose three ways of using MISC to accelerate learning in addition to the task reward. The first method is using the MISC pretrained policy as the parameter initialization and fine-tuning the agent with rewards. We denote this variant as “MISC-f”, where “-f” stands for fine-tuning. The second variant is to use the MI intrinsic reward to help the agent to explore high mutual information states. We name this method as “MISC-r”, where “-r” stands for reward. The third approach is to use the mutual information quantity from MISC to prioritize trajectories for replay. We name this method as “MISC-p”, where “-p” stands for prioritization.

3.6. Skill Discovery with MISC and DIAYN

One of the most relevant works on unsupervised reinforcement learning, DIAYN (Eysenbach et al., 2019), introduces an information-theoretical objective $\mathcal{F}_{\text{DIAYN}}$, which learns diverse discriminable skills indexed by the latent variable Z , mathematically,

$$\begin{aligned} \mathcal{F}_{\text{DIAYN}} &= I(S; Z) + \mathcal{H}(A | S, Z) \\ &\geq \mathbb{E}_{\mathbb{P}_{Z\mathbb{P}_S}} [\log q_\phi(z | s) - \log p(z)] + \mathcal{H}(A | S, Z). \end{aligned}$$

The first term, $I(S; Z)$, in the objective, $\mathcal{F}_{\text{DIAYN}}$, is implemented via a skill discriminator, which serves as a variational lower bound of the original objective (Barber & Agakov, 2003; Eysenbach et al., 2019). The skill discriminator assigns high rewards to the agent, if it can predict the skill-options, Z , given the states, S . The second term, $\mathcal{H}(A | S, Z)$, is implemented through SAC (Haarnoja et al., 2018) conditioned on skill-options (Szepesvari et al., 2014).

We adapt DIAYN to goal-oriented tasks by replacing the full states, S , with goal states, S^g , as $I(S^g; Z)$. In comparison, our method MISC proposes to maximize the mutual information between the controllable states and the goal states, $I(S^c; S^g)$. These two methods can be combined as follows:

$$\mathcal{F}_{\text{MISC+DIAYN}} = I(S^c; S^g) + I(S^g; Z) + \mathcal{H}(A | S, Z).$$

The combination of MISC and DIAYN helps the agent to learn control primitives via skill-conditioned policy for hierarchical reinforcement learning (Eysenbach et al., 2019).

4. Experiments

To evaluate the proposed methods, we used the robotic manipulation tasks provided by OpenAI Gym and also a newly designed navigation task using Gazebo, see Figure 1 (Brockman et al., 2016; Plappert et al., 2018). First, we analyze the

control behaviors learned purely with the intrinsic reward (refer to the [video starting from 0:04](#) and Figure 8 in Appendix B). Secondly, we show that the pretrained models can be used for improving performance in conjunction with the task rewards. Interestingly, we show that the pretrained MI estimator can be transferred among different tasks and still improve performance. We compared MISC with other methods, including DDPG (Lillicrap et al., 2016), SAC (Haarnoja et al., 2018), DIAYN (Eysenbach et al., 2019), PER (Schaul et al., 2016), VIME (Houthoofd et al., 2016), ICM (Pathak et al., 2017), and Empowerment (Mohamed & Rezende, 2015). Thirdly, we show some insights about how the MISC rewards are distributed across a trajectory. The experimental details are shown in Appendix C. Our code is available at <https://github.com/ruizhaogit/misc> and <https://github.com/HorizonRobotics/alf>.

4.1. Analysis of the Learned Behaviors

Question 1. *What behavior does MISC learn?*

We tested MISC in the robotic manipulation tasks. The object is randomly placed on the table at the beginning of each episode. During training, the agent only receives the intrinsic MISC reward. In all three environments, the behavior of reaching objects emerges. In the push environments, the agent learns to push the object around on the table. In the slide environment, the agent learns to slide the object into different directions. Perhaps surprisingly, in the pick & place environment, the agent learns to pick up the object from the table without any task reward. All the observations are shown in the uploaded [video starting from 0:04](#).

We implemented MISC with both DDPG and SAC and ran the experiments with 5 different random seeds. To compare DDPG+MISC and SAC+MISC, we ran 20 trials using the learned policy in the pick & place environment with each seed. We observed that, in all the 5 random seed settings, SAC+MISC learns the picking-up behavior, while DDPG+MISC learns to pick up an object in only 1 out of 5 random seed settings. Mostly, the agent learns to push, flip, or grip the object. These observations show that the entropy bonus, $\mathcal{H}(A | S)$, of SAC can incorporate with MISC and helps the agent to better explore the behavior space.

Question 2. *Can we use learned behaviors to directly maximize the task reward?*

We tested our method in the navigation task, which is based on the Gazebo simulator. The task reward is 1 if the agent reaches the ball, otherwise, the task reward is 0. We combined our method with PPO (Schulman et al., 2017) and compared the performance with ICM (Pathak et al., 2017) and Empowerment (Mohamed & Rezende, 2015). During training, we only used one of the intrinsic rewards such as MISC, ICM, or Empowerment to train the agent. Then, we used the averaged task reward as the evaluation metric.

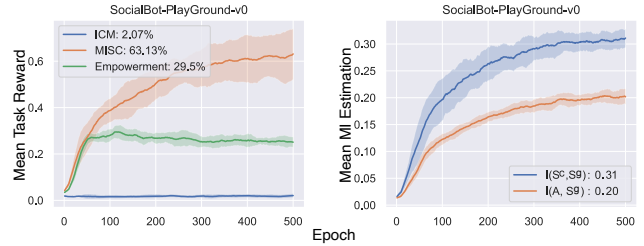


Figure 3. Experimental results in the navigation task

The experimental results are shown in Figure 3 (left). The y-axis represents the mean task reward and the x-axis denotes the training epochs. From Figure 3 (left), we can see that the proposed method, MISC, has the best performance. Empowerment has the second-best performance. Figure 3 (right) shows that the MISC reward signal $I(S^c, S^g)$ is relatively strong compared to the Empowerment reward signal $I(A, S^g)$. Subsequently, higher mutual information reward encourages the agent to explore more states with higher mutual information. A theoretical connection between Empowerment and MISC is shown in Appendix A. Furthermore, the ICM method does not enable the agent to navigate to the ball because it seeks only novel states and does not control these states. The uploaded [video starting from 1:44](#) shows the learned navigation behaviors.

Question 3. *How does MISC compare to DIAYN?*

We compared MISC, DIAYN and MISC+DIAYN in the pick & place environment. For implementing MISC+DIAYN, we first pre-train the agent with only MISC, and then fine-tune the policy with DIAYN. After pre-training, the MISC-trained agent learns manipulation behaviors such as, reaching, pushing, sliding, and picking up an object. Compared to MISC, the DIAYN-trained agent rarely learns to pick up the object. It mostly pushes or flicks the object with the gripper. However, the combined model, MISC+DIAYN, learns to pick up the object and moves it to different locations, depending on the skill-option. These observations are shown in the [video starting from 0:48](#). In short, MISC helps the agent to learn the DIAYN objective. The agent first learns to control the object with MISC, and then discovers diverse manipulation skills with DIAYN.

4.2. Accelerating Learning with MISC

Question 4. *How can we use the learned behaviors or the trained MI estimator to accelerate learning?*

We investigated three ways of using MISC to accelerate learning in addition to the task reward, see Section 3.5. We combined these three variants with DDPG and SAC and tested them in the multi-goal robotic tasks. The environments, including push, pick & place, and slide, have a set of predefined goals, which are represented as the red dots,

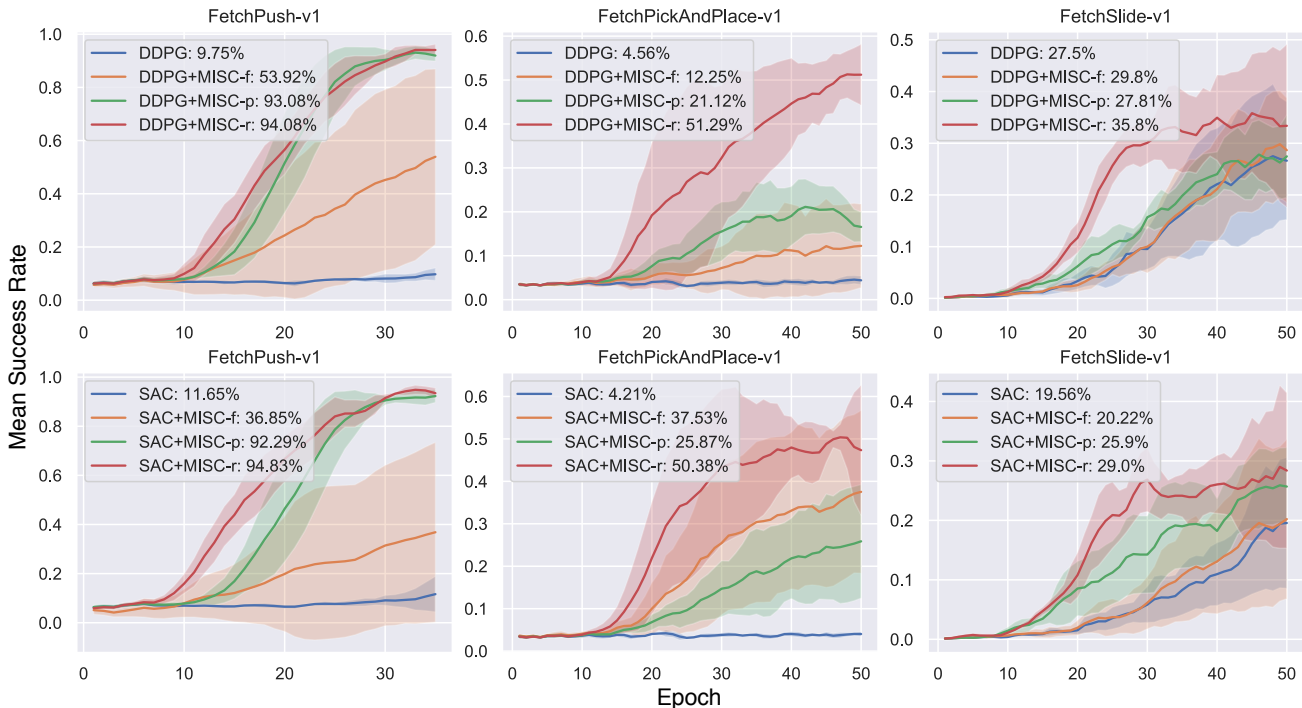


Figure 4. Mean success rate with standard deviation: The percentage values after colon (:) represent the best mean success rate during training. The shaded area describes the standard deviation.

as shown in Figure 1. The task for the RL agent is to manipulate the object to the goal positions. We ran all the methods in each environment with 5 different random seeds and report the mean success rate and the standard deviation, as shown in Figure 4. The percentage values alongside the plots are the best mean success rates during training. Each experiment is carried out with 16 CPU-cores.

From Figure 4, we can see that all these three methods, including MISC-f, MISC-p, and MISC-r, accelerate learning in the presence of task rewards. Among these variants, the MISC-r has the best overall improvements. In the push and pick & place tasks, MISC enables the agent to learn in a short period of training time. In the slide tasks, MISC-r also improves the performances by a decent margin.

We also compare our methods with more advanced RL methods. To be more specific, we compare MISC-f against the parameter initialization using DIAYN (Eysenbach et al., 2019); MISC-p against Prioritized Experience Replay (PER), which uses TD-errors for prioritization (Schaul et al., 2016); and MISC-r versus Variational Information Maximizing Exploration (VIME) (Houthoofd et al., 2016). The experimental results are shown in Figure 5. From Figure 5 (1st row), we can see that MISC-f enables the agent to learn, while DIAYN does not. In the 2nd row of Figure 5, MISC-r performs better than VIME. This result indicates that the

mutual information between states is a crucial quantity for accelerating learning. The mutual information intrinsic rewards boost performance significantly compared to VIME. This observation is consistent with the experimental results of MISC-p and PER, as shown in Figure 5 (3rd row), where the MI-based prioritization framework performs better than the TD-error-based approach, PER. On all tasks, MISC enables the agent to learn the benchmark task more quickly.

4.3. Transfer Learning with MISC

Question 5. Can the learned MI estimator be transferred to new tasks?

It would be beneficial if the pretrained MI estimator could be transferred to a new task and still improve the performance (Pan et al., 2010; Bengio, 2012). To verify this idea, we directly applied the pretrained MI estimator from the pick & place environment to the push and slide environments, respectively. We denote this transferred method as “MISC-t”, where “-t” stands for transfer. The MISC reward function trained in its corresponding environments is denoted as “MISC-r”. We compared the performances of DDPG baseline, MISC-r, and MISC-t. The results are shown in Figure 6. Perhaps surprisingly, the transferred MISC still improved the performance significantly. Furthermore, as expected, MISC-r performed better than MISC-t

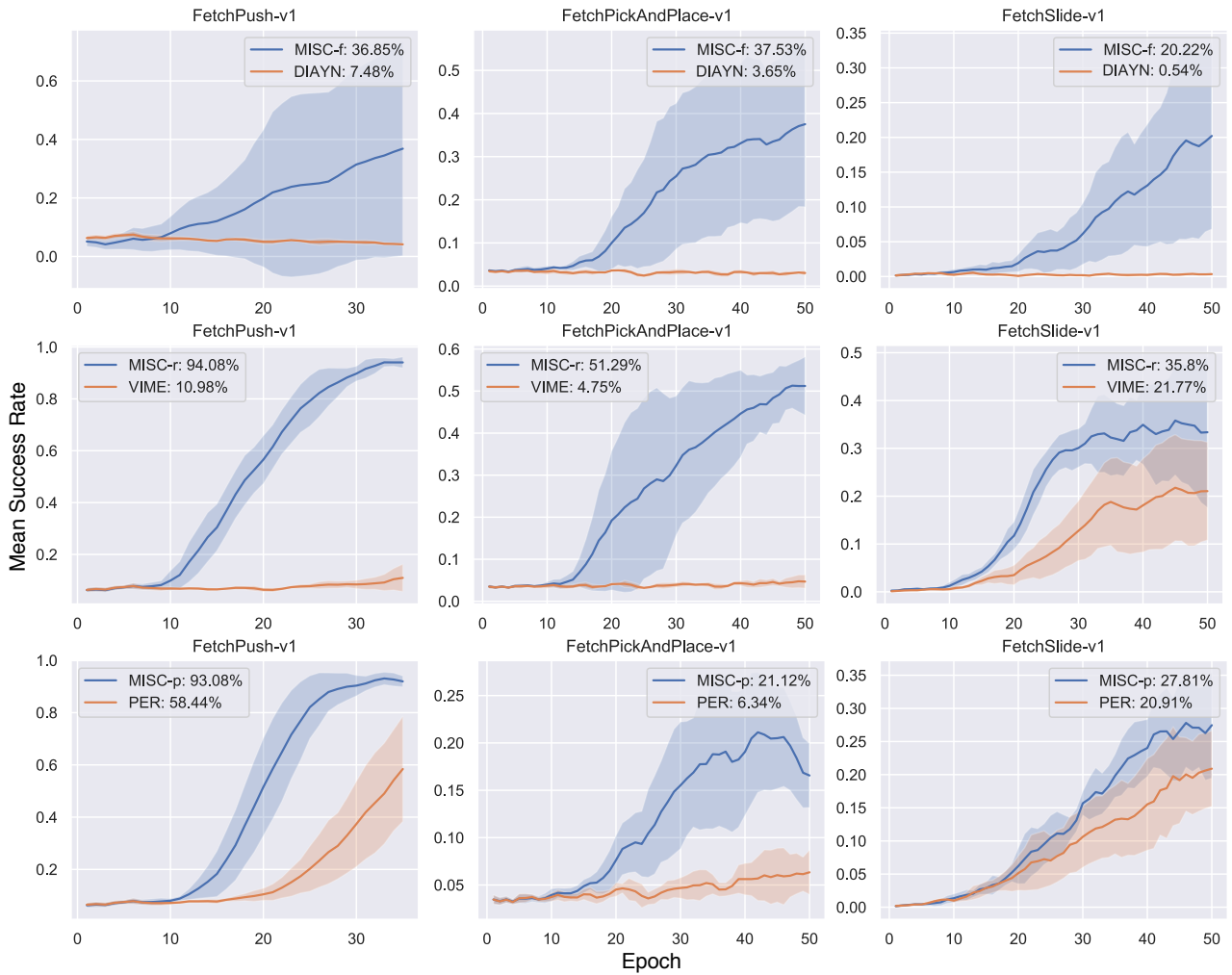


Figure 5. Performance comparison: We compare the MISC variants, including MISC-f, MISC-r, and MISC-p, with DIAYN, VIME, and PER, respectively.

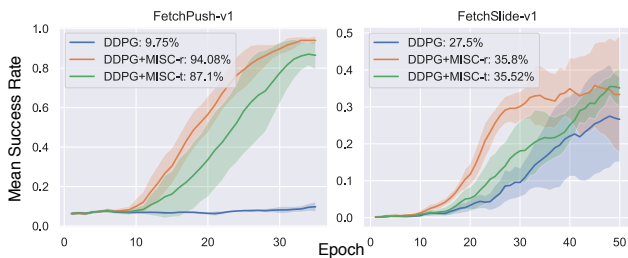


Figure 6. Transferred MISC

in both tasks. We can see that the MI estimator can be trained in a task-agnostic (Finn et al., 2017) fashion and later utilized in unseen tasks.

4.4. Insights and More

Question 6. How does MISC distribute rewards over a trajectory?

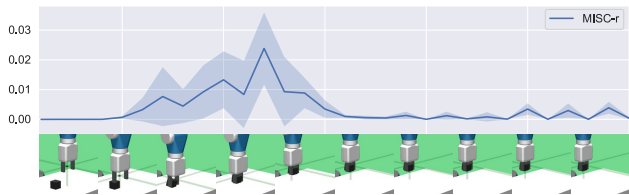


Figure 7. MISC rewards over a trajectory

To understand why MISC works and how MISC distributes rewards, we visualize the learned MISC rewards in Figure 7 and in the uploaded [video starting from 1:32](#). From Figure 7, we can observe that the mutual information reward peaks between the fourth and fifth frame, where the robot quickly picks up the cube from the table. Around the peak reward value, the middle range reward values are corresponding to the relatively slow movement of the object and the gripper (see the third, ninth, and tenth frame). When there is

no contact between the gripper and the cube (see the first two frames in Figure 7), or the gripper holds the object still (see the sixth to eighth frames) the intrinsic reward remains nearly zero. From this example, we see that MISC distributes positive intrinsic rewards when the goal state has correlated changes with the controllable state.

Question 7. *Can MISC help the agent to learn control behaviors when there are no objects involved?*

In the navigation task, we define the MISC objective to be the MI between the left wheel and the right wheel. We observe that the agent learns to balance itself and run in a straight line, as shown in the [video starting from 2:14](#).

Question 8. *What happens if there are multiple objects involved?*

When there are multiple objects to control, we define the MISC objective as follows:

$$\mathcal{F}_{\text{MISC}} = \sum_i I(S^c; S_i^g).$$

In the case that there is a red and a blue ball on the ground, with MISC, the agent learns to reach both balls and sometimes also learns to use one ball to hit the other ball. The results are shown in the uploaded [video starting from 2:29](#).

4.5. Summary

From these examples, we can see that, with different combinations of the goal states and the controllable states, the agent is able to learn different control behaviors. When there are no specific goal states involved, we can train a skill-conditioned policy corresponding to different combinations of the two sets of states and later use the pretrained policy for the tasks at hand.

5. Related Work

Deep RL led to great successes in various tasks (Ng et al., 2006; Peters & Schaal, 2008; Mnih et al., 2015; Levine et al., 2016; Zhao & Tresp, 2018a;c). However, RL via intrinsic motivation is still a challenging topic. Intrinsic rewards are often used to help the agent learn more efficiently to solve tasks. For example, Jung et al. (2011) and Mohamed & Rezende (2015) use empowerment, which is the channel capacity between states and actions, for intrinsically motivated RL agents. A theoretical connection between MISC and empowerment is shown in Appendix A. VIME (Houthoof et al., 2016) and ICM (Pathak et al., 2017) use curiosity as intrinsic rewards to encourage the agents to explore the environment more thoroughly.

Another line of work on intrinsic motivation for RL is to discover meaningful skills. Variational Intrinsic Control (VIC) (Gregor et al., 2016) proposes an information-theoretical objective (Barber & Agakov, 2003) to jointly maximize the

entropy of a set of options while keeping the options distinguishable based on the final states of the trajectory. Recently, Eysenbach et al. (2019) introduced DIAYN, which maximizes the MI between a fixed number of skill-options and the entire states of the trajectory. Eysenbach et al. (2019) show that DIAYN can scale to more complex tasks compared to VIC and provides a handful of low-level primitive skills as the basis for hierarchical RL.

Intrinsic motivation also helps the agent to learn goal-conditioned policies. Warde-Farley et al. (2019) proposed DISCERN, a method to learn a MI objective between the states and goals, which enables the agent to learn to achieve goals in environments with continuous high-dimensional observation spaces. Based on DISCERN, Pong et al. (2019) introduced Skew-fit, which adapts a maximum entropy strategy to sample goals from the replay buffer (Zhao & Tresp, 2019; Zhao et al., 2019) in order to make the agent learn more efficiently in the absence of rewards. More recently, Hartikainen et al. (2019) proposed to automatically learn dynamical distances, which are defined as a measure of the expected number of time steps to reach a given goal that can be used as intrinsic rewards for accelerating learning to achieve goals.

Based on a similar motivation as previous works, we introduce MISC, a method that uses the MI between the goal states and the controllable states as intrinsic rewards. MISC enables the agent to learn control behaviors without supervision. Our method is complementary to the previous works, such as DIAYN, and can be combined with them. The idea of MISC is to encourage the agent to learn to be “prepared” to reach any goal, as one step forward towards mastery of the environment. Inspired by previous works (Schaul et al., 2016; Houthoof et al., 2016; Zhao & Tresp, 2018b; Eysenbach et al., 2019), we additionally demonstrate three variants, including MISC-based fine-tuning, rewarding, and prioritizing mechanisms, to accelerate learning in the case when the task rewards are available.

6. Conclusion

This paper introduces Mutual Information-based State-Control (MISC), an unsupervised RL framework for learning useful control behaviors. The derived efficient mutual information-based theoretical objective encourages the agent to control states without any task reward. MISC enables the agent to self-learn different control behaviors, which are non-trivial, intuitively meaningful, and useful for learning and planning. Additionally, the pretrained policy and the mutual information estimator significantly accelerate learning in the presence of task rewards. We evaluated three MISC-based variants in different environments and demonstrate a substantial improvement in learning efficiency compared to state-of-the-art methods.

References

- Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, O. P., and Zaremba, W. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pp. 5048–5058, 2017.
- Banerjee, A. On bayesian bounds. In *Proceedings of the 23rd international conference on Machine learning*, pp. 81–88. ACM, 2006.
- Barber, D. and Agakov, F. V. The im algorithm: a variational approach to information maximization. In *Advances in neural information processing systems*, pp. None, 2003.
- Belghazi, M. I., Baratin, A., Rajeswar, S., Ozair, S., Bengio, Y., Courville, A., and Hjelm, R. D. Mine: mutual information neural estimation. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 531–540. PMLR, 2018.
- Bengio, Y. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pp. 17–36, 2012.
- Borsa, D., Heess, N., Piot, B., Liu, S., Hasenclever, L., Munos, R., and Pietquin, O. Observational learning by reinforcement learning. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 1117–1124. International Foundation for Autonomous Agents and Multiagent Systems, 2019.
- Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.
- Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., and Efros, A. A. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018.
- Chentanez, N., Barto, A. G., and Singh, S. P. Intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pp. 1281–1288, 2005.
- Donsker, M. D. and Varadhan, S. S. Asymptotic evaluation of certain markov process expectations for large time, i. *Communications on Pure and Applied Mathematics*, 28(1):1–47, 1975.
- Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=SJx63jRqFm>.
- Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1126–1135. JMLR. org, 2017.
- Gregor, K., Rezende, D. J., and Wierstra, D. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.
- Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1861–1870. PMLR, 2018.
- Hartikainen, K., Geng, X., Haarnoja, T., and Levine, S. Dynamical distance learning for unsupervised and semi-supervised skill discovery. *arXiv preprint arXiv:1907.08225*, 2019.
- Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., and Bengio, Y. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=Bklr3j0cKX>.
- Houthoofd, R., Chen, X., Duan, Y., Schulman, J., De Turck, F., and Abbeel, P. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pp. 1109–1117, 2016.
- Jung, T., Polani, D., and Stone, P. Empowerment for continuous agent—environment systems. *Adaptive Behavior*, 19(1):16–39, 2011.
- Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- Klyubin, A. S., Polani, D., and Nehaniv, C. L. Empowerment: A universal agent-centric measure of control. In *2005 IEEE Congress on Evolutionary Computation*, volume 1, pp. 128–135. IEEE, 2005.
- Koenig, N. and Howard, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pp. 2149–2154. IEEE, 2004.
- Kraskov, A., Stögbauer, H., and Grassberger, P. Estimating mutual information. *Physical review E*, 69(6):066138, 2004.
- Levine, S., Finn, C., Darrell, T., and Abbeel, P. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

- Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529, 2015.
- Mohamed, S. and Rezende, D. J. Variational information maximisation for intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pp. 2125–2133, 2015.
- Ng, A. Y., Coates, A., Diel, M., Ganapathi, V., Schulte, J., Tse, B., Berger, E., and Liang, E. Autonomous inverted helicopter flight via reinforcement learning. In *Experimental Robotics IX*, pp. 363–372. Springer, 2006.
- Pan, S. J., Yang, Q., et al. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.
- Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning (ICML)*, volume 2017, 2017.
- Peters, J. and Schaal, S. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697, 2008.
- Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P., et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.
- Pong, V. H., Dalal, M., Lin, S., Nair, A., Bahl, S., and Levine, S. Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*, 2019.
- Salge, C., Glackin, C., and Polani, D. Empowerment—an introduction. In *Guided Self-Organization: Inception*, pp. 67–114. Springer, 2014.
- Sansone, C. and Harackiewicz, J. M. *Intrinsic and extrinsic motivation: The search for optimal motivation and performance*. Elsevier, 2000.
- Schaul, T., Horgan, D., Gregor, K., and Silver, D. Universal value function approximators. In *International Conference on Machine Learning*, pp. 1312–1320, 2015.
- Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. In *International Conference on Learning Representations*, 2016.
- Schmidhuber, J. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pp. 222–227, 1991.
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- Sharma, A., Gu, S., Levine, S., Kumar, V., and Hausman, K. Dynamics-aware unsupervised skill discovery. In *Proceeding of the International Conference on Learning Representations (ICLR), Addis Ababa, Ethiopia*, pp. 26–30, 2020.
- Still, S. and Precup, D. An information-theoretic approach to curiosity-driven reinforcement learning. *Theory in Biosciences*, 131(3):139–148, 2012.
- Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.
- Szepesvari, C., Sutton, R. S., Modayil, J., Bhatnagar, S., et al. Universal option models. In *Advances in Neural Information Processing Systems*, pp. 990–998, 2014.
- Warde-Farley, D., Van de Wiele, T., Kulkarni, T., Ionescu, C., Hansen, S., and Mnih, V. Unsupervised control through non-parametric discriminative rewards. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=r1eVMnA9K7>.
- Zhao, R. and Tresp, V. Efficient dialog policy learning via positive memory retention. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pp. 823–830. IEEE, 2018a.
- Zhao, R. and Tresp, V. Energy-based hindsight experience prioritization. In *Proceedings of the 2nd Conference on Robot Learning*, pp. 113–122, 2018b.
- Zhao, R. and Tresp, V. Learning goal-oriented visual dialog via tempered policy gradient. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pp. 868–875. IEEE, 2018c.
- Zhao, R. and Tresp, V. Curiosity-driven experience prioritization via density estimation. *arXiv preprint arXiv:1902.08039*, 2019.
- Zhao, R., Sun, X., and Tresp, V. Maximum entropy-regularized multi-goal reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 7553–7562. PMLR, 2019.

A. Connection to Empowerment

The state S contains the goal state S^g and the controllable state S^c . For example, in robotic tasks, the goal state and the controllable state represent the object state and the end-effector state, respectively. The action space is the change of the gripper position and the status of the gripper, such as open or closed. Note that, the agent’s action directly alters the controllable state.

Here, given the assumption that the transform, $S^c = F(A)$, from the action, A , to the controllable state, S^c , is a smooth and uniquely invertible mapping (Kraskov et al., 2004), then we can prove that the MISC objective, $I(S^c, S^g)$, is equivalent to the empowerment objective, $I(A, S^g)$.

The empowerment objective (Klyubin et al., 2005; Salge et al., 2014; Mohamed & Rezende, 2015) is defined as the channel capacity in information theory, which means the amount of information contained in the action A about the state S , mathematically:

$$\mathcal{E} = I(S, A). \quad (12)$$

Here, we replace the state variable S with goal state S^g , we have the empowerment objective as follows,

$$\mathcal{E} = I(S^g, A). \quad (13)$$

Theorem 1. *The MISC objective, $I(S^c, S^g)$, is equivalent to the empowerment objective, $I(A, S^g)$, given the assumption that the transform, $S^c = F(A)$, is a smooth and uniquely invertible mapping:*

$$I(S^c, S^g) = I(A, S^g) \quad (14)$$

where S^g , S^c , and A denote the goal state, the controllable state, and the action, respectively.

Proof.

$$I(S^c, S^g) = \int \int ds^c ds^g p(s^c, s^g) \log \frac{p(s^c, s^g)}{p(s^c)p(s^g)} \quad (15)$$

$$= \int \int ds^c ds^g \left\| \frac{\partial A}{\partial S^c} \right\| p(a, s^g) \log \frac{\left\| \frac{\partial A}{\partial S^c} \right\| p(a, s^g)}{\left\| \frac{\partial A}{\partial S^c} \right\| p(a)p(s^g)} \quad (16)$$

$$= \int \int ds^c ds^g J_A(s^c) p(a, s^g) \log \frac{J_A(s^c) p(a, s^g)}{J_A(s^c) p(a) p(s^g)} \quad (17)$$

$$= \int \int da ds^g p(a, s^g) \log \frac{p(a, s^g)}{p(a)p(s^g)} \quad (18)$$

$$= I(A, S^g) \quad (19)$$

□

B. Learned Control Behaviors without Supervision

The learned control behaviors without supervision are shown in Figure 8

C. Experimental Details

The experiments of the robotic manipulation tasks in this paper use the following hyper-parameters:

- Actor and critic networks: 3 layers with 256 units each and ReLU non-linearities
- Adam optimizer (Kingma & Ba, 2014) with $1 \cdot 10^{-3}$ for training both actor and critic
- Buffer size: 10^6 transitions

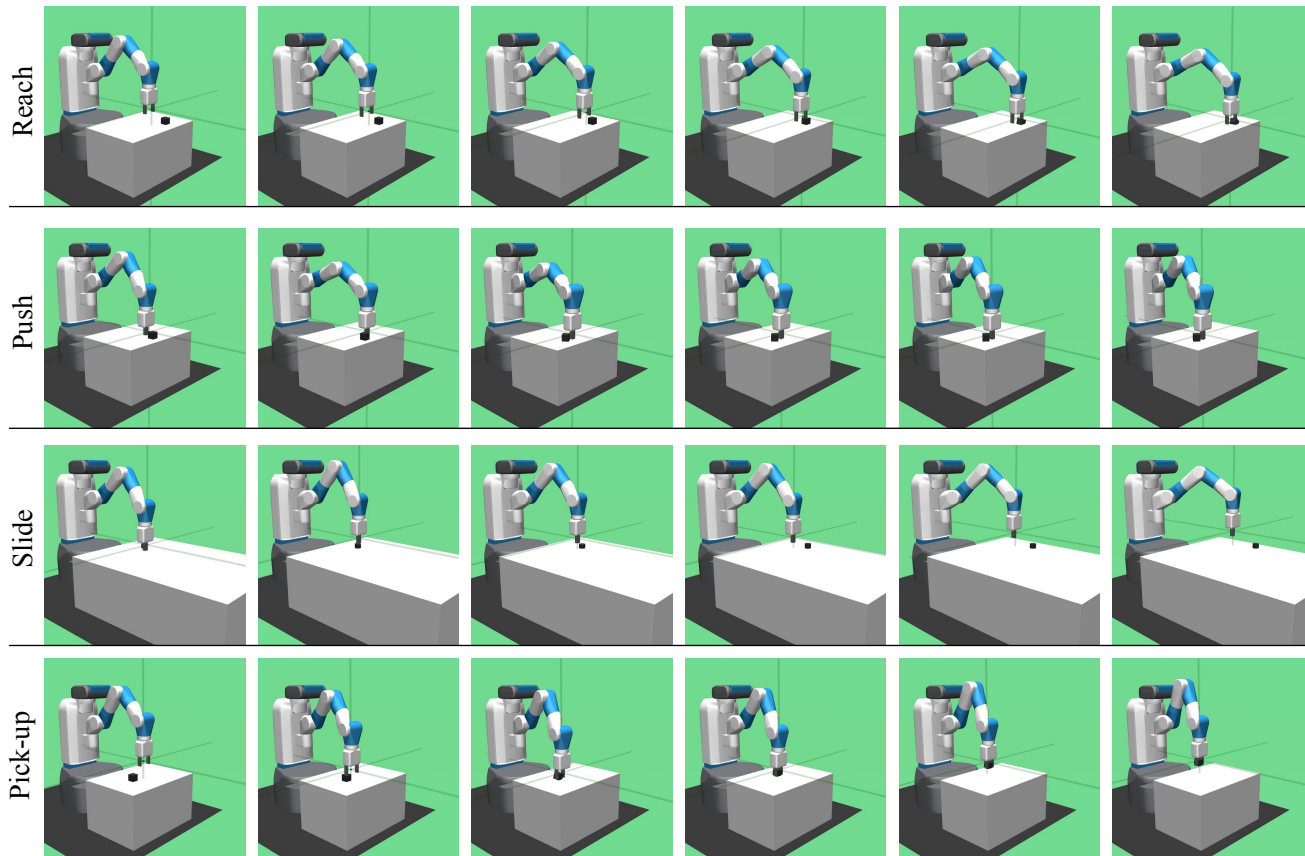


Figure 8. **Learned Control behaviors with MISC:** Without any reward, MISC enables the agent to learn control behaviors, such as reaching, pushing, sliding, and picking up an object. The learned behaviors are shown in the uploaded video starting from 0:04.

- Polyak-averaging coefficient: 0.95
- Action L2 norm coefficient: 1.0
- Observation clipping: $[-200, 200]$
- Batch size: 256
- Rollouts per MPI worker: 2
- Number of MPI workers: 16
- Cycles per epoch: 50
- Batches per cycle: 40
- Test rollouts per epoch: 10
- Probability of random actions: 0.3
- Scale of additive Gaussian noise: 0.2
- Scale of the mutual information reward: 5000

All hyper-parameters are described in greater detail at <https://github.com/ruizhaogit/misc/tree/master/params>.

Bibliography

Bibliography

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, page 5048–5058, 2017.

Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *arXiv preprint arXiv:1808.00177*, 2018.

Kavosh Asadi and Michael L Littman. An alternative softmax operator for reinforcement learning. *arXiv preprint arXiv:1612.05628*, 2016.

Kavosh Asadi and Jason D Williams. Sample-efficient deep reinforcement learning for dialog control. *arXiv preprint arXiv:1612.06000*, 2016.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Bram Bakker and Jürgen Schmidhuber. Hierarchical reinforcement learning based

- on subgoal discovery and subpolicy specialization. In *Proc. of the 8-th Conf. on Intelligent Autonomous Systems*, pages 438–445, 2004.
- Arindam Banerjee. On bayesian bounds. In *Proceedings of the 23rd international conference on Machine learning*, pages 81–88. ACM, 2006.
- David Barber and Felix V Agakov. The im algorithm: a variational approach to information maximization. In *Advances in neural information processing systems*, page None, 2003.
- Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeswar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and R Devon Hjelm. Mine: mutual information neural estimation. In *Proceedings of the 35th International Conference on Machine Learning*, pages 531–540. PMLR, 2018.
- Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016.
- Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009.
- Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 17–36, 2012.
- Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.
- Jose-Luis Blanco. A tutorial on se (3) transformation parameterizations and on-manifold optimization. *University of Malaga, Tech. Rep*, 3, 2010.
- David M Blei, Michael I Jordan, et al. Variational inference for dirichlet process mixtures. *Bayesian analysis*, 1(1):121–143, 2006.

- Antoine Bordes, Y-Lan Boureau, and Jason Weston. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*, 2016.
- Diana Borsa, Nicolas Heess, Bilal Piot, Siqu Liu, Leonard Hasenclever, Remi Munos, and Olivier Pietquin. Observational learning by reinforcement learning. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1117–1124. International Foundation for Autonomous Agents and Multiagent Systems, 2019.
- Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.
- Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259, 2018.
- Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018.
- David Carmel and Shaul Markovitch. Exploration strategies for model-based learning in multi-agent systems: Exploration strategies. *Autonomous Agents and Multi-agent systems*, 2(2):141–172, 1999.
- Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.
- Rich Caruana. Multitask learning. In *Learning to learn*, page 95–133. Springer, 1998.
- Prithvijit Chattopadhyay, Deshraj Yadav, Viraj Prabhu, Arjun Chandrasekaran, Abhishek Das, Stefan Lee, Dhruv Batra, and Devi Parikh. Evaluating visual conversational agents via cooperative human-ai games. In *Conference on Human Computation and Crowdsourcing (HCOMP)*, 2017.

- Yevgen Chebotar, Mrinal Kalakrishnan, Ali Yahya, Adrian Li, Stefan Schaal, and Sergey Levine. Path integral guided policy search. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 3381–3388. IEEE, 2017.
- Lu Chen, Xiang Zhou, Cheng Chang, Runzhe Yang, and Kai Yu. Agent-aware dropout dqn for safe and efficient on-line dialogue policy learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2454–2464, 2017.
- Nuttapong Chentanez, Andrew G Barto, and Satinder P Singh. Intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pages 1281–1288, 2005.
- Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.
- R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011.
- Bruno Da Silva, George Konidaris, and Andrew Barto. Learning parameterized skills. *arXiv preprint arXiv:1206.6398*, 2012.
- Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José MF Moura, Devi Parikh, and Dhruv Batra. Visual dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 326–335, 2017a.
- Abhishek Das, Satwik Kottur, José M.F. Moura, Stefan Lee, and Dhruv Batra. Learning cooperative visual dialog agents with deep reinforcement learning. In *International Conference on Computer Vision (ICCV)*, 2017b.

- Harm de Vries, Florian Strub, Sarath Chandar, Olivier Pietquin, Hugo Larochelle, and Aaron C. Courville. Guesswhat?! visual object discovery through multi-modal dialogue. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- Thomas Degris, Martha White, and Richard S Sutton. Off-policy actor-critic. In *International Conference on Machine Learning (ICML)*, 2012.
- Marc Peter Deisenroth, Peter Englert, Jan Peters, and Dieter Fox. Multi-task policy search for robotics. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 3876–3881. IEEE, 2014.
- Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.
- Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee, 2009.
- Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. Learning modular neural network policies for multi-task and multi-robot transfer. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2169–2176. IEEE, 2017.
- Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines. <https://github.com/openai/baselines>, 2017.
- Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. End-to-end reinforcement learning of dialogue agents for information access. *arXiv preprint arXiv:1609.00777*, 2016.

- Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430, 2015.
- Monroe D Donsker and SR Srinivasa Varadhan. Asymptotic evaluation of certain markov process expectations for large time, i. *Communications on Pure and Applied Mathematics*, 28(1):1–47, 1975.
- Richard O Duda and Peter E Hart. Pattern classification and scene analysis. *A Wiley-Interscience Publication, New York: Wiley, 1973*, 1973.
- Sašo Džeroski, Luc De Raedt, and Kurt Driessens. Relational reinforcement learning. *Machine learning*, 43(1-2):7–52, 2001.
- Jeffrey L Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99, 1993.
- Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. *arXiv preprint arXiv:1802.01561*, 2018.
- Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=SJx63jRqFm>.
- Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.
- Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

- Carlos Florensa, David Held, Markus Wulfmeier, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. *arXiv preprint arXiv:1707.05300*, 2017.
- Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *International Conference on Machine Learning*, pages 1514–1523, 2018.
- Sébastien Forestier, Yoan Mollard, and Pierre-Yves Oudeyer. Intrinsically motivated goal exploration processes with automatic curriculum learning. *arXiv preprint arXiv:1708.02190*, 2017.
- David Foster and Peter Dayan. Structure in the space of value functions. *Machine Learning*, 49(2-3):325–346, 2002.
- Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484, 2012.
- Michael S Gazzaniga, Richard B Ivry, and GR Mangun. *Cognitive Neuroscience. The biology of the mind, (2014)*. Norton: New York, 2006.
- Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.
- Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.
- Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

- Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.
- Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. *arXiv preprint arXiv:1704.03003*, 2017.
- Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.
- Matthias J Gruber, Bernard D Gelman, and Charan Ranganath. States of curiosity modulate hippocampus-dependent learning via the dopaminergic circuit. *Neuron*, 84(2):486–496, 2014.
- Matthias J Gruber, Maureen Ritchey, Shao-Fang Wang, Manoj K Doss, and Charan Ranganath. Post-learning hippocampal dynamics promote preferential retention of rewarding events. *Neuron*, 2016.
- Audrunas Gruslys, Mohammad Gheshlaghi Azar, Marc G Bellemare, and Remi Munos. The reactor: A sample-efficient actor-critic architecture. *arXiv preprint arXiv:1704.04651*, 2017.
- Shixiang Gu, Tim Lillicrap, Richard E Turner, Zoubin Ghahramani, Bernhard Schölkopf, and Sergey Levine. Interpolated policy gradient: Merging on-policy and off-policy gradient estimation for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, page 3849–3858, 2017.
- Silviu Guiaşu. Weighted entropy. *Reports on Mathematical Physics*, 2(3):165–179, 1971.
- David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

- Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. *arXiv preprint arXiv:1702.08165*, 2017.
- Tuomas Haarnoja, Kristian Hartikainen, Pieter Abbeel, and Sergey Levine. Latent space policies for hierarchical reinforcement learning. *arXiv preprint arXiv:1804.02808*, 2018a.
- Tuomas Haarnoja, Vitchyr Pong, Aurick Zhou, Murtaza Dalal, Pieter Abbeel, and Sergey Levine. Composable deep reinforcement learning for robotic manipulation. *arXiv preprint arXiv:1803.06773*, 2018b.
- Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1861–1870. PMLR, 2018c.
- Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J Russell, and Anca Dragan. Inverse reward design. In *Advances in neural information processing systems*, pages 6765–6774, 2017.
- Kristian Hartikainen, Xinyang Geng, Tuomas Haarnoja, and Sergey Levine. Dynamical distance learning for unsupervised and semi-supervised skill discovery. *arXiv preprint arXiv:1907.08225*, 2019.
- Frank S He, Yang Liu, Alexander G Schwing, and Jian Peng. Learning to play in a day: Faster deep reinforcement learning by optimality tightening. *arXiv preprint arXiv:1611.01606*, 2016.
- Haibo He and Eduardo A Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge & Data Engineering*, (9):1263–1284, 2008.
- David Held, Xinyang Geng, Carlos Florensa, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. *arXiv preprint arXiv:1705.06366*, 2017.

- Geoffrey E Hinton. To recognize shapes, first learn to generate images. *Progress in brain research*, 165:535–547, 2007.
- R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=Bklr3j0cKX>.
- Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.
- Rein Houthoofd, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117, 2016.
- David R Hunter and Kenneth Lange. A tutorial on mm algorithms. *The American Statistician*, 58(1):30–37, 2004a.
- David R Hunter and Kenneth Lange. A tutorial on mm algorithms. *The American Statistician*, 58(1):30–37, 2004b. doi: 10.1198/0003130042836. URL <https://doi.org/10.1198/0003130042836>.
- Xu Jia, Efstratios Gavves, Basura Fernando, and Tinne Tuytelaars. Guiding long-short term memory for image caption generation. *arXiv preprint arXiv:1509.04942*, 2015.
- Tang Jie and Pieter Abbeel. On a connection between importance sampling and the likelihood ratio policy gradient. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.
- Tobias Jung, Daniel Polani, and Peter Stone. Empowerment for continuous agent—environment systems. *Adaptive Behavior*, 19(1):16–39, 2011.

- Zoran Kadelburg, Dusan Dukic, Milivoje Lukic, and Ivan Matic. Inequalities of karamata, schur and muirhead, and some applications. *The Teaching of Mathematics*, 8(1):31–45, 2005.
- Leslie Pack Kaelbling. Hierarchical learning in stochastic domains: Preliminary results. In *Proceedings of the tenth international conference on machine learning*, volume 951, pages 167–173, 1993.
- Kirthivasan Kandasamy, Yoram Bachrach, Ryota Tomioka, Daniel Tarlow, and David Carter. Batch policy gradient methods for improving neural conversation models. In *International Conference on Learning Representations (ICLR)*, 2017.
- Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137, 2015.
- Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.
- Mark Kelbert, Izabella Stuhl, and Yuri Suhov. Weighted entropy: basic inequalities. *Modern Stochastics: Theory and Applications*, 4(3):233–252, 2017. doi: 10.15559/17-VMSTA85. URL www.i-journals.org/vmsta.
- Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- Scott Kirkpatrick, C Daniel Gelatt, Mario P Vecchi, et al. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.
- Alexander S Klyubin, Daniel Polani, and Chrystopher L Nehaniv. Empowerment: A universal agent-centric measure of control. In *2005 IEEE Congress on Evolutionary Computation*, volume 1, pages 128–135. IEEE, 2005.

- Jens Kober, Andreas Wilhelm, Erhan Oztop, and Jan Peters. Reinforcement learning to adjust parametrized motor primitives to new situations. *Autonomous Robots*, 33(4):361–379, 2012.
- Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2149–2154. IEEE, 2004.
- Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical review E*, 69(6):066138, 2004.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40, 2017.
- Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
- Oliver Lemon, Xingkun Liu, Daniel Shapiro, and Carl Tollander. Hierarchical reinforcement learning of dialogue policies in a development environment for dialogue systems: Reall-dude. In *BRANDIAL’06, Proceedings of the 10th Workshop on the Semantics and Pragmatics of Dialogue*, pages 185–186, 2006.
- Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge university press, 2014.

- Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.
- Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.
- Mike Lewis, Denis Yarats, Yann N Dauphin, Devi Parikh, and Dhruv Batra. Deal or no deal? end-to-end learning for negotiation dialogues. *arXiv preprint arXiv:1706.05125*, 2017.
- Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*, 2016.
- Xuijun Li, Yun-Nung Chen, Lihong Li, and Jianfeng Gao. End-to-end task-completion neural dialogue systems. *arXiv preprint arXiv:1703.01008*, 2017.
- Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.
- Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 1991.
- Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.
- Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

- Zachary Lipton, Xiujun Li, Jianfeng Gao, Lihong Li, Faisal Ahmed, and Li Deng. Bbq-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems. *arXiv preprint arXiv:1711.05715*, 2017.
- Zachary C Lipton, Jianfeng Gao, Lihong Li, Xiujun Li, Faisal Ahmed, and Li Deng. Efficient exploration for dialogue policy learning with bbq networks & replay buffer spiking. *arXiv preprint arXiv:1608.05081*, 2016.
- Yang Liu, Prajit Ramachandran, Qiang Liu, and Jian Peng. Stein variational policy gradient. *arXiv preprint arXiv:1704.02399*, 2017.
- Manuel Lopes, Tobias Lang, Marc Toussaint, and Pierre-Yves Oudeyer. Exploration in model-based reinforcement learning by empirically estimating learning progress. In *Advances in Neural Information Processing Systems*, pages 206–214, 2012.
- Albert W Marshall, Ingram Olkin, and Barry C Arnold. *Inequalities: theory of majorization and its applications*, volume 143. Springer, 1979.
- James L Meriam and L Glenn Kraige. *Engineering mechanics: dynamics*, volume 2. John Wiley & Sons, 2012.
- Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, 2010.
- Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. Learning to navigate in complex environments. In *International Conference on Learning Representations*, 2017.
- Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212, 2014.

- Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.
- Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.
- Shakir Mohamed and Danilo Jimenez Rezende. Variational information maximisation for intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pages 2125–2133, 2015.
- Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc Bellemare. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, page 1054–1062, 2016.
- Kevin P Murphy. Machine learning: A probabilistic perspective. adaptive computation and machine learning, 2012.
- Ofir Nachum, Mohammad Norouzi, and Dale Schuurmans. Improving policy gradient by exploring under-appreciated rewards. *arXiv preprint arXiv:1611.09321*, 2016.
- Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*, pages 9191–9200, 2018.
- Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. Language understanding for text-based games using deep reinforcement learning. *arXiv preprint arXiv:1506.08941*, 2015.
- Radford M Neal. Sampling from multimodal distributions using tempered transitions. *Statistics and computing*, 6(4):353–366, 1996.

- Gergely Neu, Anders Jonsson, and Vicenç Gómez. A unified view of entropy-regularized markov decision processes. *arXiv preprint arXiv:1705.07798*, 2017.
- Andrew Y Ng, Adam Coates, Mark Diel, Varun Ganapathi, Jamie Schulte, Ben Tse, Eric Berger, and Eric Liang. Autonomous inverted helicopter flight via reinforcement learning. In *Experimental Robotics IX*, page 363–372. Springer, 2006.
- Pierre-Yves Oudeyer and Frederic Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 1:6, 2009.
- Pierre-Yves Oudeyer, Frdric Kaplan, and Verena V Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(2):265–286, 2007.
- Art B. Owen. *Monte Carlo theory, methods and examples*. 2013.
- Sinno Jialin Pan, Qiang Yang, et al. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.
- Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.
- Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning (ICML)*, volume 2017, 2017.
- Alexandre Péré, Sébastien Forestier, Olivier Sigaud, and Pierre-Yves Oudeyer. Unsupervised learning of goal spaces for intrinsically motivated goal exploration. *arXiv preprint arXiv:1803.00781*, 2018.
- Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.

- Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697, 2008.
- Fedor Petrov. Shannon entropy of $p(x)(1 - p(x))$ is no less than entropy of $p(x)$. MathOverflow. URL <https://mathoverflow.net/q/320726>. URL:<https://mathoverflow.net/q/320726> (version: 2019-01-12).
- Lerrel Pinto and Abhinav Gupta. Learning to push by grasping: Using multiple tasks for effective learning. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2161–2168. IEEE, 2017.
- Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.
- Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.
- Vitchyr H Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*, 2019.
- Pascal Poupart, Nikos Vlassis, Jesse Hoey, and Kevin Regan. An analytic solution to discrete bayesian reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 697–704. ACM, 2006.
- Paulo Rauber, Filipe Mutz, and Juergen Schmidhuber. Hindsight policy gradients. *arXiv preprint arXiv:1711.06006*, 2017.
- Christian P Robert. *Monte carlo methods*. Wiley Online Library, 2004.
- Murray Rosenblatt. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, pages 832–837, 1956.

- Alexander Rudnicky and Wei Xu. An agenda-based dialog management architecture for spoken language systems. In *IEEE Automatic Speech Recognition and Understanding Workshop*, volume 13, 1999.
- Christoph Salge, Cornelius Glackin, and Daniel Polani. Empowerment—an introduction. In *Guided Self-Organization: Inception*, pages 67–114. Springer, 2014.
- Carol Sansone and Judith M Harackiewicz. *Intrinsic and extrinsic motivation: The search for optimal motivation and performance*. Elsevier, 2000.
- Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International Conference on Machine Learning*, pages 1312–1320, 2015.
- Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *International Conference on Learning Representations*, 2016.
- Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pages 222–227, 1991.
- Jürgen Schmidhuber. Optimal ordered problem solver. *Machine Learning*, 54(3): 211–254, 2004.
- Jürgen Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3): 230–247, 2010.
- Jürgen Schmidhuber. Powerplay: Training an increasingly general problem solver by continually searching for the simplest still unsolvable problem. *Frontiers in psychology*, 4:313, 2013.
- Jürgen Schmidhuber and Rudolf Huber. *Learning to generate focus trajectories for attentive vision*. Institut für Informatik, 1990.

- John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.
- John Schulman, Xi Chen, and Pieter Abbeel. Equivalence between policy gradients and soft q-learning. *arXiv preprint arXiv:1704.06440*, 2017a.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017b.
- Paul Hongsuck Seo, Andreas Lehrmann, Bohyung Han, and Leonid Sigal. Visual reference resolution using attention memory for visual dialog. In *Advances in Neural Information Processing Systems*, 2017.
- Archit Sharma, Shane Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised skill discovery. In *Proceeding of the International Conference on Learning Representations (ICLR), Addis Ababa, Ethiopia*, pages 26–30, 2020.
- David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.
- Bernard W Silverman. *Density estimation for statistics and data analysis*. Routledge, 2018.
- Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- Satinder P Singh, Michael J Kearns, Diane J Litman, and Marilyn A Walker. Reinforcement learning for spoken dialogue systems. In *Advances in Neural Information Processing Systems*, pages 956–962, 2000.

- Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
- Rupesh Kumar Srivastava, Bas R Steunebrink, and Jürgen Schmidhuber. First experiments with powerplay. *Neural Networks*, 41:130–136, 2013.
- Bradly C Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015.
- Susanne Still and Doina Precup. An information-theoretic approach to curiosity-driven reinforcement learning. *Theory in Biosciences*, 131(3):139–148, 2012.
- Florian Strub and Harm de Vries. Guesswhat?! models. <https://github.com/GuessWhatGame/guesswhat/>, 2017.
- Florian Strub, Harm de Vries, Jérémie Mary, Bilal Piot, Aaron C. Courville, and Olivier Pietquin. End-to-end optimization of goal-driven and visually grounded dialogue systems. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2017.
- Pei-Hao Su, Pawel Budzianowski, Stefan Ultes, Milica Gasic, and Steve Young. Sample-efficient actor-critic reinforcement learning with supervised data for dialogue management. *arXiv preprint arXiv:1707.00130*, 2017.
- Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448, 2015.
- Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv preprint arXiv:1703.05407*, 2017.

- Yi Sun, Faustino Gomez, and Jürgen Schmidhuber. Planning to be surprised: Optimal bayesian exploration in dynamic environments. In *International Conference on Artificial General Intelligence*, pages 41–51. Springer, 2011.
- Ilya Sutskever, James Martens, and Geoffrey E Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024, 2011.
- Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.
- Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.
- Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.
- Richard S Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M Pilarski, Adam White, and Doina Precup. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 761–768. International Foundation for Autonomous Agents and Multiagent Systems, 2011.
- Csaba Szepesvari, Richard S Sutton, Joseph Modayil, Shalabh Bhatnagar, et al. Universal option models. In *Advances in Neural Information Processing Systems*, pages 990–998, 2014.
- Sebastian B Thrun. Efficient exploration in reinforcement learning. 1992.

- Paul A Tipler and Gene Mosca. *Physics for scientists and engineers*. Macmillan, 2007.
- Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.
- Volker Tresp, Cristóbal Esteban, Yinchong Yang, Stephan Baier, and Denis Krompaß. Learning with memory embeddings. In *Advances in neural information processing systems (NIPS Workshop)*, 2015.
- Volker Tresp, Sahand Sharifzadeh, Dario Konopatzki, and Yunpu Ma. The tensor brain: Semantic decoding for perception and memory. *arXiv preprint arXiv:2001.11027*, 2020.
- Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1703.01161*, 2017.
- Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2015.
- Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. In *International Conference on Learning Representations (ICLR)*, 2017.
- David Warde-Farley, Tom Van de Wiele, Tejas Kulkarni, Catalin Ionescu, Steven Hansen, and Volodymyr Mnih. Unsupervised control through non-parametric discriminative rewards. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=r1eVMnA9K7>.
- Paweł Wawrzyński. Real-time reinforcement learning by sequential actor-critics and experience replay. *Neural Networks*, 2009.

- Gellért Weisz, Paweł Budzianowski, Pei-Hao Su, and Milica Gašić. Sample efficient deep reinforcement learning for dialogue systems with large action spaces. *arXiv preprint arXiv:1802.03753*, 2018.
- Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.
- Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.
- Jason D Williams and Geoffrey Zweig. End-to-end lstm-based dialog control optimized with supervised and reinforcement learning. *arXiv preprint arXiv:1606.01269*, 2016.
- Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 1992.
- Ronald J Williams and Jing Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.
- Yuxin Wu and Yuandong Tian. Training agent for first-person shooter game with actor-critic curriculum learning. 2016.
- Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015.
- Hugh D Young, Roger A Freedman, and Albert Lewis Ford. *Sears and Zemansky's university physics*, volume 1. Pearson education, 2006.
- Wojciech Zaremba and Ilya Sutskever. Learning to execute. *arXiv preprint arXiv:1410.4615*, 2014.
- Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

- Rui Zhao and Volker Tresp. Efficient dialog policy learning via positive memory retention. In *2018 IEEE Spoken Language Technology (SLT)*, pages 823–830. IEEE, 2018a.
- Rui Zhao and Volker Tresp. Energy-based hindsight experience prioritization. In *Proceedings of the 2nd Conference on Robot Learning*, pages 113–122, 2018b.
- Rui Zhao and Volker Tresp. Improving goal-oriented visual dialog agents via advanced recurrent nets with tempered policy gradient. In *International Joint Conference on Artificial Intelligence (IJCAI Workshop)*, 2018c.
- Rui Zhao and Volker Tresp. Learning goal-oriented visual dialog via tempered policy gradient. In *2018 IEEE Spoken Language Technology (SLT)*, pages 868–875. IEEE, 2018d.
- Rui Zhao and Volker Tresp. Curiosity-driven experience prioritization via density estimation. In *Conference on Neural Information Processing Systems (NeurIPS Deep RL Workshop)*, 2019.
- Rui Zhao, Haider Ali, and Patrick Van der Smagt. Two-stream rnn/cnn for action recognition in 3d videos. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4260–4267. IEEE, 2017.
- Rui Zhao, Xudong Sun, and Volker Tresp. Maximum entropy-regularized multi-goal reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning*, pages 7553–7562. PMLR, 2019.
- Rui Zhao, Volker Tresp, and Wei Xu. Mutual information-based state-control for intrinsically motivated reinforcement learning. *arXiv preprint arXiv:2002.01963*, 2020.
- Tiancheng Zhao and Maxine Eskenazi. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. *arXiv preprint arXiv:1606.02560*, 2016.

Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3357–3364. IEEE, 2017.