CONTINUOUS–DEPTH VALUE NETWORKS FOR PARAMETRIZED ACTIONS

Stefano Massaroli[*1,3], **Michael Poli**[*2,3],
Sanzhar Bakhtiyarov[2], **Jinkyoo Park**[2], **Atsushi Yamashita**[1], **Hajime Asama**[1]
[1]Department of Precision Engineering, The University of Tokyo, Tokyo, Japan
[2]Department of Industrial & Systems Engineering, KAIST, Daejeon, South Korea
[3]DiffEqML
[*]Equal contribution authors
{massaroli, yamashita, asama}@robot.t.u-tokyo.ac.jp
{poli_m, jinkyoo.park}@kaist.ac.kr,

ABSTRACT

Action spaces equipped with parameter sets are a common occurrence in reinforcement learning applications. Solutions to problems of this class have been developed under different frameworks, such as *parametrized action Markov decision processes* (PAMDP) or *hierarchical reinforcement learning* (HRL). These approaches often require extensions or modifications to standard existing algorithms developed on standard MDPs. For this reason they can be unwieldy and, particularly in the case of HRL, computationally inefficient. We propose adopting a different parametrization scheme for state–action value networks based on *neural ordinary differential equations* (NODEs) as a scalable, plug–and–play approach for parametrized action spaces. NODEs value networks do not require extensive modification to existing algorithms nor the adoption of HRL methods. Our solution can directly be integrated into existing training algorithms and opens up new opportunities in single–agent and multi–agent settings with tight precision constraints on the action parameters such as robotics.

1 INTRODUCTION

Reinforcement learning (RL) has achieved great success in an array of challenging problems spanning settings with discrete [1] and continuous [2; 3], action spaces. A different, important class of action spaces has been explored in [4] where parametrized action MDPs (PAMDPs) are introduced as an extension to MDPs fashioning a discrete set of actions equipped with sets of continuous parameters specifying their *intensity* or quantitative properties. Action spaces that can be categorized in this way occur frequently across application areas, such as simulations and games [5], navigation [6] and robotics [7]. A noteworthy advantage of parametrized actions is reducing the need for action embedding or action space reduction techniques [8]. In *multi–agent reinforcement learning* (MARL) settings, they offer a natural means to model intentional delay actions which are often required to construct optimal policies and achieve consensus [9]. It can be noted that parametrized actions have also been discussed as a relaxation of hybrid actions [10] since the latter can be obtained by augmenting the action space of the former with a set of non–parametrized actions or removing parameters from a subset of the actions. Due to the similarities between parametrized actions and hierarchies of actions, alternative approaches to solving PAMDPs involve hierarchical reinforcement learning (HRL) [11]. However, HRL methods [12] usually assume a hierarchy of arbitrary depth and are thus ill–suited for simple two–level hierarchies of actions and parameters that comprise the majority of PAMDPs.

Parametrized action spaces can be approached with policy gradient algorithms [13]. On the other hand, value–based methods for PAMDPs [10] rely on auxiliary neural networks to perform regression on optimal parameters given state or observations and are therefore suboptimal for high precision applications with robustness requirements. Additionally, such algorithms often require great engineering efforts [5] to ensure that the parameters stay within allowed interval bounds. This drawback is particularly problematic since ad–hoc techniques such as gradient squashing or flipping are often necessary to achieve satisfying performance.

With the aim of bridging this gap, we introduce a framework for solving PAMDP problems using Neural Ordinary Differential Equations (NODEs) as state–value networks and aim to provide answers to some of the following questions:

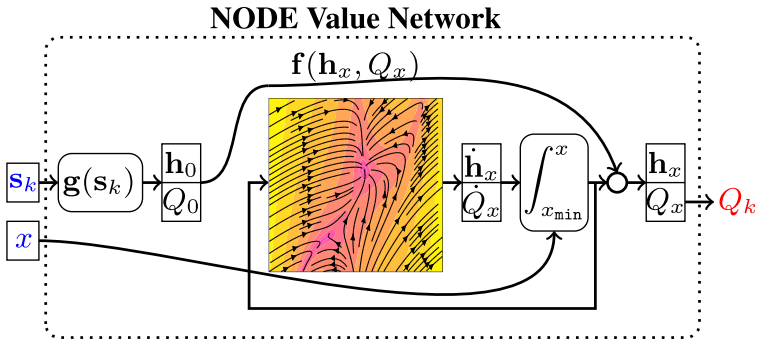

Figure 1: Schematic of NODE Q–value network as defined in (6).

- Is it possible to introduce parametrized actions into a given problem and apply out–of–the–box solutions developed for standard MDPs or *partially observed MDPs* (POMDPs)?
- How can one obtain Q–values for high resolution parameter intervals, often necessary in settings with high precision requirements such as robotics?
- Is it possible to use a single model for value networks in order to treat both continuous as well as discretized parameter intervals?

2 Related work

Previous works on parametrized actions include Q–PAMDP [4] in which selection of discrete actions and continuous parameters is learned in an alternating fashion. Recent developments [10; 14], utilize auxiliary function approximators to compute optimal parameters given state or observation vectors. In *multi–agent reinforcement learning* (MARL), mixture of actor–critic experts (MACE) has been applied to solve terrain–adaptive locomotion [6] with parametrized actions.

Parametrized actions We loosely follow the notation of [4]. We define the action space by an ordered index set $\mathcal{A}_d = \{a_1, a_2, \ldots, a_m\}, \mathcal{A}_d \subset \mathbb{N}$ specifying the actions. In this paper we assume each action $a \in \mathcal{A}$ to be equipped with only one real parameters $x \in \mathbb{R}$. The complete action space is given by $\mathcal{A} = \bigcup_{a \in \mathcal{A}} \{(a, x) \mid x \in \mathcal{X}\}$ with $\mathcal{X} \subset \mathbb{R}$ being compact sets for all $a \in \mathcal{A}_d$.

Neural ordinary differential equations Neural ordinary differential equations (NODE) [15] are a class of novel deep learning models relying on the observation that intra–layer dynamics of neural networks with skip connections and equal input and output dimensions:

$$\mathbf{h}(x+1) = \mathbf{h}_(x) + \mathbf{f}(\mathbf{h}(x), \boldsymbol{\theta}(x)), \quad x \in \mathbb{N} \tag{1}$$

resembles the explicit Euler discretization of an ordinary differential equation (ODE) described by continuous autonomous dynamics:

$$\dot{\mathbf{h}}(x) = \mathbf{f}(\mathbf{h}(x), \boldsymbol{\theta}(x)), \quad x \in \mathcal{X} \tag{2}$$

being $\mathbf{h} \in \mathbb{R}^z$, $\boldsymbol{\theta} \in \mathbb{R}^p$, $\mathbf{f} : \mathbb{R}^z \times \mathbb{R}^p \to \mathbb{R}^z$ and $\mathcal{X}$ a compact subset of $\mathbb{R}$. Given an initial condition $\mathbf{h}(0) = \mathbf{h}_0$ (i.e. the input latent vector) and under mild assumptions on f, the ODE (2) admits an unique solution defined in the whole compact $\mathcal{X}$ provided by:

$$\mathbf{h}(x) = \mathbf{h}_0 + \int_{\mathcal{X}} \mathbf{f}(\mathbf{h}(\tau), \boldsymbol{\theta}(\tau)) d\tau \tag{3}$$

In this framework the depth variable x assumes real values bringing, in the limit, the continuous map

$$\mathbf{h}_0 \mapsto \mathbf{h}(x)$$

to resemble a network with infinitely dense layers, i.e. NODEs are the deep limit of residual networks.

Although non-autonomous extensions to NODEs exist [16; 17], i.e. $\boldsymbol{\theta} = \boldsymbol{\theta}(x)$, their computational burden increases significantly. The following discussion will focus mostly on the autonomous variant.

3 NODE VALUE NETWORKS

Let $\mathbf{s}_k \in \mathbb{R}^n$ be a tensor containing the state or observation representation at step k of the unrolled episode and $(a, x) \in \mathcal{A}$ be a parametrized action. For the sake of clarity of notation, the concepts developed in the rest of this section will be referring to a single action a. Additionally, let x be bounded, $\mathcal{X} := [x_{\texttt{min}}, x_{\texttt{max}}]$.

In this paper, we propose leveraging NODEs as an effective, scalable, plug–and–play approach for parametrization of state–value maps

$$(\mathbf{s}_k, x) \mapsto Q_k$$

in settings with parametrized action spaces.

The proposed approach relies on two key components:

- Shallow embedding neural network $\mathbf{g}$ that given a state $\mathbf{s}_k$ or observation vector $\mathbf{o}_k$ at step k is tasked with computing a latent vector $\mathbf{h}_{x_{\texttt{min}}} \in \mathbb{R}^z$, in addition to state-action value $Q_{x_{\texttt{min}}} = Q(\mathbf{s}_k, x_{\texttt{min}})$:
$$\begin{bmatrix} \mathbf{h}_{x_{\texttt{min}}} \\ Q_{x_{\texttt{min}}} \end{bmatrix} = \mathbf{g}(\mathbf{s}_k) \tag{4}$$
- NODE solving for intermediate values of the embedding neural network outputs according to (3). More specifically, the NODE value network is augmented with one additional dimension describing the dynamics of Q:
$$\begin{bmatrix} \dot{\mathbf{h}}(x) \\ \dot{Q}(x) \end{bmatrix} = \mathbf{f}(\mathbf{h}(x), Q(x), \boldsymbol{\theta}). \tag{5}$$

Therefore, we define the NODE value model as

$$\begin{cases} \begin{bmatrix} \dot{\mathbf{h}}(x) \\ \dot{Q}(x) \end{bmatrix} = \mathbf{f}(\mathbf{h}(x), Q(x), \boldsymbol{\theta}) \\ \begin{bmatrix} \mathbf{h}_0 \\ Q_0 \end{bmatrix} = \mathbf{g}(\mathbf{s}_k) \\ Q_k = Q(x) \end{cases} \tag{6}$$

Thus, for each tuple $(\mathbf{s}_k, x)$, ODE (6) admits a unique solution in $[x_{\texttt{min}}, x]$. Hence, there is a mapping $\boldsymbol{\varphi} = (\boldsymbol{\varphi}_{\mathbf{h}}, \varphi_Q)$ from $\mathbb{R}^n \times \mathcal{X}$ to the space of absolutely continuous functions $\mathcal{X} \to \mathbb{R}^{z+1}$ such that $(\mathbf{h}_k, Q_k) := \boldsymbol{\varphi}(\mathbf{s}_k, x)$ satisfies (6). This, in turn, implies that the map

$$(\mathbf{s}_k, x) \mapsto \varphi_Q(\mathbf{s}_k, x)$$

satisfies

$$Q_k = \varphi_Q(\mathbf{s}_k, x) \tag{7}$$

While the training of (6) is performed integrating on the whole $\mathcal{X}$, during the inference phase the system may be solved for any arbitrary $x \in \mathcal{X}$. The result in (7) allows for a selection of optimal parameter $x = x^*$ for action a via argmax:

$$\pi(\mathbf{s}_k) := x^* = \arg\max_{x \in \mathcal{X}} Q_k(\mathbf{s}_k, x) \tag{8}$$

or any exploration–exploitation scheme. The process is repeated simultaneously for all parametrized actions at state $\mathbf{s}_k$ to obtain optimal parameters as $\pi_a(\mathbf{s}_k)$. Decision making is then carried out via argmax over the actions:

$$a^* = \arg\max_{a \in \mathcal{A}} Q_k^a(\mathbf{s}_k, \pi_a(\mathbf{s}_k)) \tag{9}$$

where Q_k^a are the Q–value corresponding to action a, respectively.

Obtaining optimal parameter values The solution of the proposed continuous model (6) in the whole parameter space $\mathcal{X}$ is given by the integration of the ODE, i.e.

$$\begin{bmatrix} \mathbf{h}_k \\ Q_k \end{bmatrix} = \int_{\mathcal{X}} \mathbf{f}(\mathbf{h}(\tau), Q(\tau), \boldsymbol{\theta}) d\tau + \mathbf{g}(\mathbf{s}_k).$$

Two different paradigms to solve (8) are hereby proposed.

Lower resolution: Once (6) is solved numerically, we obtain a sequence of values $\{Q_k\}_i$ $(i = 1, \ldots, p)$ with their corresponding parameters $\{x\}_i$, being p the number of integration steps. Therefore (8) is simply solved choosing the

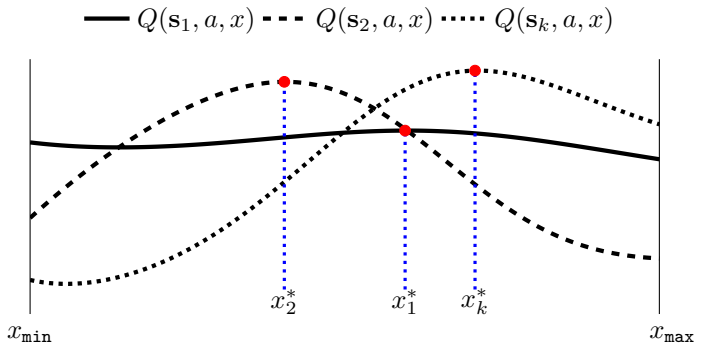

Figure 2: Evolution of $Q(x)$ as defined in (6). Neural network $\mathbf{g}$ determines the initial value of the ODE as function of state or observation vectors, launching trajectories from different points at different steps of the episode. Parameter x enters NODE $\mathbf{f}$ exclusively as an integration bound.

Algorithm 1 Projected Gradient ascent

for $t = 1 \rightarrow$ convergence **do**
  Input: $\mathbf{s}_k,\ x_t,\ 0 < \eta \ll 1$
  $\begin{bmatrix}\mathbf{h}(x_t)\\ Q(x_t)\end{bmatrix} = \int_{x_{\texttt{min}}}^{x_t} \mathbf{f}(\mathbf{h}(\tau), Q(\tau), \boldsymbol{\theta})d\tau + \mathbf{g}(\mathbf{s}_k)$
  $x_t \leftarrow x_t + \eta f_Q(\mathbf{h}(x_t), Q(x_t), \boldsymbol{\theta})$
  if $x_t \notin \mathcal{X}$ **then**
    $x_t = \Pi(x_t)$
    stop
  Output: x_t

best parameter in the finite sequence. Indeed, this naive method does not guarantee true optimality in case of low resolution in the integration phase. On the other hand, fine–grained integration can lead to high computation costs. A balance is often obtained by relaxing absolute and relative ODE integration tolerances.

Higher resolution We propose the following optimization algorithm for situations with high accuracy restrictions on the optimal parameter. Let us assume both the NODE vector field $\mathbf{f}$ and the shallow embedding network $\mathbf{g}$ to be partitioned as follows:

$$\mathbf{f} = [\mathbf{f}_{\mathbf{h}}, f_Q]^\top, \quad \mathbf{g} = [\mathbf{g}_{\mathbf{h}}, g_Q]^\top$$

The optimization problem (8) can be reformulated as

$$\begin{aligned} \underset{x\in\mathcal{X}}{\text{maximize}} \quad & \int_{x_{\texttt{min}}}^{x} f_Q(\mathbf{h}(\tau), Q(\tau), \boldsymbol{\theta})d\tau + g_Q(\mathbf{s}_k) \\ \text{subject to} \quad & \mathbf{h}(x) = \int_{x_{\texttt{min}}}^{x} \mathbf{f}_{\mathbf{h}}(\mathbf{h}(\tau), Q(\tau), \boldsymbol{\theta})d\tau + \mathbf{g}_{\mathbf{h}}(\mathbf{s}_k) \\ & x \in \mathcal{X} \end{aligned}$$

The proposed method is to perform a projected gradient ascent. First, the gradient of the cost function with respect to x is calculated via the Leibniz integral rule as,

$$\begin{aligned} \frac{\partial}{\partial x} Q_k(\mathbf{s}_k, x) &= \frac{\partial}{\partial x}\left[\int_{x_{\texttt{min}}}^{x} f_Q(\mathbf{h}(\tau), Q(\tau), \boldsymbol{\theta})d\tau + g_Q(\mathbf{s}_k)\right] \\ &= f_Q(\mathbf{h}(x), Q(x), \boldsymbol{\theta}). \end{aligned}$$

Moreover, we define a discontinuous projection function $\Pi : \mathbb{R} \rightarrow \mathcal{X}$ as

$$\Pi(x) = \begin{cases} x_{\texttt{min}} & x < x_{\texttt{min}} \\ x_{\texttt{max}} & x > x_{\texttt{max}} \end{cases}$$

The gradient ascent optimization is then developed as in Algorithm 1. Note that the optimization can be carried out in parallel for all the different actions $a \in \mathcal{A}$. As any gradient descent/ascent algorithm, convergence rates suffer

from the non–convexity/concavity of the cost function. Nevertheless, further analysis may allow to determine special structures of the NODE imposing concavity of trajectories $Q(x)$. This is left for future work.

4 DISCUSSION

Determining optimal values As is the case in standard value–based methods, parametrizing value networks with NODEs allows for the selection of optimal action values a^* via argmax operations. HRL or PAMDP approaches have historically presented additional challenges since they require extensions to the MDP framework [12] or a rederivation of theoretical convergence guarantees of standard algorithms. Reparametrizing the value network with NODEs does not break common assumptions of modern reinforcement learning and can therefore be coupled with state–of–the–art algorithms without additional overhead.

Robustness of learned representations Existing approaches for PAMDPs [10], [14] introduce auxiliary function approximators tasked with performing regression on optimal parameters given state or observation vectors. These approaches suffer from standard neural network drawbacks; they are brittle to noise or adversarial attacks [18] to their inputs and cannot increase precision of their predictions at inference time. Additionally, large amounts of engineering efforts are often expended to guarantee that the parameters lie in their allowed intervals [5]. Our approach guarantees feasibility of the optimal parameter as a direct consequence of its interval bounds being utilized as ODE integration bounds. NODEs have also been empirically shown to be more robust to adversarial attacks and noise [19].

Resolution invariance of NODE value networks A convenient advantage of the NODE framework is the possiblity of trading amount of computation for precision by taking advantage of over a century of literature on ODE solvers [20]. NODE value networks can be used when the optimal discretization of the continuous parameter range $[x_{\texttt{min}}, x_{\texttt{max}}]$ is known a-priori by solving 3 with fixed-step ODE solvers such as Runge-Kutta 4 or Euler. On the other hand, NODEs value networks excel over their discretized counterparts when no such information is available a-priori, since the usage of adaptive solvers i.e. Dormand-Prince automatically adapts the step size in $[x_{\texttt{min}}, x_{\texttt{max}}]$ such that the solution is within a set of tolerances. At inference time any value $Q(x), x \in [x_{\texttt{min}}, x_{\texttt{max}}]$ can be obtained, with no restriction on the resolution of the interval.

Computational cost Assuming non-excessive stiffness of the learned ODEs [21], NODEs tend to be more parameter efficient than their discretized neural network counterparts [15]. The computational burden of NODE value networks is therefore comparable to that of value networks relying on discrete neural networks as function approximators.

5 CONCLUSION

We proposed a novel approach to Q-value parametrization in PAMDPs that relies on neural ordinary differential equations (NODEs) as its main computation block. Our approach is advantageous in that it can be applied for a action parameters both discrete or continuous by a selecting the appropriate class of ODE solvers. Compared to existing solutions [10], [22] that utilize auxiliary network architectures to obtain optimal parameters as additional outputs, NODEs are robust due to a learned representation that is intrinsically smooth with respect to action parameters. Finally, NODEs value networks do not require an a-priori specification of the parameter interval resolution and can therefore be utilized in applications with high precision requirements. Accurate and scalable solutions for parametrized actions opens new avenues of research for multiagent settings where precise intentional delay actions are often necessary to construct an optimal policy and to achieve consensus [9], particularly in the case of stochastic environments.

REFERENCES

[1] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[2] 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.

[3] Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. In *International Conference on Machine Learning*, pages 2829–2838, 2016.

[4] Warwick Masson, Pravesh Ranchod, and George Konidaris. Reinforcement learning with parameterized actions. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[5] Matthew Hausknecht and Peter Stone. Deep reinforcement learning in parameterized action space. *arXiv preprint arXiv:1511.04143*, 2015.

[6] Xue Bin Peng, Glen Berseth, and Michiel Van de Panne. Terrain-adaptive locomotion skills using deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 35(4):81, 2016.

[7] Mark Pfeiffer, Michael Schaeuble, Juan Nieto, Roland Siegwart, and Cesar Cadena. From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots. In *2017 ieee international conference on robotics and automation (icra)*, pages 1527–1533. IEEE, 2017.

[8] Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*, 2015.

[9] Adrián Ramírez and Rifat Sipahi. Multiple intentional delays can facilitate fast consensus and noise reduction in a multiagent system. *IEEE transactions on cybernetics*, 49(4):1224–1235, 2018.

[10] Jiechao Xiong, Qing Wang, Zhuoran Yang, Peng Sun, Lei Han, Yang Zheng, Haobo Fu, Tong Zhang, Ji Liu, and Han Liu. Parametrized deep q-networks learning: Reinforcement learning with discrete-continuous hybrid action space. *arXiv preprint arXiv:1810.06394*, 2018.

[11] 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.

[12] Andrew Levy, Robert Platt, and Kate Saenko. Hierarchical actor-critic. *arXiv preprint arXiv:1712.00948*, 2017.

[13] Carson Eisenach, Haichuan Yang, Ji Liu, and Han Liu. Marginal policy gradients: A unified family of estimators for bounded action spaces with applications. *arXiv preprint arXiv:1806.05134*, 2018.

[14] Craig J Bester, Steven D James, and George D Konidaris. Multi-pass q-networks for deep reinforcement learning with parameterised action spaces. *arXiv preprint arXiv:1905.04388*, 2019.

[15] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems*, pages 6571–6583, 2018.

[16] Tianjun Zhang, Zhewei Yao, Amir Gholami, Kurt Keutzer, Joseph Gonzalez, George Biros, and Michael Mahoney. Anodev2: A coupled neural ode evolution framework. *arXiv preprint arXiv:1906.04596*, 2019.

[17] Stefano Massaroli, Michael Poli, Jinkyoo Park, Atsushi Yamashita, and Hajime Asama. Dissecting neural odes. *arXiv preprint arXiv:2002.08071*, 2020.

[18] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[19] Hanshu Yan, Jiawei Du, Vincent YF Tan, and Jiashi Feng. On robustness of neural ordinary differential equations. *arXiv preprint arXiv:1910.05513*, 2019.

[20] George D Byrne and Alan C Hindmarsh. Stiff ode solvers: A review of current and coming attractions. *Journal of Computational physics*, 70(1):1–62, 1987.

[21] Lawrence F Shampine and Charles William Gear. A user's view of solving stiff ordinary differential equations. *SIAM review*, 21(1):1–17, 1979.

[22] Haotian Fu, Hongyao Tang, Jianye Hao, Zihan Lei, Yingfeng Chen, and Changjie Fan. Deep multi-agent reinforcement learning with discrete-continuous hybrid action spaces. *arXiv preprint arXiv:1903.04959*, 2019.