KOOPMAN Q-LEARNING: OFFLINE REINFORCEMENT LEARNING VIA SYMMETRIES OF DYNAMICS

Anonymous authors
Paper under double-blind review

ABSTRACT

Offline reinforcement learning leverages large datasets to train policies without interactions with the environment. The learned policies may then be deployed in real-world settings where interactions are costly or dangerous. Current algorithms over-fit to the training dataset and as a consequence perform poorly when deployed to out-of-distribution generalizations of the environment. We aim to address these limitations by learning a Koopman latent representation which allows us to infer symmetries of the system's underlying dynamic. The latter is then utilized to extend the otherwise static offline dataset during training; this constitutes a novel data augmentation framework which reflects the system's dynamic and is thus to be interpreted as an exploration of the environments phase space. To obtain the symmetries we employ Koopman theory in which nonlinear dynamics are represented in terms of a linear operator acting on the space of measurement functions of the system and thus symmetries of the dynamics may be inferred directly. We provide novel theoretical results on the existence and nature of symmetries relevant for control systems such as reinforcement learning settings. Moreover, we empirically evaluate our method on several benchmark offline reinforcement learning tasks and datasets including D4RL, Metaworld and Robosuite and find that by using ourframework we consistently improve the state-of-the-art for Q-learning methods.

1 INTRODUCTION

The recent impressive advances in reinforcement learning (RL) range from robotics, to strategy games and recommendation systems (Kalashnikov et al., 2018; Li et al., 2010). Reinforcement learning is canonically regarded as an active learning process - also referred to as online RL - where the agent interacts with the environment at each training run. In contrast, offline RL algorithms learn from large, previously collected static datasets, and thus do not rely on environment interactions (Agarwal et al., 2020a; Ernst et al., 2005; Fujimoto et al., 2019). Online data collection is performed by simulations or by means of real world interactions e.g. robotics and in either scenario interactions maybe costly and/or dangerous.

In principle offline datasets only need to be collected once which alleviates the before-mentioned short-comings of costly online interactions. Offline datasets are typically collected using behavioral policies for the specific task ranging from, random policies, or near-optimal policies to human demonstrations. In particular, being able to leverage the latter is a major advantage of offline RL over online approaches, and then the learned policies can be deployed or finetuned on the desired environment. Offline RL has successfully been applied to learn agents that outperform the behavioral policy used to collect the data (Kumar et al., 2020; Wu et al., 2019; Agarwal et al., 2020b; Ernst et al., 2005). However algorithms admit major shortcomings in regard to over-fitting and overestimating the true state-action values of the distribution. One solution was recently propsed by Sinha et al. (2021), where they tested several data augmentation schemes to improve the performance and generalization capabilities of the learned policies.

However, despite the recent progress, learning from offline demonstrations is a tedious endeavour as the dataset typically does not cover the full state-action space. Moreover, offline RL algorithms per definition do not admit the possibility for further environment exploration to refine their distributions towards an optimal policy. It was argued previously that is basically impossible for an offline RL agent to learn an optimal policy as the generalization to near data generically leads to compounding

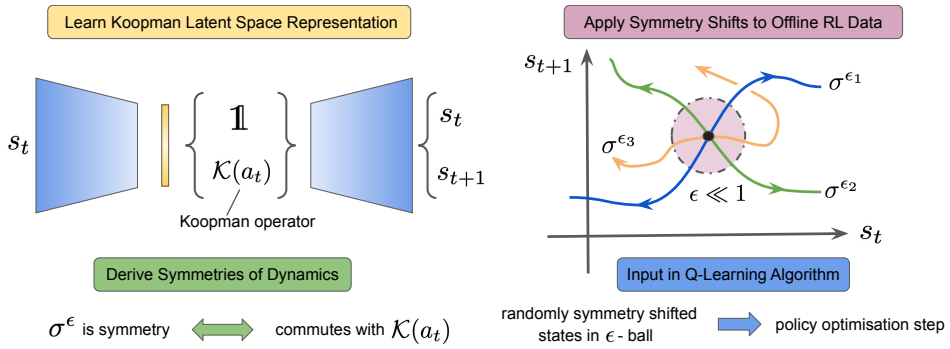


Figure 1: Overview of Koopman Q-learning. The states of the data point (s_t, s_{t+1}, a_t) are shifted along symmetry trajectories parametrized by $\epsilon_{1,2,3}$ for constant a_t. Symmetry transformations can be combined to reach other specific subset of the ϵ-ball region.

errors such as overestimation bias (Kumar et al., 2020). In this paper, we look at offline RL through the lens of Koopman spectral theory in which nonlinear dynamics are represented in terms of a linear operator acting on the space of measurement functions of the system. Through which the representation the symmetries of the dynamics may be inferred directly, and can then be used to guide data augmentation strategies see Figure 1. We further provide theoretical results on the existence on nature of symmetries relevant for control systems such as reinforcement learning. More specifically, we apply Koopman spectral theory by: first learning symmetries of the system's underlying dynamic in a self-supervised fashion from the static dataset, and second employing the latter to extend the offline dataset at training time by out-of-distribution values. As this reflects the system's dynamics the additional data is to be interpreted as an exploration of the environment's phase space.

Some prior works have explored symmetry of the state-action space in the context of Markov Decision Processes (MDP's) (Higgins et al., 2018; Balaraman & Andrew, 2004; van der Pol et al., 2020) since many control tasks exhibit apparent symmetries e.g. the classic cart-pole task which is symmetric across y-axis. However, the paradigm we introduce in this work is of a different nature entirely. The distinction is twofold: first, the symmetries are learned in a self-supervised way and are in general not apparent to the developer; second: we concern with symmetry transformation from state tuples $(s_t, s_{t+1}) \rightarrow (\tilde{s}_t, \tilde{s}_{t+1})$ which leave the action invariant inferred from the dynamics inherited by the behavioral policy of the underlying offline data. In other words we seek to derive a neighbourhood around a MDP tuple in the offline dataset in which the behavioral policy is likely to choose the same action based on its dynamics in the environment. In practice the Koopman latent space representation is learned in a self-supervised manner by training to predict the next state using a VAE model (Kingma & Welling, 2013).

To summarize, in this paper, we propose Koopman Forward (Conservative) Q-learning (KFC): a model-free Q-learning algorithm which uses the symmetries in the dynamics of the environment to guide data augmentation strategies. We also provide thorough theoretical justifications for KFC. Finally, we empirically test our approach on several challenging benchmark datasets from D4RL (Fu et al., 2021), MetaWorld (Yu et al., 2019) and Robosuite (Zhu et al., 2020) and find that by using KFC we can improve the state-of-the-art on most benchmark offline reinforcement learning tasks.

2 Preliminaries and Background

2.1 Offline RL & Conservative Q-learning

Reinforcement learning algorithms train policies to maximize the cumulative reward received by an agent who interacts with an environment. Formally the setting is given by a Markov decision process $(\mathcal{S}, \mathcal{A}, \rho, r, \gamma)$, with state space $\mathcal{S}$, action space $\mathcal{A}$, and $\rho(s_{t+1}|s_t, a_t)$ the transition density function from the current state and action to the next state. Moreover, γ is the discount factor and $r(S_t)$ the reward function. At any discrete time the agent chooses an action $a_t \in \mathcal{A}$ according to its underlying policy $\pi_\theta(a_t|s_t)$ based on the information of the current state $s_t \in \mathcal{S}$ where the policy is parametrized by θ. We focus on the Actor-Critic methods for continuous control tasks in the following. In deep

RL the parameters θ are the weights in a deep neural network function approximation of the policy or Actor as well as the state-action value function Q or Critic, respectively, and are optimized by gradient decent. The agent i.e. the Actor-Critic is trained to maximize the expected γ-discounted cumulative reward $\mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t \, r_\pi(s_t, a_t)]$, with respect to the policy network i.e. its parameters θ. For notational simplicity we omit the explicit dependency of the latter in the remainder of this work. Furthermore the state-action value function $Q(s_t, a_t)$, returns the value of performing a given action a_t while being in the state s_t. The Q-function is trained by minimizing the so called Bellman error as

$$\hat{Q}_{i+1} \leftarrow \arg\min_{Q} \mathbb{E}\big[r_t + \gamma \, \hat{Q}_i(s_{t+1}, a_{t+1}) - Q_i(s_t, a_t)\big] \quad , \tag{1}$$

This is commonly referred to as the ith policy evaluation step where the hat denotes the target Q-function. In offline RL one aims to learn an optimal policy for the given the dataset $\mathcal{D} = \bigcup_{k=1,\ldots,\text{\#data-points}} \big(s_t, a_t, r_t, s_{t+1}\big)_k$ as the option for exploration of the MDP is not available. The policy is optimized to maximize the state-action value function via the policy improvement

$$\pi_{i+1} \leftarrow \arg\max_{\pi} \mathbb{E}_{s_t \sim \mathcal{D}} \big[\hat{Q}\big(s_t, \pi_i(a_t|s_t)\big)\big] \quad . \tag{2}$$

Note that behavioural policies including sub-optimal or randomized ones may be used to generate the static dataset $\mathcal{D}$. In that case offline RL algorithms face difficulties in the learning process.

CQL algorithm: CQL is built on top of a Soft-Actor Critic algorithm (SAC) (Haarnoja et al., 2018), which employs soft-policy iteration of a stochastic policy (Haarnoja et al., 2017). A policy entropy regularization term is added to the policy improvement step in Eq. (2) as

$$\pi_{i+1} \leftarrow \arg\max_{\pi} \mathbb{E}_{s_t \sim \mathcal{D}} \Big[\hat{Q}\big(s_t, \pi_i(a_t|s_t)\big) - \alpha \log \pi_i(a_t|s_t)\Big] \quad . \tag{3}$$

where α either is a fixed hyperparameter or may be chosen to be trainable. CQL reduces the overestimation of state-values - in particular those out-of distribution from $\mathcal{D}$. It achieves this by regularizing the Q-function in Eq. (1) by a term minimizing its values over out of distribution randomly sampled actions as

$$\begin{aligned} \hat{Q}_{i+1} \;\leftarrow\; \arg\min_{Q} \Bigg(& \mathop{\mathbb{E}}_{s_t, a_t, s_{t+1} \sim \mathcal{D}} \Big[r_t + \gamma \, \hat{Q}_i(s_{t+1}, a_{t+1}) - Q_i(s_t, a_t)\Big] \\ & + \tilde{\alpha} \mathop{\mathbb{E}}_{s_t \sim \mathcal{D}} \Big[\log \sum_a \exp\big(Q_i(s_t, a)\big) - \mathop{\mathbb{E}}_{a \sim \pi(s_t)} \big[Q_i(s_t, a)\big]\Big]\Bigg) \end{aligned} \tag{4}$$

where a_{t+1} is given by the prediction of the policy $\pi_i(a_{t+1}|s_{t+1})$ and $\tilde{\alpha}$ is a hyperparameter balancing the regulizer term.

2.2 Koopman theory

Historically, the Koopman theoretic perspective of dynamical systems was introduced to describe the evolution of measurements of Hamiltonian systems (Koopman, 1931; Mezić, 2005). The underlying dynamic of most modern reinforcement learning tasks is of nonlinear nature, i.e. the agents actions lead to changes of it state described by a complex non-linear dynamical system. In contrast to linear systems which are completely characterized by their spectral decomposition non-linear systems lack such a unified characterisation. The Koopman operator theoretic framework describes nonlinear dynamics via a linear infinite-dimensional Koopman operator and thus inherits certain tools applicable to linear control systems (Mauroy et al., 2020; Kaiser et al., 2021). In practice one aims to find a finite-dimensional representation of the Koopman operator which is equivalent to obtaining a coordinate transformations in which the nonlinear dynamics are approximately linear. A general non-affine control system is governed by the system of non-linear ordinary differential equations (ODE's) as

$$\dot{s} = f(s, a) \tag{5}$$

where s, is the n-dimensional state vector, a the m-dimensional action vector with $(s, a) \in \mathcal{S} \times \mathcal{A}$ the state-action-space. Moreover, $\dot{s} = \partial_t s$ is the time derivative, and f is some general non-linear - at least $\mathcal{C}^1$-differentiable - vector valued function. For a discrete time system, Eq. (5) takes the form

$$s_{t+1} = F(s_t, a_t) \tag{6}$$

where s_t denotes the state at time t where F is at least $\mathcal{C}^1$-differentiable vector valued function.

Definition 1 (Koopman operator) *Let $\mathcal{K}(\mathcal{S} \times \mathcal{A})$ be the (Banach) space of all measurement functions (observables). Then the Koopman operator $\mathcal{K} : \mathcal{K}(\mathcal{S} \times \mathcal{A}) \to \mathcal{K}(\mathcal{S} \times \mathcal{A})$ is defined by*

$$\mathcal{K}g(s_t, a_t) = g\big(F(s_t, a_t), a_{t+1}\big) = g(s_{t+1}, a_{t+1}) \ , \ \ \forall g \in \mathcal{K}(\mathcal{S} \times \mathcal{A}) \tag{7}$$

where $g : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$.

Many systems can be modeled by a bilinearisation where the action enters the controlling equations 5 linearly as $f(s, a) = f_0(s) + \sum_i^m f_i(s)a_i$ for functions $f_{i=0,\dots,m}$. In that case the action of the Koopman operator takes the simple form

$$g(s_{t+1}) = \mathcal{K}(a)g(s_t) = \Big(\mathcal{K}_0 + \sum_i^m \mathcal{K}_i a_i\Big)g(s_t) \ , \ \ \forall g \in \mathcal{K}(\mathcal{S} \times \mathcal{A}) \tag{8}$$

where $\mathcal{K}_0, \mathcal{K}_i$ decompose the Koopman operator in a the free and forcing-term, respectively. Associated with a Koopman operator is its eigenspectrum, that is, the eigenvalues λ, and the corresponding eigenfunctions $\varphi_\lambda(s, a)$, such that $[\mathcal{K}\varphi_\lambda](s, a) = \lambda\varphi_\lambda(s, a)$. In practice one derives a finite set of observable $\vec{g} = (g_1, \dots, g_N)$ in which case the approximation to the Koopman operator admits a finite-dimensional matrix representation. The $N \times N$ matrix representing the Koopman operator may be diagonalized by a matrix U containing the eigen-vectors of $\mathcal{K}$ as columns. In which case the eigenfunctions are derived by $\vec{\varphi} = U\vec{g}$ and one infers from Eq. (5) that $\dot{\varphi}_i = \lambda_i\varphi_i \ , \ \ \text{for} \ \ i = 1, \dots, N$ with eigenvalues $\lambda_{i=1,\dots,N}$. These ODE's admit simple solutions for their time-evolution namely the exponential functions $\exp(\lambda_i t)$ which is expected for a linear system.

3 The Koopman forward framework

In this section we introduce the Koopman forward framework. We initiate our discussion with a focus on symmetries in dynamical control systems in Subsection 3.1 where we additionally present some theoretical results. We then proceed in Subsection 3.2 by presenting the Koopman forward framework for Q-learning based on CQL. Moreover, we discuss the algorithm as well as the Koopman Forward model's deep neural network architecture.

Overview: Simply, our theoretical results culminate in Theorem 3.6 and 3.7, which provides a road-map on how specific symmetries of the dynamical control system are to be inferred from a VAE forward prediction model. In particular, Theorem 3.6 guarantees that the procedure leads to true symmetries of the system and Theorem 3.7 that actual new data points can be derived by applying symmetry transforming to existing ones. The practical limitation is that the VAE model, parameterized by a neural net, is trained on the collected data from one of many behavior policies which amounts to learning the approximate dynamics of the real system. The theoretical limitations are twofold, firstly the theorems only hold for dynamical systems with differentiable state-transitions; secondly, we employ a Bilinearisation Ansatz for the Koopman operator of the system. Practically many RL environments incorporate dynamics with discontinuous "contact" events where the Bilinearisation Ansatz may not be applicable. However, empirically we find that our approach nevertheless is successful for such RL environments and such "contact" events do not affect the performance significantly (see Appendix C.1).

3.1 Symmetries of dynamical control system

Let us start by introducing symmetries in the simpler context of dynamical system without control (Sinha et al., 2020) given by $\dot{s} = f(s)$ where we use analog notations as in Eq. (5).

Definition 2 (Equivariant Dynamical System) *Consider the dynamical system $\dot{s} = f(s)$ and let Σ be a group acting on the state-space $\mathcal{S}$. Then the system is called Σ-equivariant if $f(\sigma \cdot s) = \sigma \cdot f(s)$, for $s \in \mathcal{S}$, $\forall \sigma \in \Sigma$. For a discrete time dynamical system $s_{t+1} = F(s_t)$ one defines equivariance analogously, namely if $F(\sigma \cdot s_t) = \sigma \cdot F(s_t)$, for $s_t \in \mathcal{S}$, $\forall \sigma \in \Sigma$.*

Lemma 3.1 *The map $\phi : \Sigma \times \mathcal{K}(\mathcal{S}) \to \mathcal{K}(\mathcal{S})$ given by $(\sigma * g)(s) \longmapsto g(\sigma^{-1} \cdot s)$ defines a group action on the Koopman space of observables $\mathcal{K}(\mathcal{S})$.*

Theorem 3.2 *Let $\mathcal{K}$ be the Koopman operator associated with a Σ-equivariant system $s_{t+1} = F(s_t)$. Then*

$$[\sigma * (\mathcal{K}g)](s) = [\mathcal{K}(\sigma * g)](s) \ \ . \tag{9}$$

Theorem 3.2 states that for a Σ-equivariant system any symmetry transformation commutes with the Koopman operator. For the proof see (Sinha et al., 2020). Let us next turn to the case relevant for RL, namely control systems. In the remainder of this section we focus on dynamical systems given as in Eq. (6) and Eq. (8).

Definition 3 (Action Equivariant Dynamical System) *Let $\bar{\Sigma}$ be a group acting on the state-action-space $\mathcal{S} \times \mathcal{A}$ of a general control system as in Eq. (6) such that it acts as the identity operation on $\mathcal{A}$ i.e. $\sigma \cdot (s_t, a_t) = (\sigma|_{\mathcal{S}} \cdot s_t, a_t)$, $\forall \sigma \in \bar{\Sigma}$. Then the system is called $\bar{\Sigma}$-action-equivariant if*

$$F(\sigma \cdot (s_t, a_t)) = \sigma|_{\mathcal{S}} \cdot F(s_t, a_t) \ , \ \ \textit{for } (s_t, a_t) \in \mathcal{S} \times \mathcal{A} \ , \ \ \forall \sigma \in \bar{\Sigma} \ . \tag{10}$$

In particular, it is easy to see that the biliniarisation in Eq. (8) is Σ-action-equivariant if $f_i(\sigma|_{\mathcal{S}} \cdot s) = \sigma|_{\mathcal{S}} \cdot f_i(s)$, $\forall i = 0, \dots, m$.

Lemma 3.3 *The map $\bar{\phi} : \bar{\Sigma} \times \mathcal{K}(\mathcal{S} \times \mathcal{A}) \to \mathcal{K}(\mathcal{S} \times \mathcal{A})$ given by $(\sigma \bar{*} g)(s, a) \longmapsto g(\sigma^{-1} \cdot s, a)$ defines a group action on the Koopman space of observables $\mathcal{K}(\mathcal{S} \times \mathcal{A})$.*

Theorem 3.4 *Let $\mathcal{K}$ be the Koopman operator associated with a $\bar{\Sigma}$-action-equivariant system $s_{t+1} = F(s_t, a_t)$. Then*

$$[\sigma \bar{*} (\mathcal{K} g)](s_t, a_t) = [\mathcal{K}(\sigma \bar{*} g)](s_t, a_t) \ . \tag{11}$$

Neither Theorem 3.2 nor 3.4 serve as a theoretical foundation for the control setup aimed for in the next section. Let us thus turn our focus on the relevant case of a control system $s_{t+1} = \tilde{F}(s_t, a_t)$ which admits a Koopman operator description as

$$g(s_{t+1}) = \mathcal{K}(a_t) g(s_t) \ , \ \ \text{for } a_t \in \mathcal{A} \, , \ \forall g \in \mathcal{K}(\mathcal{S}) \ , \tag{12}$$

where $\{\mathcal{K}(a)\}_{a \in \mathcal{A}}$ is a family of operators with analytical dependence on $a \in \mathcal{A}$. Note that the bilinearisation in Eq. (8) is a special case of Eq. (12). Furthermore, let $\{U(a)\}_{a \in \mathcal{A}}$ be a family of invertible operators s.t. $U(a) : \mathcal{K}(\mathcal{S}) \to \mathcal{F}(\mathcal{S} \times \mathcal{A})$ is a mapping to the (Banach) space of eigenfunctions $\varphi(s, a) := U(a) g(s) \in \mathcal{F}(\mathcal{S} \times \mathcal{A})$. Which moreover obeys $U(a) \mathcal{K}(a) U(a)^{-1} = \Lambda(a)$, with $\Lambda(a) \varphi(s, a) = \lambda_\varphi(a) \varphi(s, a)$ and where $\lambda_\varphi(a) : \mathcal{A} \to \mathbb{R}$. The existence of such operators puts further restriction on the Koopman operator in Eq. (12).[1] As we are in the end concerned with the finite-dimensional approximation of the Koopman operator i.e. its matrix representation this amounts simple for $\mathcal{K}(a)$ to be diaganolisable and $U(a)$ is the matrix containing its eigen-vectors as columns.

Lemma 3.5 *The map $\hat{\phi} : \bar{\Sigma} \times \mathcal{K}(\mathcal{S}) \times \{U(a)\}_{a \in \mathcal{A}} \to \mathcal{K}(\mathcal{S})$ given by*

$$(\sigma_a \hat{*} g)(s) \longmapsto \Big(U^{-1}(a) \big(\sigma \bar{*} (U(a) g) \big) \Big)(s) = g(\sigma^{-1} \cdot s) \ , \tag{13}$$

defines a group action on the Koopman space of observables $\mathcal{K}(\mathcal{S})$. Where $\bar{*}$ is defined analog to Lemma 3.3 but acting on $\mathcal{F}(\mathcal{S} \times \mathcal{A})$ instead of $\mathcal{K}(\mathcal{S} \times \mathcal{A})$ by $(\sigma \bar{*} \varphi)(s, a) \longmapsto \varphi(\sigma^{-1} \cdot s, a)$.

We refer to a system in Eq. (12) admitting a symmetry as in Lemma 3.5 as $\hat{\Sigma}$-action-equivariant control system in the following.

Theorem 3.6 *Let $s_{t+1} = \tilde{F}(s_t, a_t)$ be a $\hat{\Sigma}$-action-equivariant control system with a symmetry action as in Lemma 3.5 which furthermore admits a Koopman operator representation as*

$$g(s_{t+1}) = \mathcal{K}(a_t) g(s_t) \ , \ \ \textit{for } a_t \in \mathcal{A} \, , \ \forall g \in \mathcal{K}(\mathcal{S}) \ . \tag{14}$$

Then

$$\big[\sigma_{a_t} \hat{*} \big(\mathcal{K}(a_t) g \big) \big](s) = \big[\mathcal{K}(a_t) \, \big(\sigma_{a_t} \hat{*} g \big) \big](s) \ . \tag{15}$$

Moreover, the converse is also true. If a control system $s_{t+1} = \tilde{F}(s_t, a_t)$ obeys Equations 14 and 15, then it is $\hat{\Sigma}$-action-equivariant, i.e. $\sigma \cdot \tilde{F}(s_t, a_t) = \tilde{F}(\sigma \cdot (s_t, a_t))$.

Let us next build on the core of Theorem 3.6 to provide theoretical statements on how data-points may be shifted by symmetry transformations. For that let us change our notation to establish an easier connection to the. Let $E : \mathcal{S} \to \mathcal{K}(S)$ and $D : \mathcal{K}(S) \to \mathcal{S}$ denote the encoder and decoder to and from the Koopman space approximation, respectively, i.e. $E \circ D = D \circ E = id$.

[1] See Appendix A for the details.

Theorem 3.7 *Let $s_{t+1} = \tilde{F}(s_t, a_t)$ be a $\hat{\Sigma}$-action-equivariant control system as in Theorem 3.6. Then one can use a symmetry transformation to shift both s_t as well as s_{t+1}*

$$\sigma^{\epsilon}_{a_t} : \ \big(s_t, s_{t+1}, a_t\big) \ \longmapsto \ \big(\tilde{s}_t, \tilde{s}_{t+1}, a_t\big) \ , \ \ s.t. \ \ \tilde{s}_{t+1} = \tilde{F}(\tilde{s}_t, a_t) + \mathcal{O}(\epsilon) \ , \tag{16}$$

where $\epsilon \ll 1$. Thus the symmetry shift of the data-point reflects the real dynamic of the system up to an controllable error. More formally, the above map in Eq. (16) is given by [2]

$$\tilde{s}_t = D\Big(\big(\mathbb{1} \ + \ \epsilon\, \sigma_{a_t}\big) \hat{*} E(s_t)\Big) \ , \ \ \tilde{s}_{t+1} = D\Big(\big(\mathbb{1} \ + \ \epsilon\, \sigma_{a_t}\big) \hat{*} E(s_{t+1})\Big) \ . \tag{17}$$

3.2 KFC ALGORITHM

In the previous section we laid the theoretical foundation for the KFC-algorithm by providing a raod-map on how to derive symmetries of dynamical control systems based on a Koopman latent space representation. The goal is to generate new data points for the RL algorithm at training-time as Eq. (16) in Theorem 3.7. The reward r_t is not part of the symmetry shift process and will just remain unchanged as an assertion.

On the power of using symmetries: Let us emphasize the practical advantage of employing symmetries to generate augmented data points. It is evident from Theorem 3.7 that a symmetry transformation shifts both s_t as well as s_{t+1} which evades the necessity of forecasting states. Thus the use of an inaccurate fore-cast model is avoided and the accuracy and generalisation capabilities of the VAE are fully utilized. The theoretically induced error can be controlled by the parameter $\epsilon \ll 1$ which lead to a state shift $|s - \tilde{s}| = \delta(\epsilon)$. [3]

Let us next discuss technical details of the incorporation of the symmetry shifts into a specific Q-learning algorithm. In practice the Koopman latent space representation is N-dimensional i.e. finite. One uses the matrix representations of the Koopman operator and the symmetry generators and matrix multiplication instead of the formal mapping $\hat{*}$. Moreover, we use a normally distributed random variable for the control parameter ϵ. We base our analysis on the CQL algorithm, however we expect that our approach of exploration should benefit a wider range of RL algorithms. Following in the footsteps of (Sinha et al., 2021) our approach leaves the policy improvement as in Eq. (3) unchanged but modifies the policy optimisation step in Eq. (4) as

$$\begin{aligned} \hat{Q}_{i+1} \ \leftarrow \ \arg\min_{Q} \Bigg(& \mathop{\mathbb{E}}_{s_t, a_t, s_{t+1} \sim \mathcal{D}} \Big[r_t + \gamma\, \hat{Q}_i\Big(\sigma^{\epsilon}_{a_t}\big(\tilde{s}_{t+1} | s_{t+1}\big), a_{t+1}\Big) - Q_i\Big(\sigma^{\epsilon}_{a_t}\big(\tilde{s}_t | s_t\big), a_t\Big) \Big] \\ & + \tilde{\alpha} \mathop{\mathbb{E}}_{s_t \sim \mathcal{D}} \Big[\log \sum_a \exp\big(Q_i(s_t, a)\big) - \mathop{\mathbb{E}}_{a \sim \pi(s_t)} \big[Q_i(s_t, a) \big] \Big] \Bigg) \end{aligned} \tag{18}$$

The state-space symmetry generating function $\sigma^{\epsilon}_{a} : \mathcal{S} \to \mathcal{S}$ depends on the normally distributed random variables ϵ. Note that we only modify the Bellman error of Eq. (18) and leave the CQL specific regulizer untouched. We study two distinct cases which differ on an algorithmic level[4]

$$\begin{aligned} &\text{(I)} \ \ \sigma^{\epsilon}_{a_t} : s \mapsto \tilde{s} = D\Big(\big(\mathbb{1} \ + \ \epsilon\, \sigma_{a_t}\big) E(s)\Big) \qquad \text{(II)} \ \ \sigma^{\epsilon}_{a_t} : s \mapsto \tilde{s} = D\Big(\big(\mathbb{1} \ + \ \sigma_{a_t}(\vec{\epsilon})\big) E(s)\Big) \ , \\ &\qquad\qquad \text{where} \quad \sigma_{a_t}(\vec{\epsilon}) = \mathcal{R}\mathrm{e}\big(\, U(a_t)\, \mathrm{diag}(\epsilon_1, \ldots, \epsilon_N)\, U^{-1}(a_t)\big) \ , \end{aligned} \tag{19}$$

and $U(a_t)$ diagonalizes the Koopman operator with eigenvalues $\lambda_i \ , \ i = 1, \ldots, N$ i.e. the latter can be expressed as $\mathcal{K}(a_t) = U(a_t)\mathrm{diag}(\lambda_1, \ldots, \lambda_N) U^{-1}(a_t)$. Moreover, $\sigma_{a_t} \in \hat{\Sigma}$ is a fixed $N \times N$ matrix representation of a symmetry transformation of the dynamical system in Koopman space. Note that we abuse our notation as $E(s) \equiv \vec{g}(s) = [g_1(s), \ldots, g_N(s)]$ i.e. the encoder provides the Koopman space observables. Following the guidelines provided by Theorem 3.6 for a symmetry acting on the state-action space of dynamical control systems in Eq. (8) as $(s_{t+1}, s_t, a_t) \to (\tilde{s}_{t+1}, \tilde{s}_t, a_t)$ it requires the symmetry generators to commute with the Koopman operator. Thus on the one hand, for the case (I) in Eq. (19) the symmetry generator matrix σ_a is obtained by solving the equation $\sigma_{a_t} \cdot \mathcal{K}(a_t) - \mathcal{K}(a_t) \cdot \sigma_{a_t} = 0$ which may be accomplished by employing a Sylvester algorithm

[2] The equivalent theorem holds when $\epsilon\, \sigma_{a_t} \to \sum_{I=1} \epsilon_I\, \sigma^I_{a_t}$.

[3] For the details of the errors, refer the appendix section A.

[4] Eq. (19) holds for both s_t as well as s_{t+1} thus we have deliberately dropped the subscript. Moreover, note that in case (II) the symmetry transformations are of the form $\sigma_{a_t}(\vec{\epsilon})\big) = \sum_{I=1} \epsilon_I\, \sigma^I_{a_t}$.

(syl). While on the other hand case (II) is solved by computing the eigen-vectors of $\mathcal{K}(a)$ which then constitute the columns of $U(a)$. Thus in particular one infers that

$$\left[\, \sigma_{a_t}(\vec{\epsilon}) \,,\, \mathcal{K}(a_t) \,\right] = 0 \quad , \tag{20}$$

where $[\cdot, \cdot]$ denotes the commutator of two matrices. Eq. (20) is solved by construction of $\sigma_a(\vec{\epsilon})$ which commutes with the Koopman operator for all values of the random variables. [5] Concludingly, the advantage of case (I) is that it is computationally less expensive, however it it provides less freedom to explore different symmetry directions as case (II). Lastly, it proofed useful empirically to train the RL-algorithm not only on symmetry shifted data as in Eq. (19). We employ our data exploration shift only with probability p_K. Otherwise we use $s \to \tilde{s} = s + \tilde{\epsilon}$, i.e. a shift by a normally distributed random variable.

The Koopman forward model: The KFC algorithm requires pre-training of a Koopman forward model $\mathcal{F} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ which is closely related to a VAE architecture as

$$\mathcal{F}^c(s_t, a_t) = \begin{cases} D\big(E(s_t)\big) = s_t & \text{if } c = 0\text{: VAE} \\ D\Big(\big(\mathcal{K}_0 + \sum_{i=1}^{m} \mathcal{K}_i \, a_{t,i}\big) E(s_t) \Big) = s_{t+1} & \text{if } c = 1\text{: forward prediction} \end{cases} \tag{21}$$

where both of E and D are approximated by Multi-Layer-Perceptrons (MLP's) and the bilinear Koopman-space operator approximation are implemented by a single fully connected layer for $\mathcal{K}_{i=0,\dots,m}$, respectively. The model is trained on batches of the offline data-set tuples (s_{t+1}, s_t, a_t) and optimized via an additive loss-function of the VAE and the forward prediction part of the model. We refer the reader to Appendix B for details.

4 Empirical Evaluation

In this section, we will first experiment with the popular D4RL benchmark commonly used for offline RL (Fu et al., 2021). The benchmark covers various different tasks such as locomotion tasks with Mujoco Gym (Brockman et al., 2016), tasks that require hierarchical planning such as antmaze, and other robotics tasks such as kitchen and adroit (Rajeswaran et al., 2017). Furthermore, similar to S4RL (Sinha et al., 2021), we perform experiments on 6 different challenging robotics tasks from MetaWorld (Yu et al., 2019) and RoboSuite (Zhu et al., 2020). We compare KFC to the baseline CQL algorithm (Kumar et al., 2020), and two best performing augmentation variants from S4RL, S4RL-$\mathcal{N}$ and S4RL-adv (Sinha et al., 2021). We use the exact same hyperparameters as proposed in the respective papers. Furthermore, similar to S4RL, we build KFC on top of CQL (Kumar et al., 2020) to ensure conservative Q-estimates during for policy evaluation.

4.1 D4RL benchmarks

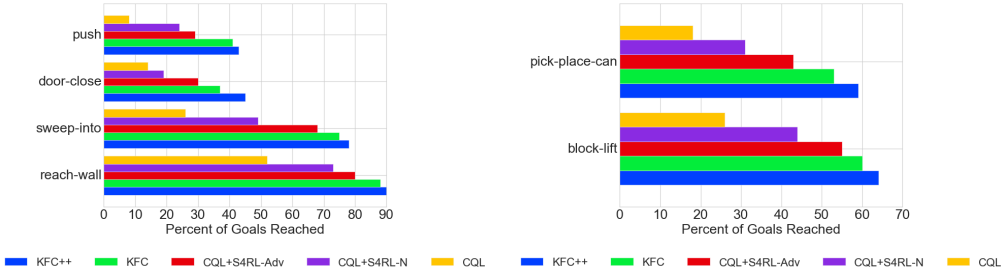
We present results in the benchmark D4RL test suite and report the normalized return in Table 1. We see that both KFC and KFC++ consistently outperform both the baseline CQL and S4RL across multiple tasks and data distributions. Outperforming S4RL-$\mathcal{N}$ and S4RL-adv on various different types of environments suggests that KFC and KFC++ fundamentally improves the data augmentation strategies discussed in S4RL. KFC-variants also improve the performance of learned agents on challenging environments such as antmaze: which requires hierarchical planning, kitchen and adroit tasks: which are sparse reward and have large action spaces. Similarly, KFC-variants also perform well on difficult data distributions such as "medium-replay": which is a collected by simply using all the data that the policy encountered while training base SAC policy, and "-human" which is collected using human demonstrations on robotic tasks which results in a non-Markovian behaviour policy (more details can be found in the D4RL manuscript (Fu et al., 2021)). Furthermore, to our knowledge, the results for KFC++ are state-of-the-art in policy learning from D4RL datasets for most environments and data distributions.

We do note that S4RL outperforms KFC on the "-random" split of the data distributions, which is expected as KFC depends on learning a simple dynamics model of the data to use to guide the data augmentation strategy. Since the "-random" split consists of random actions in the environment, the dynamics model is unable to learn a useful dynamics model.

[5] Moreover, let us emphasize that the Koopman operators eigenvalues and eignevectors generically are $\mathbb{C}$-valued. However, the definition in Eq. (19) ensures that $\sigma_a(\vec{\epsilon})$ is a $\mathbb{R}$-valued matrix.

Domain	Task Name	CQL	S4RL-($\mathcal{N}$)	S4RL-(Adv)	KFC	KFC++
AntMaze	antmaze-umaze	74.0	91.3	94.1	**96.9**	**99.8**
	antmaze-umaze-diverse	84.0	87.8	88.0	**91.2**	**91.1**
	antmaze-medium-play	61.2	61.9	61.6	60.0	**63.1**
	antmaze-medium-diverse	53.7	78.1	82.3	87.1	**90.5**
	antmaze-large-play	15.8	24.4	**25.1**	24.8	**25.6**
	antmaze-large-diverse	14.9	27.0	26.2	**33.1**	**34.0**
Gym	cheetah-random	35.4	**52.3**	**53.9**	48.6	49.2
	cheetah-medium	44.4	48.8	48.6	55.9	**59.1**
	cheetah-medium-replay	42.0	51.4	51.7	**58.1**	**58.9**
	cheetah-medium-expert	62.4	**79.0**	78.1	**79.9**	**79.8**
	hopper-random	**10.8**	**10.8**	**10.7**	**10.4**	**10.7**
	hopper-medium	58.0	78.9	81.3	90.6	**94.2**
	hopper-medium-replay	29.5	35.4	36.8	**48.6**	**49.0**
	hopper-medium-expert	111.0	113.5	117.9	121.0	**125.5**
	walker-random	7.0	**24.9**	**25.1**	19.1	17.6
	walker-medium	79.2	93.6	93.1	102.1	**108.0**
	walker-medium-replay	21.1	30.3	35.0	**48.0**	46.1
	walker-medium-expert	98.7	112.2	107.1	114.0	**115.3**
Adroit	pen-human	37.5	44.4	51.2	**61.3**	60.0
	pen-cloned	39.2	57.1	58.2	**71.3**	68.4
	hammer-human	4.4	5.9	6.3	7.0	**9.4**
	hammer-cloned	2.1	2.7	2.9	3.0	**4.2**
	door-human	9.9	27.0	35.3	44.1	**46.1**
	door-cloned	0.4	2.1	0.8	3.6	**5.6**
	relocate-human	**0.2**	**0.2**	**0.2**	**0.2**	**0.2**
	relocate-cloned	**-0.1**	**-0.1**	**-0.1**	**-0.1**	**-0.1**
Franka	kitchen-complete	43.8	77.1	88.1	**94.1**	**94.9**
	kitchen-partial	49.8	74.8	83.6	92.3	**95.9**

Table 1: We experiment with the full set of the D4RL tasks and report the mean normalized episodic returns over 5 random seeds using the same protocol as Fu et al. (2021). We compare against 3 competitive baselines including CQL and the two best performing S4RL-data augmentation strategies. We see that KFC and KFC++ consistently outperforms the baselines. We use the baseline numbers reported in Sinha et al. (2021).



(a) MetaWorld Environments **(b)** RoboSuite Environments

Figure 2: Results on challenging dexterous robotics environments using data collected using a similar strategy as S4RL (Sinha et al., 2021). We report the % of goals that the agent is able to reach during evaluation, where the goal is set by the environments. **We see that KFC and KFC++ consistently outperforms both CQL and the two best performing S4RL variants.**

4.2 METAWORLD AND ROBOSUITE BENCHMARKS

To further test the ability of KFC, we perform additional experiments on challenging robotic tasks. Following (Sinha et al., 2021), we perform additional experiments with 4 MetaWorld environments

(Yu et al., 2019) and 2 RoboSuite environments (Zhu et al., 2020). We followed the same method to collect the data as described in Appendix F of S4RL (Sinha et al., 2021), and report the mean percent of goals reached, where the condition of reaching the goal is defined by the environment.

We report the results in Figure 2, where we see that using KFC to guide the data augmentation strategy for a base CQL agent, we are able to learn an agent that performs significantly better. Furthermore, we see that for more challenging tasks such as "push" and "door-close" in the MetaWorld, KFC++ outperforms the base CQL algorithm and the S4RL agent by a significant margin. These set of experiments further highlight the ability of KFC to guide the data augmentation strategy.

5 Related works

The use of data augmentation techniques in Q-learning has been discussed recently (Laskin et al., 2020b;a; Sinha et al., 2021). In particular, our work shares strong parallels with (Sinha et al., 2021). Our modification of the policy evaluation step of the CQL algorithm (Kumar et al., 2020) is analogous to the one in Sinha et al. (2021). However, the latter randomly augments the data while our augmentation framework is based on symmetry state shifts. Regarding the connection to world models (Ha & Schmidhuber, 2018). Here a VAE is used to decode the state information while a recurrent separate neural network predicts future states. Their latent representation is not of Koopman type. Also no symmetries and data-augmentations are derived.

Algebraic symmetries of the state-action space in Markov Decision Processes (MDP) originate (Balaraman & Andrew, 2004) an were discussed recently in the context of RL in (van der Pol et al., 2020). Their goal is to preserve the essential algebraic homomorphism symmetry structure of the original MDP while finding a more compact representation. The symmetry maps considered in our work are more general and are utilized in a different way. Symmetry-based representation learning (Higgins et al., 2018) refers to the study of symmetries of the environment manifested in the latent representation. The symmetries in our case are derived form the Koopman operator not the latent representation directly. In (Caselles-Dupré et al., 2019) the authors discuss representation learning of symmetries (Higgins et al., 2018) allowing for interactions with the environment. A Forward-VAE model which is similar to our Koopman-Forward VAE model is employed. However, our approach is based on theoretical results providing a road-map to derive explicit symmetries of the dynamical systems as well as their utilisation for state-shifts.

In (Sinha et al., 2020) the authors extend the Koopman operator from a local to a global description using symmetries of the dynamics. They do not discuss action-equivariant dynamical control systems nor data augmentation. In (Salova et al., 2019) the imprint of known symmetries on the block-diagonal Koopman space representation for non-control dynamical systems is discussed. This is close to the spirit of disentanglement (Higgins et al., 2018). Our results are on control setups and deriving symmetries. On another front, the application of Koopman theory in control or reinforcement learning has also been discussed recently. For example, Li et al. (2020) propose to use compositional Koopman operators using graph neural networks to learn dynamics that can quickly adapt to new environments of unknown physical parameters and produce control signals to achieve a specified goal. Kaiser et al. (2021) discuss the use of Koopman eigenfunction as a transformation of the state into a globally linear space where the classical control techniques is applicable. To the best of our knowledge, this paper is the first to discuss Koopman latent space for data augmentation.

6 Conclusions

In this work we proposed a symmetry-based data augmentation technique derived from a Koopman latent space representation. It enables a meaningful extension of offline RL datasets describing dynamical systems, i.e. further "exploration" without additional environment interactions. The approach is based on our theoretical results on symmetries of dynamical control systems and symmetry shifts of data. Both hold for systems with differentiable state transitions and with a Bilinearisation Ansatz for the Koopman operator. However, the empirical results show that the framework is successfully applicable beyond those limitations. We empirically evaluated our method on several benchmark offline reinforcement learning tasks D4RL, Metaworld and Robosuite and find that by using our framework we consistently improve the state-of-the-art of Q-learning algorithms.

REFERENCES

Sylvester algorithm - scipy.org. URL https://docs.scipy.org/doc/scipy-0.15.1/reference/generated/scipy.linalg.solve_sylvester.html.

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning, 2020a.

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, 2020b.

Ravindran Balaraman and G. Barto Andrew. Approximate homomorphisms: A framework for non-exact minimization in markov decision processes. In *In International Conference on Knowledge Based Computer Systems*, 2004.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Steven L. Brunton, Marko Budišić, Eurika Kaiser, and J. Nathan Kutz. Modern koopman theory for dynamical systems, 2021.

Hugo Caselles-Dupré, Michael Garcia Ortiz, and David Filliat. Symmetry-based disentangled representation learning requires interaction with environments. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(18):503–556, 2005.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2021.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2052–2062. PMLR, 09–15 Jun 2019.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/2de5d16682c3c35007e4e92982f1a2ba-Paper.pdf.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1352–1361. PMLR, 06–11 Aug 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 10–15 Jul 2018.

Irina Higgins, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, and Alexander Lerchner. Towards a definition of disentangled representations, 2018.

Eurika Kaiser, J. Nathan Kutz, and Steven L. Brunton. Data-driven discovery of Koopman eigenfunctions for control. *Machine Learning: Science and Technology*, 2:035023, 2021.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Scalable deep reinforcement learning for vision-based robotic manipulation. In Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto (eds.), *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pp. 651–673. PMLR, 29–31 Oct 2018.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Bernard Osgood Koopman. Hamiltonian systems and transformation in Hilbert space. *Proceedings of the National Academy of Sciences of the United States of America*, 17(5):315–318, 1931.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1179–1191. Curran Associates, Inc., 2020.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. CURL: Contrastive unsupervised representations for reinforcement learning. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5639–5650. PMLR, 13–18 Jul 2020a.

Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 19884–19895. Curran Associates, Inc., 2020b.

Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pp. 661–670, 2010.

Yunzhu Li, Hao He, Jiajun Wu, Dina Katabi, and Antonio Torralba. Learning compositional Koopman operators for model-based control. In *Proc. of the 8th Int'l Conf. on Learning Representation (ICLR'20)*, 2020.

Alexandre Mauroy, Igor Mezić, and Yoshihiko Susuki. *The Koopman Operator in Systems and Control: Concepts, Methodologies, and Applications*. Springer, 2020.

I. Mezić. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, 41:309–325, 2005.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32: 8026–8037, 2019.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

Anastasiya Salova, Jeffrey Emenheiser, Adam Rupe, James P. Crutchfield, and Raissa M. D'Souza. Koopman operator and its approximations for systems with symmetries. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(9):093128, 2019. doi: 10.1063/1.5099091.

Samarth Sinha, Ajay Mandlekar, and Animesh Garg. S4RL: Surprisingly Simple Self-Supervision for Offline Reinforcement Learning. In *Conference on Robot Learning*, 2021.

Subhrajit Sinha, Sai P. Nandanoori, and Enoch Yeung. Koopman operator methods for global phase space exploration of equivariant dynamical systems, 2020.

Elise van der Pol, Daniel Worrall, Herke van Hoof, Frans Oliehoek, and Max Welling. Mdp homomorphic networks: Group symmetries in reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 4199–4210. Curran Associates, Inc., 2020.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning, 2019.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. *arXiv preprint arXiv:1910.10897*, 2019.

Yuke Zhu, Josiah Wong, Ajay Mandlekar, and Roberto Martín-Martín. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

A PROOFS

In this section we provide the proofs of the theoretical results of Section 3.

Proof of Lemma 3.1 and Theorem 3.2 The proofs of the lemma as well as the theorem can be found in (Sinha et al., 2020).

Proof of Lemma 3.3: We aim to show that map $\bar{\phi} : \bar{\Sigma} \times \mathcal{K}(\mathcal{S} \times \mathcal{A}) \to \mathcal{K}(\mathcal{S} \times \mathcal{A})$ given by $(\sigma \bar{*} g)(s, a) \longmapsto g(\sigma^{-1} \cdot s, a)$ defines a group action on the Koopman space of observables $\mathcal{K}(\mathcal{S} \times \mathcal{A})$, where $\bar{\Sigma}$ defines the symmetry group of definition 3. Firstly, let $\sigma_0 \in \bar{\Sigma}$ and $g \in \mathcal{K}(\mathcal{S} \times \mathcal{A})$ be the identity element, then we see that it provides existence of an identity element of $\bar{\phi}$ by

$$(\sigma_0 \bar{*} g)(s, a) = g\big(\sigma_0^{-1} \cdot s, a\big) = g(s, a) \ .$$

Secondly, let $\sigma_1, \sigma_2 \in \bar{\Sigma}$ and $\odot$ denoting the group operation i.e. $\sigma_1 \odot \sigma_2 = \sigma_3 \in \bar{\Sigma}$. Then

$$\begin{aligned}\Big(\sigma_2 \bar{*} \big(\sigma_1 \bar{*} g\big)\Big)(s, a) \quad &= \sigma_2 \bar{*} \Big(g\big(\sigma_1^{-1} \cdot s, a\big)\Big) = g\big(\sigma_2^{-1} \cdot (\sigma_1^{-1} \cdot s), a\big) = g\big((\sigma_2^{-1} \odot \sigma_1^{-1}) \cdot s), a\big) \\ &\overset{(I)}{=} g\big((\sigma_1 \odot \sigma_2)^{-1} \cdot s, a\big) = \Big(\big(\sigma_1 \odot \sigma_2\big) \bar{*} g\Big)(s, a) = \Big(\sigma_3 \bar{*} g\Big)(s, a) \ ,\end{aligned}$$

where in (I) we have used the invertibility of the group property of $\bar{\Sigma}$. Lastly it follows analogously that for $\sigma, \sigma^{-1} \in \bar{\Sigma}$ that

$$\Big(\sigma \bar{*} \big(\sigma^{-1} \bar{*} g\big)\Big)(s, a) = \Big(\big(\sigma^{-1} \odot \sigma\big) \bar{*} g\Big)(s, a) = \Big(\sigma_0 \bar{*} g\Big)(s, a) = g(s, a) \ .$$

Thus the existence of an inverse is established which concludes to show the group property of $\bar{\phi}$.

Proof of Theorem 3.4: We aim to show that with $\mathcal{K}$ being the Koopman operator associated with a $\bar{\Sigma}$-action-equivariant system $s_{t+1} = F(s_t, a_t)$. Then

$$\Big[\sigma \bar{*} \big(\mathcal{K} g\big)\Big](s, a) = \Big[\mathcal{K}\big(\sigma \bar{*} g\big)\Big](s, a) \ .$$

First of all note that by the definition of the Koopman operator of non-affine control systems it obeys

$$[\mathcal{K} g](s_t, a_t) = g(F(s_t, a_t), a_{t+1}) \ .$$

Using the latter one thus infers that

$$\Big[\sigma \bar{*} \big(\mathcal{K} g\big)\Big](s_t, a_t) = \sigma \bar{*} g(F(s_t, a_t), a_{t+1}) = g(\hat{\sigma}^{-1} \cdot F(s_t, a_t), a_{t+1}) \overset{(I)}{=} g(F(\sigma^{-1} \cdot s_t, a_t), a_{t+1}) \, ,$$

where in (I) we have used that it is a $\bar{\Sigma}$-action-equivariant system. Moreover, one finds that

$$g(F(\sigma^{-1} \cdot s_t, a_t), a_{t+1}) = \mathcal{K} g(\sigma^{-1} \cdot s_t, a_t) = \Big[\mathcal{K}\big(\sigma \bar{*} g\big)\Big](s_t, a_t) \ ,$$

which concludes the proof.

To proceed let us re-evaluate some information from the main text of this work. Let $\{U(a)\}_{a \in \mathcal{A}}$ be a family of invertible operators s.t. $U(a) : \mathcal{K}(\mathcal{S}) \to \mathcal{F}(\mathcal{S} \times \mathcal{A})$ is a mapping to the (Banach) space of eigenfunctions $\varphi(s, a) := U(a) g(s) \in \mathcal{F}(\mathcal{S} \times \mathcal{A})$. Which moreover obeys $U(a) \mathcal{K}(a) U(a)^{-1} = \Lambda(a)$, with $\Lambda(a) \varphi(s, a) = \lambda_\varphi(a) \varphi(s, a)$ and where $\lambda_\varphi(a) : \mathcal{A} \to \mathbb{R}$. The existence of such operators puts further restriction on the Koopman operator in Eq. (12). However, as our algorithm employs the finite-dimensional approximation of the Koopman operator i.e. its matrix representation this amounts simple for $\mathcal{K}(a)$ to be diagonalizable and $U(a)$ is the matrix containing its eigen-vectors as columns. To evolve a better understanding on the required criteria for the infinite-dimensional case we employ an alternative formulation of the so called spectral theorem below.

Theorem A.1 (Spectral Theorem) *Let $\mathcal{K}$ be a bounded self-adjoint operator on a Hilbert space $\mathcal{H}$. Then there is a measure space $(\mathcal{S}, \Sigma, \mu)$ and a real-valued essentially bounded measurable function λ on $\mathcal{S}$ and a unitary operator $U : \mathcal{H} \to L^2_\mu(\mathcal{S})$ i.e. $U^* U = U U^* = id$ such that*

$$U \, \Lambda \, U^* = \mathcal{K} \ , \quad \textit{with} \quad \big[\Lambda \varphi\big](s) = \lambda(s) \varphi(s) \ . \tag{22}$$

In other words every bounded self-adjoint operator is unitarily equivalent to a multiplication operator. In contrast to the finite-dimensional case we need to slightly alter our criteria to $U(a)\mathcal{K}(a)U(a)^{-1} = \Lambda(a)$, with $\Lambda(a)\varphi(s,a) = \lambda_\varphi(a,s)\varphi(s,a)$ and where $\lambda_\varphi(s,a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. Concludingly, a sufficient condition for our criteria to hold in terms of operators on Hilbert spaces is that the Koopman operator $\mathcal{K}(a)$ is self-adjoint i.e. that

$$\mathcal{K}(a) = \mathcal{K}(a)^* \quad . \tag{23}$$

Proof of Lemma 3.5: We aim to show that the map $\hat{\phi} : \bar{\Sigma} \times \mathcal{K}(\mathcal{S}) \times \{U(a)\}_{a\in\mathcal{A}} \to \mathcal{K}(\mathcal{S})$ given by

$$\big(\sigma_a \hat{*} g\big)(s) \longmapsto \Big(U^{-1}(a)\big(\sigma \bar{*} (U(a)g)\big)\Big)(s) = g\big(\sigma^{-1}\cdot s\big) \quad ,$$

defines a group action on the Koopman space of observables $\mathcal{K}(\mathcal{S})$. Where $\bar{*}$ is defined analog to Lemma 3.3 but acting on $\mathcal{F}(\mathcal{S}\times\mathcal{A})$ instead of $\mathcal{K}(\mathcal{S}\times\mathcal{A})$ by $(\sigma\bar{*}\varphi)(s,a) \longmapsto \varphi(\sigma^{-1}\cdot s, a)$. First of all note that

$$\Big(U^{-1}(a)\big(\bar{\sigma} * (U(a)g)\big)\Big)(s) = U^{-1}(a)\big(\bar{\sigma} * \varphi(s,a)\big) = U^{-1}(a)\varphi\big(\sigma^{-1}\cdot s, a\big) = g\big(\sigma^{-1}\cdot s\big) \tag{24}$$

We proceed analogously as in the proof of Lemma 3.3 above. Firstly, let $\sigma_0 \in \bar{\Sigma}$ and $g \in \mathcal{K}(\mathcal{S})$ be the identity element then on infers from Eq. (24) that it provides existence of an identity element of $\hat{\phi}$ by

$$(\sigma_{a,0}\hat{*}g)(s) = g\big(\sigma_0^{-1}\cdot s\big) = g(s) \quad ,$$

where we have used the notation

$$(\sigma_{a,i}\hat{*}g)(s) = \Big(U^{-1}(a)\big(\sigma_i\bar{*}(U(a)g)\big)\Big)(s) \ , \quad \text{for} \ \ i = 0, \dots \quad .$$

Secondly, let $\sigma_1, \sigma_2 \in \bar{\Sigma}$ and $\odot$ denoting the group operation i.e. $\sigma_1 \odot \sigma_2 = \sigma_3 \in \bar{\Sigma}$. Then

$$\begin{aligned}\Big(\sigma_{a,2}\hat{*}\big(\sigma_{a,1}\hat{*}g\big)\Big)(s) \quad &= \sigma_{a,2}\hat{*}\Big(g\big(\sigma_1^{-1}\cdot s\big)\Big) = g\big(\sigma_2^{-1}\cdot(\sigma_1^{-1}\cdot s)\big) = g\big((\sigma_2^{-1}\odot\sigma_1^{-1})\cdot s\big) \\ &\overset{(I)}{=} g\big((\sigma_1\odot\sigma_2)^{-1}\cdot s\big) = g\big(\sigma_3^{-1}\cdot s\big) = \Big(\sigma_{a,3}\bar{*}g\Big)(s) \quad ,\end{aligned}$$

where in (I) we have used the invertibility of the group property of Σ. Lastly, it follows analogously that for $\sigma, \sigma^{-1} \in \bar{\Sigma}$ that

$$\Big(\sigma\hat{*}\big(\sigma^{-1}\hat{*}g\big)\Big)(s) = \Big(\big(\sigma^{-1}\odot\sigma\big)\hat{*}g\Big)(s) = \Big(\sigma_0\hat{*}g\Big)(s) = g(s) \quad .$$

Thus the existence of an inverse is established which concludes to show the group property of $\hat{\phi}$.

Proof of Theorem 3.6: We aim to show twofold.

$\underline{\Longrightarrow}$**:** Firstly, that with $s_{t+1} = \tilde{F}(s_t, a_t)$ be a $\hat{\Sigma}$-action-equivariant control system with a symmetry action as in Lemma 3.5 which furthermore admits a Koopman operator representation as

$$g(s_{t+1}) = \mathcal{K}(a)g(s_t) \ , \quad \text{for} \ a \in \mathcal{A} \, , \ \forall g \in \mathcal{K}(\mathcal{S}) \quad .$$

Then

$$\Big[\sigma_a\hat{*}\big(\mathcal{K}(a)g\big)\Big](s) = \Big[\mathcal{K}(a)\ \big(\sigma_a\hat{*}g\big)\Big](s) \quad .$$

$\underline{\Longleftarrow}$**:** Secondly, the converse. Namely, if a control system $s_{t+1} = \tilde{F}(s_t, a_t)$ obeys Eqs. (14) and (15), then it is $\hat{\Sigma}$-action-equivariant, i.e. $\sigma \cdot \tilde{F}(s,a) = \tilde{F}(\sigma\cdot(s,a))$.

Let us start with the first implication i.e. $\Longrightarrow$.

First of all note that by the definition the Koopman operator it obeys

$$\big[\mathcal{K}(a)g\big](s) = g\big(\tilde{F}(s,a)\big) \quad .$$

Using the latter one infers that

$$\begin{aligned}
\Big[\sigma_a \hat{*}\big(\mathcal{K}(a)g\big)\Big](s) &= \sigma_a \hat{*} g(\tilde{F}(s,a)) \ , \\
&= U^{-1}(a)\Big(\sigma \bar{*}\big(U(a)g\big(\tilde{F}(s,a)\big)\big)\Big) \ , \\
&= U^{-1}(a)\Big(\sigma \bar{*}\varphi\big(\tilde{F}(s,a),a\big)\Big) \ , \\
&= U^{-1}(a)\Big(\varphi\big(\sigma^{-1}\cdot\tilde{F}(s,a),a\big)\big)\Big) \ , \\
&= U^{-1}(a)\Big(\varphi\big(\tilde{F}\big(\sigma^{-1}\cdot s,a\big),a\big)\big)\Big) \ , \\
&= g\big(\tilde{F}\big(\sigma^{-1}s,a\big)\big) \ .
\end{aligned}$$

Moreover, one derives that

$$\begin{aligned}
g\big(\tilde{F}\big(\sigma^{-1}s,a\big)\big) &= \mathcal{K}(a)g\big(\sigma^{-1}s\big) \ , \\
&= \mathcal{K}(a)\ \underbrace{U^{-1}(a)\,U(a)}_{=id}\,g\big(\sigma^{-1}s\big) \ , \\
&= \mathcal{K}(a)\,U^{-1}(a)\,\varphi\big(\sigma^{-1}s,a\big) \ , \\
&= \mathcal{K}(a)\Big(\,U^{-1}(a)\,\big(\sigma\bar{*}\varphi\big(s,a\big)\big)\Big) \ , \\
&= \mathcal{K}(a)\Big(\,U^{-1}(a)\,\big(\sigma\bar{*}\big(U(a)g(s)\big)\big)\Big) \ , \\
&= \Big[\mathcal{K}(a)\,\big(\sigma_a\hat{*}g\big)\Big](s) \ ,
\end{aligned}$$

which concludes the proof of the first part of the theorem. Let us next show the converse implication i.e. $\Longleftarrow$. For this case it is practical to use the discrete time-system notation explicitly. Let $\sigma \in \Sigma$ be a symmetry of the state-space and be $\tilde{s}_t = \sigma \cdot s_t$ and $\tilde{s}_{t+1} = \sigma \cdot s_{t+1}$ the σ-shifted states. Then

$$\begin{aligned}
g\big(s_{t+1}\big) &= g\big(\tilde{F}(s_t,a_t)\big) \ , \\
&= \mathcal{K}(a_t)\,g(s_t) \ , \\
&= \mathcal{K}(a_t)\,g\big(\sigma^{-1}\cdot\tilde{s}_t\big) \ , \\
&= \mathcal{K}(a_t)\Big(\sigma_a\hat{*}g\big(\tilde{s}_t\big)\Big) \ , \\
&= \Big[\mathcal{K}(a_t)\big(\sigma_a\hat{*}g\big)\Big](\tilde{s}_t) \ , \\
&\overset{(I)}{=} \Big[\sigma_a\hat{*}\big(\mathcal{K}(a_t)g\big)\Big](\tilde{s}_t) \ .
\end{aligned}$$

Thus in particular

$$\sigma_a^{-1}\hat{*}g\big(s_{t+1}\big) = \Big[\sigma_a^{-1}\hat{*}\big(\sigma_a\hat{*}\big(\mathcal{K}(a_t)g\big)\big)\Big](\tilde{s}_t) \overset{Lemma\ 3.5}{=} \Big[\mathcal{K}(a_t)g\Big](\tilde{s}_t) \ ,$$

where in (I) we have used that the symmetry operator commutes with the Koopman operator. Moreover, one finds that

$$\sigma_a^{-1}\hat{*}g\big(s_{t+1}\big) = g\Big(\big(\sigma^{-1}\big)^{-1}\cdot s_{t+1}\Big) = g\big(\sigma\cdot s_{t+1}\big) = g\big(\tilde{s}_{t+1}\big) \ ,$$

from which one concludes that

$$g(\tilde{s}_{t+1}) = \big[\mathcal{K}(a_t)g\big](\tilde{s}_t) = g\big(\tilde{F}(\tilde{s}_t,a_t)\big) \ .$$

Finally, we us the invertibility of the Koopman space observables i.e. $g^{-1}(g(s)) = s$ to infer

$$\tilde{s}_{t+1} = \tilde{F}(\tilde{s}_t, a_t) \ .$$

Thus

$$\tilde{s}_{t+1} = \sigma \cdot s_{t+1} \ \Rightarrow \ \tilde{F}(\tilde{s}_t, a_t) = \sigma \cdot \tilde{F}(s_t, a_t) \ \Rightarrow \ \tilde{F}(\sigma \cdot s_t, a_t) = \sigma \cdot \tilde{F}(s_t, a_t) \ , \tag{25}$$

which at last concludes our proof of the second part of the theorem.

Proof of Theorem 3.7: Let $s_{t+1} = \tilde{F}(s_t, a_t)$ be a $\hat{\Sigma}$-action-equivariant control system as in Theorem 3.6. For symmetry maps

$$\tilde{s}_t = D\Big(\big(\mathbb{1} \ + \ \epsilon\, \sigma_{a_t}\big) \hat{*} E(s_t) \Big) \ , \ \ \tilde{s}_{t+1} = D\Big(\big(\mathbb{1} \ + \ \epsilon\, \sigma_{a_t}\big) \hat{*} E(s_{t+1}) \Big) \ . \tag{26}$$

we aim to show that one can use a symmetry transformation to shift both s_t as well as s_{t+1}

$$\sigma^{\epsilon}_{a_t} : \ \big(s_t, s_{t+1}, a_t\big) \ \longmapsto \ \big(\tilde{s}_t, \tilde{s}_{t+1}, a_t\big) \ , \ \text{ s.t. } \ \tilde{s}_{t+1} = \tilde{F}(\tilde{s}_t, a_t) + \mathcal{O}(\epsilon) \ , \tag{27}$$

where $\epsilon \ll 1$. Thus the symmetry shift of the data-point reflects the real dynamic of the system up to an controllable error. By definition of the Koopman operator one has

$$\tilde{s}_{t+1} = D\Big(\mathcal{K}(a_t) E(\tilde{s}_t) \Big) = D\Big(\mathcal{K}(a_t) \underbrace{E(D(}_{=\mathbb{1}} \sigma_{a_t} \hat{*} E(s_t))) \Big) = D\Big(\mathcal{K}(a_t) \sigma_{a_t} \hat{*} E(s_t) \Big) \ . \tag{28}$$

By using that $[\sigma_{a_t}, \mathcal{K}(a_t)] = 0$ one infers that

$$\tilde{s}_{t+1} = D\Big(\sigma_{a_t} \hat{*} \mathcal{K}(a_t) E(s_t) \Big) \ . \tag{29}$$

Let us next use the fact that in the theorem $\sigma_{a_t} \to \big(\mathbb{1} \ + \ \epsilon\, \sigma_{a_t}\big)$ admits an expansion in the parameter ϵ. For $\epsilon \ll 1$ one can Taylor expand the differentiable functions D, E to find that

$$\tilde{s}_t = s_t + \delta(\epsilon) \ , \ \text{ with } \ \delta(\epsilon) = \epsilon \sum_I g^{\sigma}_I \frac{\partial D}{\partial g_I}\Big|_{E(s_t)} + \mathcal{O}(\epsilon^2) \ , \tag{30}$$

where $g^{\sigma} = \sigma_{a_t} \hat{*} E(s_t)$ and $g = E(s_t)$, and with the index $I = 1, \ldots, N$. Next let us assume that the action is given by a differentiable function of the states $a_t = a_t(s_t)$.[6] Then one may Taylor expand the Koopman operator using equation 30 as[7]

$$K(a_t(\tilde{s}_t)) = K(a_t(s_t)) + \sum_i \sum_n \delta_n(\epsilon)\, a'_{in}(s_t)\, \mathcal{K}_i = K(a_t(s_t)) + \epsilon\, \Delta + \mathcal{O}(\epsilon^2) \ . \tag{31}$$

where and $a'_{in}(s_t) = \frac{\partial a_t}{\partial_{s_{t,n}}}|_{s_t}$ and $n = 1, \ldots, dim(s_t)$ and with

$$\Delta = \sum_i \sum_n \sum_I g^{\sigma}_I \frac{\partial D_n}{\partial g_I}\Big|_{E(s_t)} a'_{in}(s_t)\, \mathcal{K}_i \ . \tag{32}$$

Combining equation 29 and equation 31 one infers that

$$\begin{aligned} \tilde{s}_{t+1} = D\Big(\sigma_{a_t} \hat{*} \mathcal{K}(a_t(\tilde{s}_t)) E(s_t) \Big) &= D\Big(\sigma_{a_t} \hat{*} \big(K(a_t(s_t)) + \epsilon\, \Delta\big) E(s_t) \Big) \\ &= D\Big(\sigma_{a_t} \hat{*} K(a_t(s_t)) E(s_t) + \epsilon\, \sigma_{a_t} \hat{*}\, \Delta\, E(s_t) \Big) \\ &= D\Big(\sigma_{a_t} \hat{*} E(s_{t+1}) + \epsilon\, \underbrace{\sigma_{a_t} \hat{*}\, \Delta\, E(s_t)}_{:= g^{\Delta,\sigma}} \Big) \ . \end{aligned} \tag{33}$$

[6] Although neural networks modeling the policies used for data-collection contain non-differentiable components they are differentiable almost everywhere.

[7] The reader may wonder about the explicit use of the index $\delta_n(\epsilon)$ expressing the dimension of the state. it is necessary here although we have simply used s_t as an implicit vector without explicitly specifying any components throughout the work.

Now one may Taylor expand D making use of $\epsilon \ll 1$ to arrive at

$$\tilde{s}_{t+1} = D\big(\, \sigma_{a_t} \hat{*} E(s_{t+1})\big) \, + \, \epsilon \sum_I g_I^{\Delta,\sigma} \frac{\partial D}{\partial g_I}\Big|_{\sigma_{a_t} \hat{*} E(s_{t+1})} + \mathcal{O}(\epsilon^2) \tag{34}$$

This concludes the proof of theorem 3.7. One concludes from equation 34 that the error made at order ϵ is

$$\sum_I g_I^{\Delta,\sigma} \frac{\partial D}{\partial g_I}\Big|_{\sigma_{a_t} \hat{*} E(s_{t+1})} \quad . \tag{35}$$

Let us note that the proof for state shifts arising from a sum of symmetry transformations as

$$\tilde{s}_t = D\Big(\, (\mathbb{1} \, + \, \sum_{I=1}^{N} \epsilon_I \, \sigma_{a_t}^I) \hat{*} E(s_t)\Big) \; , \;\; \tilde{s}_{t+1} = D\Big(\, (\mathbb{1} \, + \, \sum_{I=1}^{N} \epsilon_I \, \sigma_{a_t}^I) \hat{*} E(s_{t+1})\Big) \quad . \tag{36}$$

is completely analogous, and an analogous theorem holds. We choose the number of symmetries to run over the dimension of the Koopman latent space N as it is relevant for the **KFC++** setup.

B Implementation details

The Koopman forward model The KFC algorithm requires pre-training of a Koopman forward model $\mathcal{F} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ as

$$\mathcal{F}^c(s_t, a_t) = \begin{cases} D\big(E(s_t)\big) = s_t & \text{if } c = 0 \text{ corresponds to a VAE.} \\ D\Big(\, (\mathcal{K}_0 + \sum_{i=1}^m \mathcal{K}_i \, a_{t,i}) E(s_t)\Big) = s_{t+1} & \text{if } c = 1\text{: forward prediction model.} \end{cases} \tag{37}$$

where both of E and D are approximated by Multi-Layer-Perceptrons (MLP's) and the bilinear Koopman-space operator approximation are implemented by a single fully connected layer for $\mathcal{K}_{i=0,\dots,m}$, respectively. The model is trained on batches of the offline data-set tuples (s_{t+1}, s_t, a_t) and optimized via the loss-function

$$\bar{l}_2(\mathcal{F}^1(s_t, a_t) \, , \, s_{t+1}) \; + \; \gamma_2 \, \bar{l}_2(\mathcal{F}^0(s_t + s_\epsilon, a_t) \, , \, s_t + s_\epsilon) \quad , \tag{38}$$

where s_ϵ is a normally distributed random sate vector shift with zero mean and a standard deviation of $6 \cdot 10^{-2}$ where the $\bar{l}_2$-loss is the Huberloss. For training the Koopman forward model in Eq. (21) we split the dataset in a randomly selected training and validation set with ratios $70\%/30\%$. We then loop over the dataset to generate the required symmetry transformations and extend the replay buffer dataset with the latter according to Eq. (19) as

$$\begin{aligned} \text{I}: \quad & \big(s_t, s_{t+1}, a_t, r_t\big) \overset{symmetries}{\longmapsto} \big(s_t, s_{t+1}, a_t, r_t, \sigma_{a_t}\big) \quad , \\ \text{II}: \quad & \big(s_t, s_{t+1}, a_t, r_t\big) \overset{symmetries}{\longmapsto} \big(s_t, s_{t+1}, a_t, r_t, U(a_t), U(a_t)^{-1}\big) \quad . \end{aligned} \tag{39}$$

It is more economic to compute the symmetry transformation as a pre-processing step rather than at runtime of the RL algorithm.

Let us reemphasize a couple of points addressed already in the main text to make this section self-contained. Note that we have built upon the implementation of CQL (Kumar et al., 2020), which is based on SAC (Haarnoja et al., 2018). **We also use the same hyperparameters for all the experiments as presented in the CQL and S4RL papers for baselines.**

Koopman forward VAE-model: For the encoder as well as the decoder we choose a three layer MLP architecture, with hidden-dimensions $[512, 512, N]$ and $[512, 512, m]$, respectively. Where $N = 32$ is the Koopman latent space dimension and m is the state space dimensions following our notation from section 2. The activation functions are ReLU (Nair & Hinton, 2010). We use an ADAM optimizer with learning rate $3 \cdot 10^{-4}$. We train the model for 75 epochs. The batch size is chosen to be 256. Lastly, we employ a random normally distributed state-augmentation for the VAE training i.e. $s \to s + s_\epsilon$ with $s_\epsilon \in \mathcal{N}(0, 1 \cdot 10^{-2})$.

Hyper-parameters KFC: We take over the hyper-parameter settings from the CQL paper (Kumar et al., 2020) except for the fact that we do not use automatic entropy tuning of the policy optimisation

step Eq. (3) but instead a fixed value of $\alpha = 0.2$. The remaining hyper-parameters of the conservative Q-learning algorithm are as follows: $\gamma = 0.99$ and $\tau = 5 \cdot 10^{-3}$ for the discount factor and target smoothing coefficient, respectively. Moreover, the policy learning rate is $1 \cdot 10^{-4}$ and the value function learning rate is $3 \cdot 10^{-4}$ for the ADAM optimizers. We enable Lagrange training for $\tilde{\alpha}$ with threshold of 10.0 and learning rate of $3 \cdot 10^{-4}$ where the optimisation is done by an ADAM optimizer. The minimum Q-weight is set to 10.0 and the number of random samples of the Q-minimizer is 10. For more specifics on these quantities see (Kumar et al., 2020). The model architectures of the policy as well as the value function are three layer MLP's with hidden dimension 256 and $ReLU$-activation. The batch size is chosen to be 256. Moreover, the algorithm performs behavioral cloning for the first $40k$ training-steps, i.e. time-steps in an online RL notation.

The KFC specific choices are as follows: the split of random to Koopman-symmetry state shifts is $20/80$ i.e. $p_K = 80\%$, whereas the option for the symmetry-type is either "Eigenspace" or "Sylvester" referring to case (I) and (II) of Eq. (19), respectively. Regarding the random variables in the symmetry transformation generation process, we choose $\epsilon_i \in \mathcal{N}(0, 1.5 \cdot 10^{-3})$ for case (II) and $\epsilon \in \mathcal{N}(0, 5 \cdot 10^{-7})$ for case (I) in Eq. (19) after normalizing σ_a by its mean. While for the random shift we use $\tilde{\epsilon} \in \mathcal{N}(0, 3 \cdot 10^{-3})$ which is the hyper-parameter used in (Sinha et al., 2021).

Computational setup: We performed the empirical experiments on a system with PyTorch 1.9.0a (Paszke et al., 2019) The hardware was as follows: NVIDIA DGX-2 with 16 V100 GPUs and 96 cores of Intel(R) Xeon(R) Platinum 8168 CPUs and NVIDIA DGX-1 with 8 A100 GPUs with 80 cores of Intel(R) Xeon(R) E5-2698 v4 CPUs. The models are trained on a single V100 or A100 GPU.

C Ablation Study

Task Name	**S4RL-($\mathcal{N}$)**	**KFC**	**KFC++**	**KFC++-contact**
cheetah-random	**52.3**	48.6	49.2	49.4
cheetah-medium	48.8	55.9	59.1	**61.4**
cheetah-medium-replay	51.4	**58.1**	**58.9**	**59.3**
cheetah-medium-expert	**79.0**	**79.9**	**79.8**	**79.8**
hopper-random	**10.8**	**10.4**	**10.7**	**10.7**
hopper-medium	78.9	90.6	**94.2**	**95.0**
hopper-medium-replay	35.4	**48.6**	**49.0**	**49.1**
hopper-medium-expert	113.5	121.0	125.5	**129**
walker-random	**24.9**	19.1	17.6	18.3
walker-medium	93.6	102.1	**108.0**	**108.3**
walker-medium-replay	30.3	**48.0**	46.1	**48.5**
walker-medium-expert	112.2	114.0	**115**	**118.1**

Table 2: We study the effect of combining S4RL-$\mathcal{N}$ based augmentation training on "contact" event transitions (state transitions where the agent makes contact with a surface, read C.1 for more details), and KFC++ augmentation training on non-"contact" events. We see that KFC++-contact performs similarly and in most cases, slightly better than the KFC++ baseline, which is expected. We experiment with the Open AI gym subset of the D4RL tasks and report the mean normalized returns over 5 random seeds.

C.1 Training on non contact events

To test the theoretical limitations in a practical setting one interesting experiment is to use the S4RL-$\mathcal{N}$ augmentation scheme when the current state is a "contact" state and to choose KFC++ otherwise. We define a "contact" state as one where the agent makes contact with another surface; in practice we looked at the state dimensions which characterize the velocity of the agent parts, and if there is a sign difference between s_t and s_{t+1}, then contact with a surface must have been made. The details regarding which dimension maps to agent limb velocities is available in the official implementation of the Open AI Gym suite (Brockman et al., 2016).

Note that the theoretical limitations are twofold, firstly the theorems 3.6 and 3.7 only hold for dynamical systems with differentiable state-transitions; secondly, we employ a Bilinearisation Ansatz

for the Koopman operator. Contact events lead to non-continuous state transitions and moreover would require incorporating for external forces in our Koopman description, thus both theoretical assumptions are practically violated in our main empirical evaluations of KFC and KFC++. By performing this ablation study which is more suitable for our Ansatz, i.e. the KFC++ part is only trained on "differentiable" trajectories without contact events. In other words, the Koopman description is unlikely to give good information regarding s_{t+1} if there is a "contact event", we simply use S4RL-$\mathcal{N}$ when training the policies, which we expect to give slightly better performance than the KFC++ baseline.

We report the results in Table 2, where we perform results on the OpenAI Gym subset of the D4RL benchmark test suite. We denote the proposed scheme as **KFC++-contact** and compare to both S4RL-$\mathcal{N}$ and KFC++. We see that KFC++-contact performs comparably and slightly better in some instances, compared to KFC++. This is expected as contact events may be rare, or the data augmentation given by the KFC++ model during contact events is pseudo-random and itself is normally distributed, which may suggest that KFC++ behaves similarly to KFC++-contact. However, we do not say this conclusively.

C.2 Magnitude of KFC augmentation

To further try to understand the effect of using KFC for guiding data augmentation, we measure the mean L2-norm of the strength of the augmentation during training. We measured the mean L2-norm between the original state and the augmented state, and found the distance to be 2.6×10^{-4}. For comparison, S4RL uses a fixed magnitude of 1×10^{-4} (Sinha et al., 2021) for all experiments with S4RL-Adv. Since S4RL-Adv does a gradient ascent step towards the direction with the highest change, its reasonable for the magnitude of the step to be smaller in comparison. An adversarial step too large may be detrimental to training as the step can be too strong. KFC minimizes the need for making such a hyperparameter search decision.

C.3 Prediction Model

As a further ablation study it is of interest to compare the symmetry induced state shifts tuple $(\tilde{s}_t, \tilde{s}_{t+1})$ to one obtained by the forward prediction of our VAE-forward model equation 21. In particular we use the forward prediction on $\tilde{s}_t$ obtained by the KFC++ shift, i.e. $\tilde{s}_{t+1} = \mathcal{F}^{c=1}(\tilde{s}_t, a_t)$. We refer to this setup in the following as **KFC++prediction**. See table 3 for the results. We conclude that

Task Name	KFC	KFC++	KFC++-prediction
cheetah-random	48.6	**49.2**	46.5
cheetah-medium	55.9	**59.1**	53.7
cheetah-medium-replay	**58.1**	**58.9**	55.3
cheetah-medium-expert	**79.9**	**79.8**	76.3
hopper-random	**10.4**	**10.7**	**10.8**
hopper-medium	90.6	**94.2**	90.5
hopper-medium-replay	**48.6**	**49.0**	44.2
hopper-medium-expert	121.0	**125.5**	121.2
walker-random	**19.1**	17.6	15.6
walker-medium	102.1	**108.0**	105.3
walker-medium-replay	**48.0**	46.1	45.2
walker-medium-expert	114.0	**115.6**	114.5

Table 3: Results with prediction model on the Open AI Gym subset of the D4RL tasks. We report the mean normalized episodic rewards over 5 random seeds similar to the original D4RL paper Fu et al. (2021).

KFC++prediction falls behind both **KFC** as well as **KFC++**. This was expected as the symmetry shift in the latter alters the original data points by means of the VAE which admits not only a much higher accuracy bu also advanced generalisation capabilities to out of distribution values.

C.4 DISCUSSION

Although our KFC framework is suited best for the description of environments described by control dynamical systems the learned (in self-supervised manner) linear Koopman latent space representation may be applicable to a much wider set of tasks. However, there are notable shortcomings to the current implementation both conceptually as well as practically. The bilinearisation in Eq. (21) of the latent space theoretically assumes the dynamical system to be governed by Eq. (8), which is rather restrictive. Although a general non-affine control system admits a bilinearisation (Brunton et al., 2021) it generically requires the observables to depend on the action-space variables implicitly. Secondly, the Koopman operator formalism is theoretically defined by its action on an infinite dimensional observable space. The finite-dimensional approximation i.e. the latent space representation of the Koopman forward model in Eq. (21) lacks accuracy due to that. On the practical side our formalism requires data pre-processing which is computationally expensive, i.e. solving the Sylvester or eigenvalue problem for every data-point. Moreover, the Koopman forward model in Eq. (21) serves as a function approximation to two distinct tasks. Thus one faces a twofold accuracy vs. over-estimation problem which needs to be balanced. The systematic error in the VAE directly imprints itself on the state-data shift in Eqs. (16) and (19) and may thus conceal any potential benefit of the symmetry considerations. Lastly, the dynamical symmetries do not infer the reward. Thus the underlying working assumption is that the reward should not vary much in Eq. (16).