Metric Residual Networks for Sample Efficient Goal-Conditioned Reinforcement Learning

Bo Liu[1], **Yihao Feng**[1], **Qiang Liu**[1], **Peter Stone**[1,2]

{bliu, yihao, lqiang, pstone}@cs.utexas.edu
[1]The University of Texas at Austin, [2]Sony AI

Abstract

Goal-conditioned reinforcement learning (GCRL) has a wide range of potential real-world applications, including manipulation and navigation problems in robotics. Especially in such robotics tasks, sample efficiency is of the utmost importance for GCRL since, by default, the agent is only rewarded when it reaches its goal. While several methods have been proposed to improve the sample efficiency of GCRL, one relatively under-studied approach is the design of neural architectures to support sample efficiency. In this work, we introduce a novel neural architecture for GCRL that achieves significantly better sample efficiency than the commonly-used monolithic network architecture. The key insight is that the *optimal* action-value function $Q^*(s, a, g)$ must satisfy the triangle inequality in a specific sense. Furthermore, we introduce the *metric residual network* (MRN) that deliberately decomposes the action-value function $Q(s, a, g)$ into the negated summation of a metric plus a residual asymmetric component. MRN provably approximates any optimal action-value function $Q^*(s, a, g)$, thus making it a fitting neural architecture for GCRL. We conduct comprehensive experiments across 12 standard benchmark environments in GCRL. The empirical results demonstrate that MRN uniformly outperforms other state-of-the-art GCRL neural architectures in terms of sample efficiency. The code is publicly available at https://github.com/Cranial-XIX/metric-residual-network.

1 Introduction

Goal-conditioned reinforcement learning (GCRL) refers to the problem in which an agent learns to solve a set of tasks indicated by different "goals" via trial and error. In contrast to the standard reinforcement learning (RL) setting, in which the per-step reward the agent receives can be an arbitrary fixed scalar function, the reward function in GCRL is usually an indicator function identifying whether the agent has achieved the goal but with a varying goal. As a result, GCRL enables the learning of a whole family of tasks with relatively little human effort toward reward design and thus has many potential real-world applications. For instance, robot manipulation tasks like picking and placing an object in a target location can be viewed as a GCRL problem where the underlying task family is parameterized by the target goal location. Similarly, robot navigation can be viewed as a GCRL problem where the goal can be any navigation destination.

Although GCRL has a straightforward reward formulation, it inherits the challenges associated with sparse reward learning: unlike in the *dense* reward setting, the reward signal is only informative when the agent reaches the goal. Therefore, sample efficiency is a major challenge in GCRL, meaning that the agent typically needs a large number of interactions with the environment to make meaningful learning progress. To address this problem, an active line of research focuses on designing novel learning algorithms that efficiently use the data. One popular approach is hindsight experience replay (HER) (Andrychowicz et al. 2017), which relabels the agent's trajectories as if they were aiming to reach the state that they in fact reached, thus rendering every trajectory an example of a successful goal achievement.

One relatively under-explored direction is the design of better neural architectures for GCRL. For actor-critic-like methods, prior work has proposed decomposing the critic function (a.k.a the action-value function $Q(s, a, g)$, see Sec. 2.1) into a bilinear network, e.g., either $Q(s, a, g) = f(s, a)^\top \phi(g)$ (Schaul et al. 2015) or $Q(s, a, g) = f(s, a)^\top \phi(s, g)$ (Hong, Yang, and Agrawal 2022), where f and g are separate neural modules. The principle behind these designs is to inject useful inductive bias into the architecture. Although empirically found to be effective, it remains unclear why such a decomposition works and whether it can be improved upon.

In this work, we argue that one fundamental inductive bias is that under the sparse reward setting, the negated optimal action-value (e.g., $-Q^*(s, a, g)$) must satisfy the *triangle inequality* in a specific sense. While there exists prior work on neural architectures that respect the triangle inequality (Pitis et al. 2020), we make the following novel contributions:

- We are the first to show that the discounted $Q^*(s, a, g)$ in the standard GCRL setting (See Sec. 2.1), when the goal is a deterministic onto mapping from the state, satisfies the triangle inequality.
- Motivated by the first point, we introduce the metric residual network (MRN) that deliberately decomposes the negative action-value function (e.g., $-Q$) into the sum of a metric and an asymmetric residual component, which provably approximates *any* quasipseudometric.[1]

[1] $(d, \mathcal{X})$ defines a quasipseudometric on $\mathcal{X}$ if 1) $\forall x \in$

- With a comprehensive experiment on 12 standard GCRL benchmark environments, MRN consistently performs better than a range of common prior designs. We hypothesize that MRN's metric component speeds up learning, and the asymmetric residual component makes the approximation accurate.

2 Preliminaries

In this section, we first introduce the formal definition of the GCRL problem. Then we review the off-policy actor-critic algorithm DDPG and the hindsight experience replay (HER) method with goal relabeling for GCRL. DDPG+HER serves as the substrate algorithm within which we compare different neural architecture designs.

2.1 Goal-conditioned Reinforcement Learning

The goal-conditioned reinforcement learning problem can be formulated as a goal-conditioned Markov Decision Process, which is an 8-tuple $M_{\text{gc}} = (\mathcal{S}, \mathcal{A}, \mathcal{G}, T, R, \gamma, \rho_0, \rho_{\mathcal{G}})$. Here, $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{G}$ are the state, action, and goal spaces of the agent. $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition dynamics. $R : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \to \mathbb{R}$ is the reward function. γ is the discount factor. ρ_0 and $\rho_{\mathcal{G}}$ are the initial state and target goal distributions. Concretely, at the start of each episode, an initial state $s_0 \sim \rho_0$ and a goal $g \sim \rho_{\mathcal{G}}$ are sampled. The agent starts at s_0, and at each time step $t \geq 0$, it chooses action from a policy π, e.g., $a_t \sim \pi(\cdot \mid s_t)$. It then receives a reward $r_{t,g} = R(s_t, a_t, g)$ and transitions to the next state $s_{t+1} \sim T(\cdot \mid s_t, a_t)$. The overall objective of the infinite-horizon GCRL problems is to maximize the following:

$$\max_{\pi} J(\pi) = \mathbb{E}_{\substack{s_0 \sim \rho_0,\, g \sim \rho_{\mathcal{G}}, \\ (s_t, a_t, r_{t,g}) \sim \pi, T, R}} \left[\sum_{t=0}^{\infty} \gamma^t r_{t,g} \mid s_0, g \right], \quad (1)$$

where the reward $r_{t,g}$ is often defined as[2]

$$r_{t,g} = R(s_t, a_t, g) = \begin{cases} 0 & M(s_t, a_t) = g \\ -1 & \text{otherwise} \end{cases}. \quad (2)$$

Here $M : \mathcal{S} \times \mathcal{A} \to \mathcal{G}$ is an onto mapping from the product space of the state and action to the goal space. Note that M_{gc} does *not* terminate when the goal is reached; the agent receives a reward every time it returns to the goal (or remains there). In practice, usually $\mathcal{G} \subset \mathcal{S}$. For example, if the goal is for an autonomous car to reach any given velocity, then $\mathcal{G}$ would naturally comprise the velocity component of the agent's full state space $\mathcal{S}$. For technical reasons in Sec. 3.1, we also include a_t in deciding whether the goal is reached.[3]

2.2 Off-Policy Actor-Critic

Actor-critic methods like Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al. 2015), Twin Delayed DDPG (TD3) (Fujimoto, Hoof, and Meger 2018) and Soft Actor-Critic (SAC) (Haarnoja et al. 2018) are popular methods for solving GCRL. In this work, we consider DDPG as the underlying RL algorithm, as it is the most commonly used method in prior GCRL research. In DDPG and under the GCRL setting, the critic $Q(s, a, g)$ evaluates the expected discounted return to reach goal g at state s by choosing a. In particular, $Q(s, a, g)$ is called the *universal value function approximator* (UVFA) as it extends the normal $Q(s, a)$ to a subset of goals indicated by g. Formally, we have

$$Q^{\pi}(s, a, g) = \mathbb{E}_{\substack{(s_t, a_t, r_{t,g}) \\ \sim \pi, T, R}} \left[\sum_{t=0}^{\infty} \gamma^t r_{t,g} \mid s_0 = s, a_0 = a, g \right]. \quad (3)$$

The critic Q is then updated by minimizing the temporal-difference (TD) error:

$$L(Q) = \mathbb{E}\left[\left(r_{t,g} + \gamma Q(s_{t+1}, \pi(s_{t+1}), g) - Q(s_t, a_t, g) \right)^2 \right], \quad (4)$$

where the expectation is taken over $(s_t, a_t, s_{t+1}, g) \sim \mathcal{D}$, and $\mathcal{D}$ is the replay buffer that stores the agent's previous experience. The actor (*a.k.a* the policy π) is then updated through the critic. In particular, the policy's gradient in DDPG is

$$\nabla_{\pi} J(\pi) = \mathbb{E}\left[\nabla_{a_t} Q(s_t, a_t, g) \big|_{a_t = \pi(s_t)} \right]. \quad (5)$$

2.3 Hindsight Experience Replay

A popular technique for mitigating the sparse reward problem is hindsight experience replay (HER) (Andrychowicz et al. 2017). Assume the agent collects a trajectory $\tau = (s_0, a_0, \ldots, s_T, a_T)$ using policy π and corresponding initial state s_0 and goal g. If $\forall t, M(s_t, a_t) \neq g$, then the agent receives -1 reward every step and therefore learns nothing from this rollout trajectory. HER relabels the trajectory as if the agent were pursuing the goal $g \in \{M(s_1, a_1), \ldots, M(s_T, a_T)\}$. In this case, the trajectory can be viewed as a successful example of reaching a different goal. By representing Q as a UVFA (e.g., $Q(s, a, g)$), one hopes that learning on these relabeled trajectories can help to generalize Q to all different goals in the goal space, including those from the original goal distribution $\rho_{\mathcal{G}}$. In practice, HER is often combined with an off-policy actor-critic algorithm. Following the original work of HER and prior work in GCRL, we use DDPG+HER as the base GCRL algorithm for policy learning and compare the performance across different neural architectures within the critic.

3 Metric Residual Network

One way to design a good neural architecture for UVFA is to base it on theoretically sound inductive biases so that the designed networks naturally inherit these inductive biases. Metric residual network (MRN) is designed based on the observation that the optimal universal action-value function $Q^*(s, a, g)$ in GCRL must satisfy the triangle inequality in a specific way. We formally prove this observation is correct in Sec. 3.1 and introduce a novel network architecture that enforces the triangle inequality in Sec. 3.2.

$\mathcal{X}$, $d(x, x) = 0$ and 2) $\forall x, y, z \in \mathcal{X}$, $d(x, y) + d(y, z) \geq d(x, z)$.

[2] There are two conventional setups: either the agent receives a reward of 1 when reaching the goal and 0 otherwise, or it receives -1 reward until it reaches the goal. In this work, without further specification, we assume the latter setting.

[3] When $r_{t,g}$ is determined by checking whether $M(s_t, a_t) = g$, as opposed to whether $M(s_{t+1} = g)$, it is a deterministic function involving no randomness from the transition dynamics T.

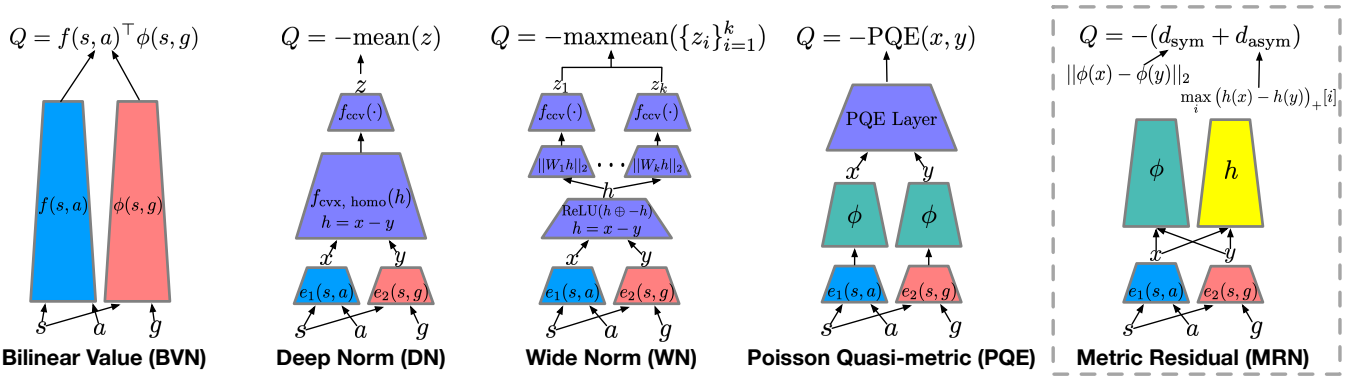

Figure 1: Comparison of different neural architecture designs for decomposing the action value function $Q(s,a,g)$ in GCRL. For the metric residual network (MRN), $d_{\text{sym}}(x,y) = \|\phi(x) - \phi(y)\|_2$ and $d_{\text{asym}}(x,y) = \max_i \big(h(x) - h(y)\big)_+[i]$. Note that x and y are passed through the *same* networks ϕ and h. In experiments, all networks are created with approximately the same number of parameters. In particular, the size of $\phi + e_1$ (or $h + e_2$) in MRN is the same as that of f (or ϕ) in BVN.

3.1 Triangle Inequality in GCRL

In this section, we start by showing that when $\mathcal{G} \equiv \mathcal{S} \times \mathcal{A}$, $Q^*(s,a,g)$ satisfies the triangle inequality. Then we extend the result to the general case when $\mathcal{G} \not\equiv \mathcal{S} \times \mathcal{A}$.

When $\mathcal{G} \equiv \mathcal{S} \times \mathcal{A}$ Under this setting, M (in Eq. (2)) becomes the identity mapping. For convenience of notation, let $\mathcal{X} = \mathcal{S} \times \mathcal{A}$, i.e. $x_t = (s_t, a_t)$. Given a policy π, the universal value function on $\mathcal{X}$ then becomes

$$Q^\pi(x, x_g) = \mathbb{E}_{\substack{(x_t, r_t) \\ \sim \pi, T, R}} \left[\sum_{t=0}^{\infty} \gamma^t r_{t,g} \mid x_0 = x, g = x_g \right]. \quad (6)$$

The optimal universal value function is therefore:

$$Q^*(x, x_g) = \max_\pi Q^\pi(x, x_g). \quad (7)$$

To build intuition regarding why Q^* satisfies the triangle inequality, consider any x^1, x^2 and x^3 in $\mathcal{X}$ (Fig. 2).

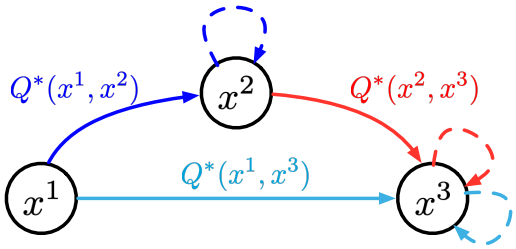

Figure 2: Intuition for triangle inequality in GCRL.

Think of $Q^*(x^1, x^2)$ as consisting of the "cost" of (discounted reward for) reaching x^2 ($\rightarrow$) plus the "cost" of staying at x^2 ($\dashrightarrow$). Decompose $Q^*(x^2, x^3)$ and $Q^*(x^1, x^3)$ similarly. Then, by definition of $Q^*(x^1, x^3)$, a proof sketch is

$$\begin{aligned} Q^*(x^1, x^3) &= (\rightarrow) + (\dashrightarrow) \geq (\rightarrow) + (\rightarrow) + (\dashrightarrow) \\ &\geq Q^*(x^1, x^2) + Q^*(x^2, x^3). \end{aligned} \quad (8)$$

This statement is formally presented in Thm. 1, and the proof is provided in Appendix A.1.

Theorem 1. *Consider the following Goal-conditioned MDP $M_{gc} = (\mathcal{S}, \mathcal{A}, \mathcal{G}, T, R, \gamma, \rho_0, \rho_g)$ as described in Sec. 2.1 except that $\mathcal{G} \equiv \mathcal{S} \times \mathcal{A}$. Then the optimal universal value function Q^* defined in Eq. (7) satisfies the triangle inequality: $\forall\, x^1, x^2, x^3 \in \mathcal{X}$,*

$$Q^*(x^1, x^2) + Q^*(x^2, x^3) \leq Q^*(x^1, x^3). \quad (9)$$

Concurrently, Wang and Isola (2022) present a claim similar to Thm. 1, where they show the optimal goal-reaching plan costs in MDPs form a quasipseudometric. The optimal goal-reaching plan cost $C(x_1, x_2)$ can be viewed as the minimum expected first hitting time from x_1 to x_2. We emphasize the two differences between Thm. 1 and their statement: **1)** the formulation in Sec. 2.1 does not assume the agent *terminates* once reaching the goal. Thus the optimal behavior will be *staying* at the goal as long as possible, if necessary, leaving the goal state and returning as quickly as possible. By contrast, the optimal behavior for the first hitting time problem is *hitting* the goal as soon as possible, even if the agent passes the goal, never to return. In practice, sometimes it is possible to change one type of problem to the other by modifying the underlying MDP, but we submit that the problem formulation we consider in Sec. 2.1 is more relevant and general, as it is the setting used in almost all common GCRL benchmarks and it can always subsume the first hitting time problem by enlarging the state space.[4] **2)** on the other hand, Wang and Isola (2022) state that the optimal goal-reaching plan costs form a quasipseudometric, while we only claim that Q^* satisfies the triangle inequality. The subtle difference is that within the formulation in Sec. 2.1, it is possible that $Q^*(x,x) \neq 0$ for some $x \in \mathcal{X}$.[5] Hence, our formulation is less restrictive.

However, as we will show next, one can still approximate Q^* well using a quasipseudometric in a larger space. For instance, assume we have a copy of x denoted $\hat{x}$ for every $x \in \mathcal{X}$. Call the space $\mathcal{Y} = \mathcal{X} \cup \hat{\mathcal{X}}$, where $\hat{\mathcal{X}} = \{\hat{x}\}$. Then

[4] One can introduce an extra sink state where the agent stays once it reaches the goal to subsume the first hitting time problem into our formulation.

[5] For instance, there are no actions that ensure the agent can stay at x forever, i.e., it needs to come out and return to x repeatedly.

consider the following function $\hat{Q} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\leq 0}$:

$$\hat{Q}^*(a,b) = \begin{cases} 0 & a = b \\ Q^*(a,a) & b = \hat{a},\ a \in \mathcal{X} \\ Q^*(a,b) & a \neq b,\ a,b \in \mathcal{X} \\ -\infty & \text{otherwise.} \end{cases} \tag{10}$$

It is easy to check that **1)** $-\hat{Q}^*$ is a quasipseudometric on $\mathcal{Y}$, **2)** $\forall x, y \in \mathcal{X}$, $\hat{Q}^*(x, e(y)) = Q^*(x,y)$, where $e(y) = y$ if $x \neq y$ and $e(y) = \hat{y}$ if $x = y$. Therefore, though Q^* only needs to satisfy triangle inequality on $\mathcal{X}$, by mapping y to $e(y)$, $\hat{Q}^*$, a negated quasipseudometric, represents Q^*.

When $\mathcal{G} \not\equiv \mathcal{S} \times \mathcal{A}$ Under this setting, M (in Eq. (2)) is an onto mapping, meaning that there might exist multiple (s,a) pairs mapping to the same goal. If the underlying transition dynamics T is *deterministic*, one observes that

$$Q^*(x,g) = \sup_{x' : M(x') = g} Q^*(x,x'). \tag{11}$$

Therefore, when the sup is attainable, denote $x_g = \arg\max_{x' : M(x') = g} Q^*(x,x')$ then $Q^*(x,g) = Q^*(x,x_g)$. The proof and more discussion are in Appendix A.3. As a result, we can view $Q^*(s,a,g)$ as a specific "distance"[6] between $x = (s,a)$ and x_g, which leads to our design of the neural architecture in the following section.

3.2 Network Design

In this section, we first present a novel construction for quasipseudometrics and prove that it universally approximates any quasipseudometric. Then we present a novel neural architecture for GCRL based on this construction and show in a toy example why such design improves generalization and therefore sample efficiency. Lastly, we provide a unified view of existing architecture designs for GCRL.

A Novel Construction for Quasipseudometrics From the observations in Eq. (10)-(11), we know $Q^*(s,a,g) = Q^*(x,x_g)$, where the latter can be represented by a negated quasipseudometric $\hat{Q}^*$. Therefore, we first present a novel construction for quasipseudometrics in Prop. 1.

Proposition 1. *Assume $x, y \in \mathcal{X}$, define*

$$d(x,y) = d_{sym}(x,y) + \underbrace{\max_{i \in [K]} \big(h_i(x) - h_i(y)\big)_+}_{d_{asym}(x,y)}, \tag{12}$$

where $(\cdot)_+ = \max(\cdot, 0)$ is the Rectified Linear Unit (ReLU) function, $h_i : \mathcal{X} \to \mathbb{R}$, and $(\mathcal{X}, d_{sym})$ is a metric. Then d ensures the three axioms of quasipseudometrics:

- *Non-negativity:* $\forall\, x, y \in \mathcal{X}, \quad d(x,y) \geq 0.$
- *Identity:* $\forall\, x \in \mathcal{X}, \quad d(x,x) = 0.$
- *Triangle inequality:*

$$\forall\, x, y, z \in \mathcal{X}, \quad d(x,z) \leq d(x,y) + d(y,z).$$

In practice, we choose $d_{sym}(x,y) = ||\phi(x) - \phi(y)||_2$ where ϕ is an arbitrary neural network. Clearly, $\forall x, y$, $d_{sym}(x,y) = d_{sym}(y,x)$, and with the ReLU function, d_{asym} is asymmetric, i.e. $\exists x, y$, $d_{asym}(x,y) \neq d_{asym}(y,x)$.

[6]Quotation marks are used to indicate that Q^* might not be a true distance function as it is not guaranteed to be symmetric.

The proof is provided in Appendix A.2. Prop. 1 ensures that any construction of d in the form of Eq. (12) is a quasipseudometric, but does not guarantee that any quasipseudometric on $\mathcal{X}$ can be represented in the form of d. The following theorem confirms that this is indeed the case.

Theorem 2 (Universal approximation of d)**.** *Given any continuous function $\nu : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, where $(\mathcal{X}, \nu)$ is a quasipseudometric and $\mathcal{X}$ is compact. Then $\forall \epsilon > 0$, with a sufficiently large K, there exists a quasipseudometric d in the form of Eq.* (12) *such that*

$$\forall x, y \in \mathcal{X}, \ \ |d(x,y) - \nu(x,y)| \leq \epsilon.$$

The proof is provided in Appendix A.4.

A Novel Architecture for GCRL Based on the novel construction of a quasipseudometric in Eq. (12), we introduce the metric residual network (MRN) for GCRL problems, which consists of two parts:

- *Projection*: with two encoders $e_1 : \mathcal{S} \times \mathcal{A} \to \mathcal{Z}$ and $e_2 : \mathcal{S} \times \mathcal{G} \to \mathcal{Z}$, we project (s,a,g) to two latent vectors:
$$h_{sa} = e_1(s,a), \quad h_{sg} = e_2(s,g).$$
- *Enforcing triangle inequality*: with d_{sym} and d_{asym} defined in Eq. (12), represent:
$$Q(s,a,g) = -\Big(d_{\text{sym}}(h_{sa}, h_{sg}) + d_{\text{aysm}}(h_{sa}, h_{sg})\Big).$$

In the projection step, one can view $e_1(s,a)$ and $e_2(s,g)$ as summarizing the sufficient statistics of x and x_g and projecting them to the same latent space $\mathcal{Z}$. In theory, x_g should be a function of s, a, g. The reason why we omit a from the inputs to e_2 is that otherwise e_2 has all the information needed to predict Q^* directly, which makes the decomposition to x and x_g meaningless and thus slows down training (See the ablation study in Sec. 5.2 for empirical evidence). After the projection step, d_{sym} and d_{asym} act as the inductive bias that constrains the learning on the latent space $\mathcal{Z}$. **The core intuition** behind decomposing d into d_{sym} and d_{asym} is that d_{sym}, due to its symmetry, *improves sample efficiency*, while d_{asym} *ensures the approximation is accurate*.

Toy Example We provide a toy example in Fig. 3 (top-left) to validate **1)** d_{sym} helps improve generalization, and **2)** d_{asym} helps improve the modeling accuracy. The environment is a square of size 1. The agent navigates freely in the white region (w/ width η), and is constrained to move in directions that do not decrease its height in the indigo region. Clearly, the shortest path length $d^*(x_0, x_g)$ is a quasipseudometric. We then approximate d^* by d_θ. To test θ's generalization ability, we consider supervised learning on 20 data points $\{(x_0, x_g), d^*(x_0, x_g)\}$, where $x_0, x_g \in [0,1]^2$ are sampled uniformly at random.[7] In the rest of Fig. 3, we plot the *generalization* error over training iteration for different thresholds η, when θ is either MRN or MRN with only d_{sym}.[8] For reference, we also plot the best generalization error of the bilinear value network (BVN) (Hong, Yang, and

[7]For simplicity, d_θ directly regresses to d^* without doing RL, hence no actions are involved.

[8]MRN with only d_{sym} is expanded to match the size of MRN.

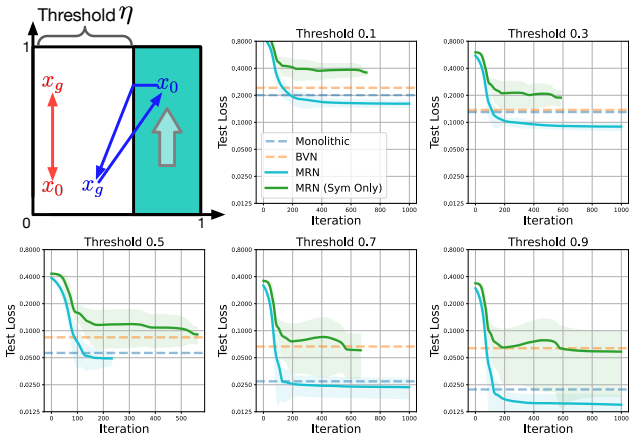

Figure 3: A toy example for illustrating the importance of both the symmetric and asymmetric parts in MRN.

Agrawal 2022) and monolithic network, as other designs including the deep/wide norms (Pitis et al. 2020) fail to work in this toy example. The generalization error is evaluated on 10000 randomly sampled (x_0, x_g) pairs different from the training data. Each curve is plotted to the iteration that reaches the lowest generalization error. From the figures, we observe that **1)** as threshold η increases, the generalization error of MRN decreases because d^* becomes more symmetric; and **2)** the symmetric part alone (green) does not approximate d^* well as d^* is asymmetric.

To summarize, the design of MRN not only incorporates the inductive bias that Q^* must satisfy the triangle inequality but also ensures it can universally approximate any Q^* with good generalization ability by having both the d_{sym} and d_{asym} parts. For the convenience of understanding and implementation of MRN, we provide the forward pass of MRN in PyTorch-like pseudocode in Alg. 1 in the Appendix.

A Unified View of Existing GCRL Architectures Recent work proposed Deep Norm (DN) and Wide Norm (WN) (Pitis et al. 2020), which also satisfy the triangle inequality by design. However, both DN and WN are norm-based networks, i.e., they are restricted to only approximate norm-induced quasi-metrics. On the other hand, the conventional monolithic multi-layer perceptron (MLP) modeling $Q(s, a, g)$, and the recently proposed bilinear value network (BVN) (Hong, Yang, and Agrawal 2022) both inherit the universal approximation guarantee from general neural networks, but they do not enforce the triangle inequality, thus making them less computationally efficient to learn. The Poisson Quasi-metric Embedding (PQE) (Wang and Isola 2022) enjoys the same theoretical guarantee as MRN. However, MRN is simpler in structure and empirically performs better than PQE (See Sec. 5.1). As such, all existing designs discussed so far can be summarized in Fig. 4.

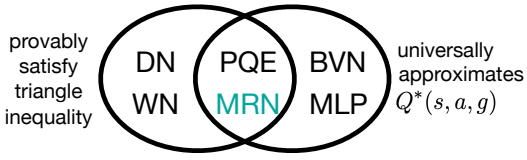

Figure 4: A Venn diagram for existing networks for GCRL.

4 Related Work

A brief summary of previous attempts to approximate the action-value function is presented in the following.

Early work considered linear functions as parametric models for value function approximation (Sutton and Barto 2018; Littman and Sutton 2001; Singh et al. 2003). Successor featured (Dayan 1993; Kulkarni et al. 2016; Borsa et al. 2018) is another line of work that assumes the reward is a linear function of the state-action features (e.g. $R(s, a) = w^\top \phi(s, a)$). Therefore, $Q(s, a) = w^\top \psi(s, a)$ where $\psi(s, a)$ is the discounted expected state-action feature occupancy. While linear functions are well-studied and relatively supportive of proving convergence properties, they can be too restrictive to approximate the value function in practical problems. Laplacian reinforcement learning (Mahadevan and Maggioni 2005) instead represents the value function based on a set of Fourier series. All of the above work discussed so far learns the action-value function in a single-task setting, without considering the more general goal-conditioned setting.

More recently, researchers have considered how to decompose the *universal value function approximator* $Q(s, a, c)$ where c can be any context variable (e.g. a goal g or a specific task k). In particular, for GCRL, Schaul et al. (2015) consider the low-rank bilinear decomposition, i.e. $Q(s, a, g) = f(s, a)^\top \phi(g)$. The motivation behind bilinear value networks is that decomposing $Q(s, a, g) = f(s, a)^\top \phi(s, g)$ may result in better learning efficiency compared to the low-rank bilinear decomposition (Hong, Yang, and Agrawal 2022). Besides the above approaches designed specifically for GCRL, Pitis et al. (2020) proposed the Deep Norm (DN) and Wide Norm (WN) families of neural networks that respect the triangle inequality. Note that norms always respect the triangle inequality, and DN is a network that essentially computes some norm between two points x and y. However, to relax one additional assumption of norms called homogeneity, i.e. $D(c \cdot (x - y)) = cD(x - y)$, DN adds a concave function on top of the convex neural network. But even with this relaxation, DN still cannot represent all functions that respect the triangle inequality as the input to DN is restricted to be $x - y$. WN is essentially any linear or maxout combination of DN networks. While DN and WN can approximate a rich family of functions, they are still restricted to norm-induced functions. Concurrently, Possion Quasi-metric Embedding (PQE) has been proposed, which approximates any quasipseudometric from a distribution perspective (Wang and Isola 2022). PQE improves upon DN/WN as it can universally approximate any quasipseudometric function. However, as explained in detail in Sec. 3.1, when applied to RL problems, PQE considers the more restrictive first hitting-time problem, while we consider the general GCRL setting. By explicitly decomposing the network into a metric part plus an asymmetric residual part, MRN can learn more efficiently. In addition, the design of DN, WN, and PQE are relatively complicated and may require careful hyperparameter tuning in practice. By contrast, MRN is much simpler and involves no hyperparameters.

5 Experimental Results

Experiments are designed to validate two hypotheses: 1) MRN achieves better sample efficiency compared to the baseline methods (Sec. 5.1), and 2) d_{sym} and d_{asym} are both important in the design of MRN (Sec. 5.2). We start by introducing the benchmarks and baseline methods and provide the experiment details. Then we provide the results and analysis indicating that they confirm our hypotheses.

Benchmarks We use the standard GCRL benchmarks (Plappert et al. 2018) including all manipulation tasks on the *Fetch* robot and *Shadow-hand* (See Fig. 5).

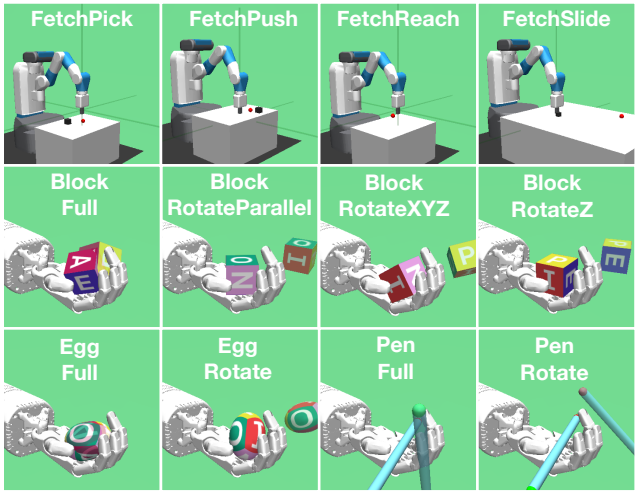

Figure 5: The 12 GCRL environments.

Baselines We compare MRN with a comprehensive list of existing UVFA architectures as illustrated in Fig. 1:

- *Monolithic*: $Q(s, a, g)$ is an unconstrained neural network that directly maps (s, a, g) to the value.
- *Bilinear Value Network* (BVN) (Hong, Yang, and Agrawal 2022): $Q(s, a, g) = f(s, a)^\top \phi(s, g)$.
- *Deep/Wide Norms* (DN/WN) (Pitis et al. 2020). DN/WN are networks that represent norm-induced metrics. The original DN/WN operates on the space where the metric is defined. To adapt them to the GCRL setting, we apply DN/WN to h_{sa} and h_{sg} with the same encoders as MRN.
- *Poisson Quasi-metric Embedding* (PQE) (Wang and Isola 2022). PQE represents quasi-metrics from a distribution perspective.

Algorithm and architecture We use deep deterministic policy gradient (DDPG) (Lillicrap et al. 2015) with hindsight experience replay (HER) (Andrychowicz et al. 2017) as the base GCRL algorithm, within which we test all different architectures. Specifically, we only change the critic architecture $Q(s, a, g)$. We constrain different networks to have the same number of parameters to equalize learning capacity. The monolithic network is a three-hidden-layer multi-layer perception (MLP) with 256 neurons per layer with ReLU activation (e.g. `[linear-relu]`×3 + `linear`). BVN has two separate networks f and ϕ, each of which is a three-layer MLP with 176 neurons per layer. For all other networks, we first define two encoders e_1 and e_2 (e.g. `[linear-relu]`×2). DN, WN, and PQE have method-specific modules on top of the two encoders, which follow the corresponding published implementation. For MRN, both the metric part d_{sym} and the residual asymmetric part d_{asym} are a single hidden layer neural network with 176 neurons (e.g., `linear-relu-linear`). The actor network is the same as the monolithic critic network except that the output layer projects to the action space.

Unified implementation on GPU To our knowledge, there has not been a consistent codebase that compares all existing neural architectures for the GCRL problem. Moreover, past implementations with HER (Andrychowicz et al. 2017) require heavy CPU computation (e.g., 19 threads per run per environment). To address both issues, we implement all existing implementations in a unified framework and train fully on GPUs, which greatly reduces the amount of training time when extensive CPU computation is not available. The implementation only requires ∼2000 mebibytes (MiB) per run per environment. Across all architectures, we use the same learning rate of 0.001 and the Adam optimizer (Kingma and Ba 2014) for updating both the actor and the critic.

Evaluation For each architecture and each environment, we evaluate with 5 independent seeds $\{100, 200, 300, 400, 500\}$. The agent is trained on 1000 episodes of data each epoch. After each training epoch, we evaluate the agent by recording its average performance (success rate) over 100 independent rollouts with randomly sampled goals. We plot the average success rate over learning epochs, averaged over the 5 seeds, along with its standard deviation (shaded region).

5.1 MRN Achieves Better Sample Efficiency

The GCRL experiment results using different state-of-the-art neural architectures as the critic is presented in Fig. 6. Note that within BVN, DN, WN, and PQE, no single method performs uniformly better than all other methods across all environments. By contrast, MRN (cyan) performs comparably to or better than all baseline methods in all environments.

5.2 On Importance of Both d_{sym} and d_{asym}

To validate the necessity of decomposing MRN into d_{sym} and d_{asym}, we conduct an ablation study that experiments with each part alone. To match the learning capacity, we enlarge the d_{sym}-only and d_{asym}-only networks to have 300 neurons per layer. The results are depicted in Fig. 7. From the figure, we can see that the d_{sym}-only network converges faster than the d_{asym}-only network. However, neither alone achieves the best sample efficiency or final performance. This result further confirms the necessity of combining both parts. To exclude the case that MRN outperforms the ablated networks due to the learning rate, we double the learning rate for the ablated methods, and the result is in Appendix C.1. Recall from Sec. 3.2 (and as illustrated in Fig. 1), that whereas in theory, x_g is a function of s, a, g, in practice, we only provide s, g as inputs for the sake of efficiency. To validate this design, we check what happens when $h_{sa} = e_1(s, a)$ but $h_{sg} = e_2(s, a, g)$. The result is in Fig. 7

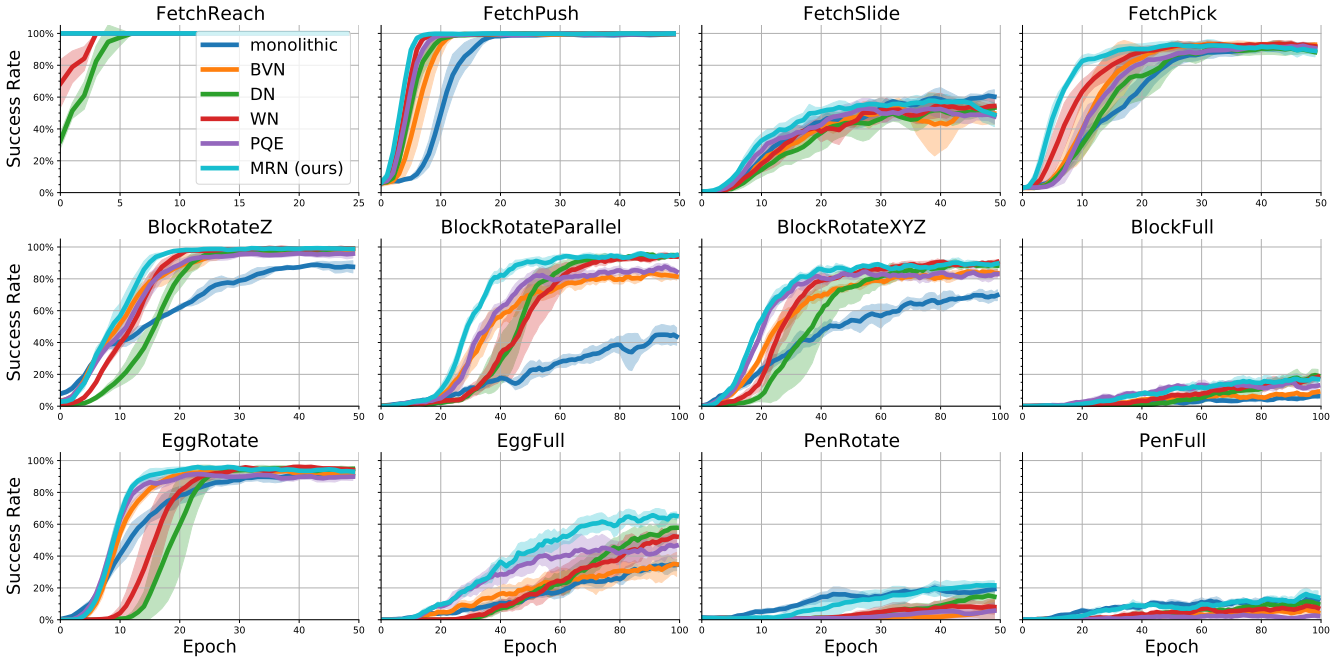

Figure 6: Success rate over training epochs for MRN (**ours**), the monolithic network, BVN (Hong, Yang, and Agrawal 2022), DN and WN (Pitis et al. 2020), and PQE (Wang and Isola 2022), on 12 GCRL environments from (Plappert et al. 2018).

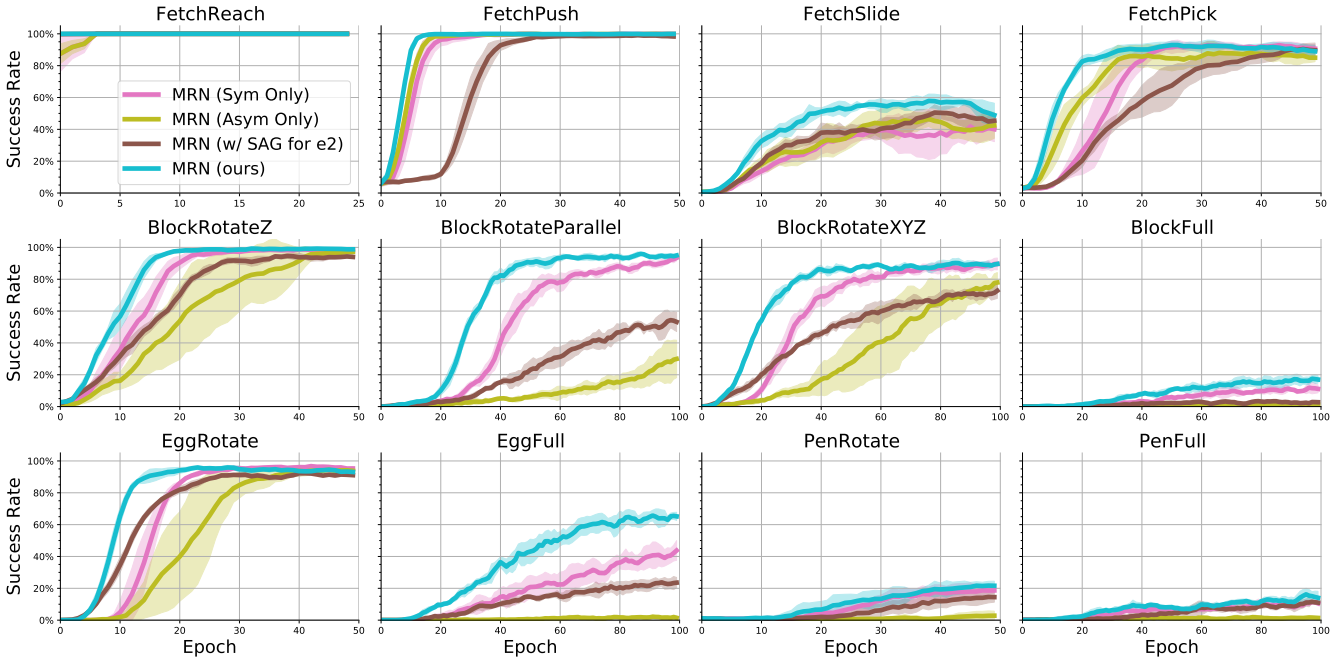

Figure 7: Ablation study on individual symmetric/asymmetric parts of MRN, and on feeding all (s, a, g) to e_2.

(w/ SAG for e2). We observe that feeding all s, a, g as input to e_2 confuses the network and prevents it from distinguishing between the roles of e_1 and e_2.

6 Conclusion

This paper introduces the metric residual network for GCRL problems that has clear theoretical motivations and is straightforward to implement in practice. Comprehensive studies on standard GCRL benchmarks demonstrate that MRN outperforms all existing designs of neural architectures in terms of sample efficiency. As large models pretrained on a large set of environments become increasingly popular, we expect that investigating useful inductive biases for saving computation will become increasingly interesting and important. An interesting future research direction is to explore effective architectures for more general RL problems where the reward function can take an arbitrary form.

7 Acknowledgement

The research was conducted in both the statistical learning and AI group (SLAI) led by Qiang Liu and the Learning Agents Research Group (LARG) led by Peter Stone in computer science at UT Austin. SLAI research is supported in part by CAREER-1846421, SenSE-2037267, EAGER-2041327, and Office of Navy Research, and NSF AI Institute for Foundations of Machine Learning (IFML). LARG research is supported in part by NSF (CPS-1739964, IIS-1724157, FAIN-2019844), ONR (N00014-18-2243), ARO (W911NF-19-2-0333), DARPA, GM, Bosch, and UT Austin's Good Systems grand challenge. Peter Stone serves as the Executive Director of Sony AI America and receives financial compensation for this work. The terms of this arrangement have been reviewed and approved by the University of Texas at Austin in accordance with its policy on objectivity in research.

References

Andrychowicz, M.; Wolski, F.; Ray, A.; Schneider, J.; Fong, R.; Welinder, P.; McGrew, B.; Tobin, J.; Pieter Abbeel, O.; and Zaremba, W. 2017. Hindsight experience replay. *Advances in neural information processing systems*, 30.

Borsa, D.; Barreto, A.; Quan, J.; Mankowitz, D.; Munos, R.; Van Hasselt, H.; Silver, D.; and Schaul, T. 2018. Universal successor features approximators. *arXiv preprint arXiv:1812.07626*.

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

Fujimoto, S.; Hoof, H.; and Meger, D. 2018. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, 1587–1596. PMLR.

Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; et al. 2018. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*.

Hong, Z.-W.; Yang, G.; and Agrawal, P. 2022. Bilinear value networks. *arXiv preprint arXiv:2204.13695*.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kulkarni, T. D.; Saeedi, A.; Gautam, S.; and Gershman, S. J. 2016. Deep successor reinforcement learning. *arXiv preprint arXiv:1606.02396*.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Littman, M.; and Sutton, R. S. 2001. Predictive representations of state. *Advances in neural information processing systems*, 14.

Mahadevan, S.; and Maggioni, M. 2005. Value function approximation with diffusion wavelets and laplacian eigenfunctions. *Advances in neural information processing systems*, 18.

Pitis, S.; Chan, H.; Jamali, K.; and Ba, J. 2020. An inductive bias for distances: Neural nets that respect the triangle inequality. *arXiv preprint arXiv:2002.05825*.

Plappert, M.; Andrychowicz, M.; Ray, A.; McGrew, B.; Baker, B.; Powell, G.; Schneider, J.; Tobin, J.; Chociej, M.; Welinder, P.; et al. 2018. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*.

Schaul, T.; Horgan, D.; Gregor, K.; and Silver, D. 2015. Universal value function approximators. In *International conference on machine learning*, 1312–1320. PMLR.

Singh, S. P.; Littman, M. L.; Jong, N. K.; Pardoe, D.; and Stone, P. 2003. Learning predictive state representations. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 712–719.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Wang, T.; and Isola, P. 2022. On the Learning and Learnablity of Quasimetrics. *arXiv preprint arXiv:2206.15478*.

A Theory

A.1 Proof of Thm. 2

Theorem 1. *Consider the following goal-conditioned MDP $M_{gc} = (\mathcal{S}, \mathcal{A}, \mathcal{G}, T, R, \gamma, \rho_0, \rho_g)$ as described in Sec. 2.1 except that $\mathcal{G} \equiv \mathcal{S} \times \mathcal{A}$. Then the optimal universal value function Q^* defined in Eq. (7) satisfies the triangle inequality: $\forall\, x^1, x^2, x^3 \in \mathcal{X}$,*

$$Q^*(x^1, x^2) + Q^*(x^2, x^3) \leq Q^*(x^1, x^3). \tag{13}$$

Proof. Assume the statement does not hold. As we fix the goals to be x^2 or x^3, the GCRL problem is transformed into two standard MDPs with fixed goals. Therefore, there exists an optimal Markov policy for each problem. Specifically, denote optimal Markov policies corresponding to $Q^*(x^1, x^2)$, $Q^*(x^2, x^3)$ and $Q^*(x^1, x^3)$ as π_1, π_2 and π_3. Then consider the following policy $\pi_{1\to 2}$ for $t > 0$:

$$\pi_{1\to 2}(a \mid s_t) = \begin{cases} \pi_1(a \mid s_t) & x^2 \notin x_{<t}, \\ \pi_2(a \mid s_t) & \text{otherwise.} \end{cases} \tag{14}$$

Here, $x_{<t} = \{x_0, x_1, \ldots, x_{t-1}\}$ denotes all past state and action pairs before time t. Let τ be the random variable that indicates the first time $\pi_{1\to 2}$ reaches x^2. Then we define

$$q^1_{1\to 2} = \mathbb{E}_{\substack{(x_t, r_t) \\ \sim \pi_{1\to 2}, T, R, \tau}} \left[\sum_{t=0}^{\tau} \gamma^t r_{t,g} \;\middle|\; x_0 = x^1, g = x^2 \right],$$

and

$$q^2_{2\to 3} = \mathbb{E}_{\substack{(x_t, r_t) \\ \sim \pi_{1\to 2}, T, R, \tau}} \left[\sum_{t=\tau}^{\infty} \gamma^t r_{t,g} \;\middle|\; x_\tau = x^2, g = x^3 \right].$$

Then, clearly

$$\begin{aligned} Q^{\pi_{1\to 2}}(x^1, x^3) &= q^1_{1\to 2} + q^2_{1\to 2} \\ &\geq Q^*(x^1, x^2) + Q^*(x^2, x^3) > Q^*(x^1, x^3). \end{aligned} \tag{15}$$

The first equality follows the definition of $Q^{\pi_{1\to 2}}$. To see why the first inequality holds, note that

$$\begin{aligned} & Q^*(x^1, x^2) - q^1_{1\to 2} \\ =& \mathbb{E}_{\substack{(x_t, r_t) \\ \sim \pi_{1\to 2}, T, R, \tau}} \left[\sum_{t=\tau}^{\infty} \gamma^t r_{t,g} \;\middle|\; x_\tau = x^2, g = x^3 \right] \\ \leq& 0 \quad (\text{since } r_{t,g} \leq 0). \end{aligned}$$

Similarly, we have

$$Q^*(x^2, x^3) - q^2_{1\to 2} \leq \mathbb{E}_\tau \left[\gamma^\tau Q^*(x^2, x^3) \right] - q^2_{1\to 2} \leq 0.$$

Combining above we can tell that the first inequality holds. The last inequality in Eq. (15) holds because of our assumption. Therefore, we have a policy $\pi_{1\to 2}$ (though possibly non-Markov) that achieves a higher expected discounted return with respect to reaching x^3 than π^3, which contradicts the assumption that π^3 is optimal. □

A.2 Proof of Prop. 1

Proposition 1. *Assume $x, y \in \mathcal{X}$, define*

$$d(x, y) = d_{sym}(x, y) + \underbrace{\max_{i \in [K]} \big(h_i(x) - h_i(y) \big)_+}_{d_{asym}(x, y)}, \tag{16}$$

where $(\cdot)_+ = \max(\cdot, 0)$ is the Rectified Linear Unit (ReLU) function, $h_i : \mathcal{X} \to \mathbb{R}$ and $(\mathcal{X}, d_{sym})$ is a metric. Then d ensures the three axioms of quasipseudometrics:

- *Non-negativity:*

$$\forall\, x, y \in \mathcal{X}, \quad d'(x, y) \geq 0.$$

- *Identity:*

$$\forall\, x \in \mathcal{X}, \quad d'(x, x) = 0.$$

- *Triangle inequality:*

$$\forall\, x, y, z \in \mathcal{X}, \quad d'(x, z) \leq d'(x, y) + d'(y, z).$$

Proof. The non-negativity is obvious because d is a metric and we use the ReLU function. The identity property can be easily checked. For the triangle inequality,

$$\begin{aligned} d'(x,y) + d'(y,z) &= d(x,y) + d(y,z) \\ &\quad + \max_i \big(h_i(x) - h_i(y)\big)_+ + \max_i \big(h_i(y) - h_i(z)\big)_+ \\ &\geq d(x,z) + \max_i \big(h_i(x) - h_i(y) + h_i(y) - h_i(z)\big)_+ \\ &= d'(x,z). \end{aligned}$$

□

A.3 Proof of $Q^*(x,g) = \sup_{x':M(x')=g} Q^*(x,x')$ in Eq. (11)

Assume the $\sup$ is attainable. Fixing a goal g, the GCRL problem becomes the standard MDP and there exists a deterministic optimal policy π^* for reaching g from x. Since we assume the dynamics is also deterministic, then it means the optimal trajectory to reach g from x is also deterministic. Denote this trajectory as $\xi = (s_0, a_0, \dots)$ where $(s_0, a_0) = x$. Also denote

$$x_g = \arg\max_{x':M(x')=g} Q^*(x,x') \tag{17}$$

Then we argue that $Q^*(x,g) = Q^*(x,x_g)$. Assume otherwise, then either $Q^*(x,x_g) > Q^*(x,g)$ or $Q^*(x,x_g) < Q^*(x,g)$.

- $Q^*(x,x_g) > Q^*(x,g)$:
  this is impossible because otherwise we can replace π^* with the policy corresponding to $Q^*(x,x_g)$ to reach a higher return, contradicting the definition of π^*.
- $Q^*(x,x_g) < Q^*(x,g)$:
  Denote
  $$\tau = \min_t(M(x_t) = g), \quad x_t \in \xi.$$
  In other words, x_τ is the first (s,a) pair along ξ that reaches the goal. Then there are two cases:
  - After π^* reaches x_τ, it will come back to x_τ again (hence repeatedly). In this case, clearly
    $$Q^*(x,x_g) \geq Q^*(x,x_\tau) \geq Q^*(x,g),$$
    contradicting our assumption that $Q^*(x,g) > Q^*(x,x_g)$. The first inequality follows the definition of x_g. The second inequality follows the definition of $Q^*(x,x_\tau)$.
  - After π^* reaches x_τ, it will *never* come back to x_τ again. In this case, one can find the next $\tau' = \min_{t>\tau}(M(x_t) = g),\ \ x_t \in \xi$. Then similarly there are two cases, if π^* repeatedly visits $x_{\tau'}$, then we follow the argument in the above case. Otherwise we can recursively find the next τ'', so on and so forth. Eventually, either we end up with the last state x_ζ such that no $t > \zeta$ satisfies $M(x_t) = g$, from which then we can see that
    $$Q^*(x,x_g) \geq Q^*(x,x_\zeta) \geq Q^*(x,g).$$
    The last inequality follows that ever since we visit x_ζ, all future rewards will be -1. Or the other case is that there exists an infinite length sequence of such $\{x_\tau\}$. Then following this sequence, the original statement is still true though the $\sup$ is no longer attainable. But in this case, one can find a x_τ in the sequence such that $Q^*(x,x_\tau)$ is arbitrarily close to $Q^*(x,g)$.

A.4 Proof of Thm. 2

Theorem 1 (Universal approximation of d)**.** *Given any continuous function $\nu : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, where $(\mathcal{X}, \nu)$ is a quasipseudometric and $\mathcal{X}$ is compact. Then $\forall \epsilon > 0$, with a sufficiently large K, there exists a quasipseudometric d in the form of Eq.* (12) *such that*

$$\forall x, y \in \mathcal{X}, \ \ |d(x,y) - \nu(x,y)| \leq \epsilon.$$

Proof. First of all, from Prop. 1, any d in the form of Eq. (12) is a quasipseudometric. Next we show that any quasipseudometric ν can be approximated by a d in the form of Eq. (12). Since ν is a quasipseudometric, we have

$$\forall\, x,y,z \in \mathcal{X}, \quad \nu(x,y) \geq \nu(x,z) - \nu(y,z) \quad \Longrightarrow \quad \nu(x,y) = \sup_{z' \in \mathcal{X}} \left(\nu(x,z') - \nu(y,z')\right)_+. \tag{18}$$

As $\nu(x_1,x_2)$ is continuous in both x_1 and x_2, for a fixed pair of (x,y), define

$$f(z) = \left(\nu(x,z) - \nu(y,z)\right)_+. \tag{19}$$

Then f is clearly continuous in z (continuous functions are closed under subtraction and max operation). By definition of continuity,

$$\forall\, \epsilon > 0,\ \ \exists\, \delta > 0, \quad d'(z_1, z_2) \le \delta \Longrightarrow |f(z_1) - f(z_2)| \le \epsilon, \tag{20}$$

where d' is a metric on $\mathcal{X}$. Since $\mathcal{X}$ is compact, for a sufficient large K, we can cover the domain $\mathcal{X}$ with K δ-radius balls centered at $\{\xi_i\}_{i=1}^K$, i.e.

$$\mathcal{X} \subseteq \bigcup_{i=1}^{K} B_i, \ \text{ where } \ B_i = \{x \in \mathcal{X} \mid d'(\xi_i, x) \le \delta\}.$$

Now, consider the function d, where for any $x, y \in \mathcal{X}$,

$$d(x, y) = \max_i \Big(\nu(x, \xi_i) - \nu(y, \xi_i)\Big)_+. \tag{21}$$

Denote $z^*(x, y) = \arg\max_{z'}(v(x, z') - v(y, z'))_+$, then we have $\forall x, y \in \mathcal{X}$,

$$\begin{aligned} |d(x, y) - \nu(x, y)| &= \left| \max_i \Big(\nu(x, \xi_i) - \nu(y, \xi_i)\Big)_+ - \Big(\nu(x, z^*(x, y)) - \nu(y, z^*(x, y))\Big)_+ \right| \\ &= \left| \max_i \Big(f(\xi_i) - f(z^*(x, y))\Big) \right| \\ &\le \max_i \Big| f(\xi_i) - f(z^*(x, y)) \Big| \\ &\le \epsilon \qquad \text{// } \exists\, \xi_i, d'(\xi_i, z^*(x, y)) \le \delta, \text{ then apply Eq. (20)} \end{aligned} \tag{22}$$

Clearly the d defined above is in the form of Eq. (12) (we can set $d_{\text{sym}}(x, y) = 0, \forall x, y \in \mathcal{X}$). □

B Algorithm

The forward pass of MRN in PyTorch-like pseudocode is provided in Alg. 1. Here, e_1, e_2, d_{sym} and d_{aym} are standard multi-layer perceptrons (MLPs).

Algorithm 1: Computation of $\texttt{Q}(s, a, g)$ with MRN

```
# The forward pass in PyTorch code
def forward(self, state, action, goal):
    # state  (batch, dim_state)
    # action (batch, dim_action)
    # goal   (batch, dim_goal)

    # 1. Projection
     sa = torch.cat([state, action], -1)
     sg = torch.cat([state, goal], -1)
     x  = self.e_1(sa)
     y  = self.e_2(sg)

    # 2. Enforcing triangle inequality
     d_sym  = self.sym(x) - self.sym(y)
     d_sym  = d_sym.pow(2).mean(-1).sqrt()
     d_asym = self.asym(x) - self.asym(y)
     d_asym = F.relu(d_asym).max(-1)[0]
     Q_sag  = -(d_sym + d_asym).view(-1,1)
     return Q_sag
```

C Experiment

C.1 Ablation study with doubled learning rate

We conduct the same ablation as in Sec. 5.1 except that we double the learning rate to 0.002. Results are shown in Fig. 8. From the figure, we can see that the claim in the main experiment section holds true.

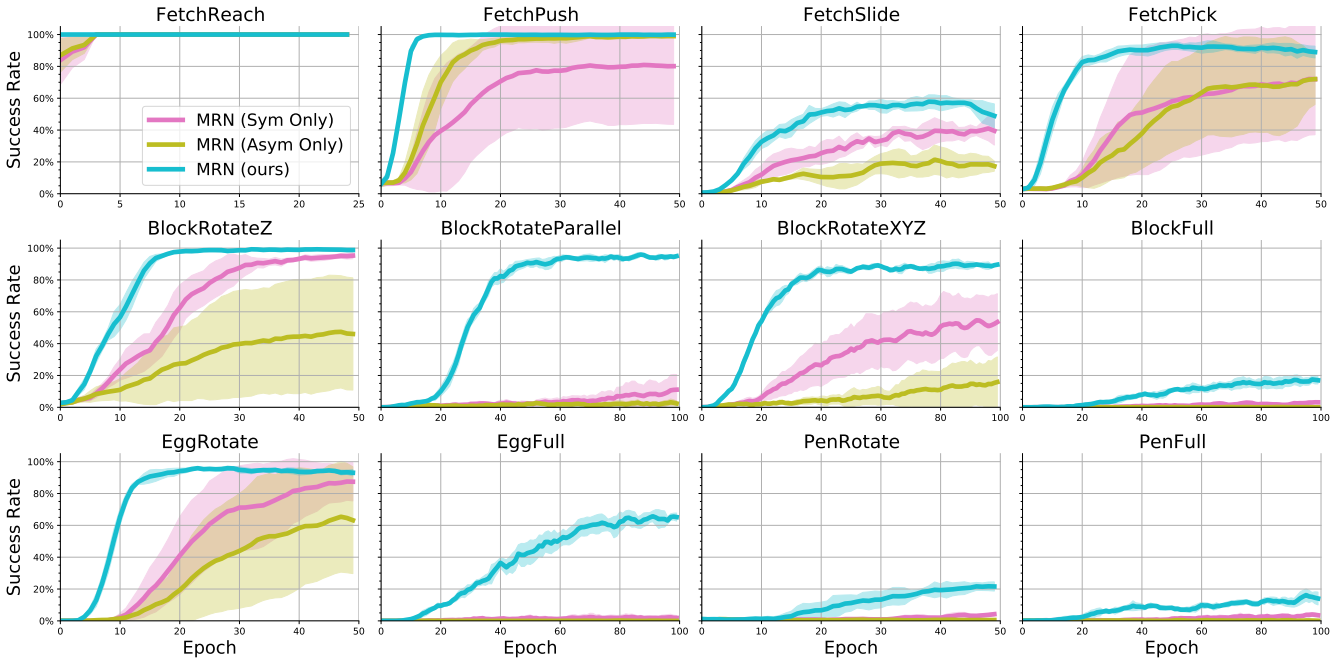

Figure 8: Ablation study on individual symmetric/asymmetric parts of MRN with the doubled learning rate (0.002).