Deep Learning of Koopman Representation for Control

Yiqiang Han*, Wenjian Hao*, and Umesh Vaidya

Abstract— We develop a data-driven, model-free approach for the optimal control of the dynamical system. The proposed approach relies on the Deep Neural Network (DNN) based learning of Koopman operator for the purpose of control. In particular, DNN is employed for the data-driven identification of basis function used in the linear lifting of nonlinear control system dynamics. The controller synthesis is purely data-driven and does not rely on a priori domain knowledge. The OpenAI Gym environment, employed for Reinforcement Learning-based control design, is used for data generation and learning of Koopman operator in control setting. The method is applied to two classic dynamical systems on OpenAI Gym environment to demonstrate the capability.

I. INTRODUCTION

The problem of data-driven control of dynamic system is challenging with applications in robotics, manufacturing system, autonomous vehicles, and transportation networks. There is a shift from model-based to data-driven control paradigm with the increasing complexity of engineered systems and easy access to a large amount of sensor data. Deep reinforcement learning consisting of deploying deep neural networks for the learning of optimal control policy is emerging as one of the powerful tools for data-driven control of dynamic systems [1]. The controller synthesis for dynamic systems in the model-based and model-free setting has a long history in control theory literature. Furthermore, the problem of control design for a system with nonlinear dynamics is recognized to be a particularly challenging problem. More recently, the use of linear operators from the dynamical systems has offered some promises towards providing a systematic approach for nonlinear control [2]–[12].

However, some challenges need to overcome for the successful extension of linear operator-based analysis methods to address the synthesis problem. The basic idea behind the linear operator framework involving Koopman is to lift the finite-dimensional nonlinear dynamics to infinite-dimensional linear dynamics. The finite-dimensional approximation of the linear operator is obtained using the time series data of the dynamic system.

One of the main challenges is the selection of appropriate choice of finite basis functions used in the finite but high dimensional linear lifting of nonlinear dynamics. The standard procedure is to make a-priori choice of basis function such as radial basis function or polynomial function for the finite approximation of the Koopman operator. However, such approach does not stand to take advantage of the known physics or the data in the choice of the basis function. An alternate approach based on the power of Deep Neural Network (DNN) could be used for the data-driven identification of the appropriate basis function in the autonomous (uncontrolled) setting [13], [14]. The conjecture is that the physical model can be retrieved from data observations and DNN are efficient in exploiting the power of the data. In this paper, we extend the application of DNN for the selection of basis functions in a controlled dynamical system setting.

The main contributions of the paper are as follows. We extend the use of DNN for the finite dimensional representation of Koopman operator in controlled dynamical system setting. Our data-driven learning algorithm can automatically search for a rich set of suitable basis functions to construct the approximated linear model in the lifted space. The OpenAI Gym environment is employed for data generation and training of DNN to learn the basis functions [15]. Example systems from OpenAI Gym environment employed for design of reinforcement learning control is used to demonstrate the capability of the developed framework. This work is on the frontier to bridge the understanding of model-based control and model-free control (e.g. Reinforcement Learning) from other domains.

II. KOOPMAN LEARNING USING DEEP NEURAL NETWORK

Consider a discrete-time, nonlinear dynamical system along with a set of observation data of length T:

$$x_{t+1} = F(x_t, u_t)$$

$$\boldsymbol{X}=[x_1,x_2,...,x_T], \quad \boldsymbol{Y}=[y_1,y_2,...,y_T], \quad \boldsymbol{U}=[u_1,u_2,...,u_T] \tag{1}$$

where, $x_t \in \mathbb{R}^n$, $u_t \in \mathbb{R}^m$ is the state and control input respectively, and $y_t = x_{t+1}$ for $t = 1,\ldots,K$. We use the capital letter X to represent the collection of all data points in training set. The objective is to provide a linear lifting of the nonlinear dynamical system for the purpose of control. The Koopman operator ($\mathcal{K}$) is an infinite-dimensional linear operator defined on the space of functions as follows:

$$[\mathcal{K}g](x) = g \circ F(x, u) \tag{2}$$

For more details on the theory of Koopman operator in dynamical system and control setting refer to [16]–[19]. Related work on extending Koopman operator methods to controlled
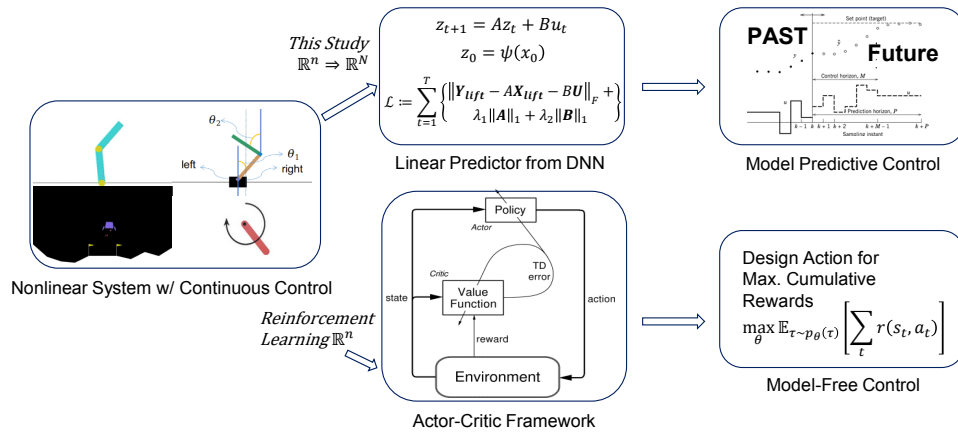
Fig. 1. Deep Koopman Representation for Control (DKRC) framework with comparison to Reinforcement Learning

dynamical systems can be found in References [7], [9], [20]–[24]. While the Koopman based lifting for autonomous system is well understood and unique, the control dynamical system can be lifted in different ways. In particular, for control affine system the lifting in the functional space will be bi-linear [24], [25]. Given the computational focus of this paper we resort to linear lifting of control system. The linear lifting also has the advantage of using state of the art control methodologies such as linear quadratic regulator (LQR) and Model Predictive Control (MPC) from linear system theory. The linear lifting of nonlinear control system is given by

$$\Psi(x_{t+1}) = A\Psi(x_t) + Bu_t \qquad (3)$$

where $\Psi(x) = [\psi_1(x), \ldots, \psi_N(x)]^\top \in \mathbb{R}^{N \gg n}$ is the state or the basis in the lifted space. The matrix $A \in \mathbb{R}^{N \times N}$ and $B \in \mathbb{R}^{N \times m}$ are the finite dimensional approximation of the Koopman operator for control system in the lifted space. One of the main challenges in the linear lifting of the nonlinear dynamics is the proper choice of basis functions used for lifting. While the use of radial basis function, polynomials, and kernel functions is most common, the choice of appropriate basis function is still open. Lack of systematic approach for the selection of appropriate basis has motivated the use of Deep Neural Network (DNN) for data-driven learning of basis function for control.

The planned control inputs considered in this study are in continuous space and later will be compared to state-of-art model-free learning methods such as Reinforcement Learning. An overview of the scope of this work is shown in Figure 1. Both the two methods (our Deep Koopman Representation for Control (DKRC) approach and Reinforcement Learning) apply a data-driven model-free learning approach to learn the dynamical system. However, our approach seeks a linear representation of nonlinear dynamics and then design the control using a model-based approach, which will be discussed in later sections.

To enable deep Koopman representation for control, the first step is to use the DNN approach to approximate

Koopman operator for control purposes. Our method seeks a multi-layer deep neural network to automate the process of finding the basis function for the linear lifting of nonlinear control dynamics via Koopman operator theory. Each feed-forward path within this neural network can be regarded as a nonlinear basis function ($\psi_i : \mathbb{R}^n \rightarrow \mathbb{R}$), whereas the DNN as a collection of basis functions can be written as $\Psi : \mathbb{R}^n \rightarrow \mathbb{R}^N$. Each output neuron is considered as a compounded result from all input neurons. Therefore, the mapping ensures that our Koopman operator maps functions of state space to functions of state space, not states to states.

Unlike existing methods used for the approximation of the Koopman operator, where the basis functions are assumed to be known apriori, the optimization problem used in the approximation of Koopman operator using DNN is non-convex. The non-convexity arises due to simultaneous identification of the basis functions and the linear system matrices.

During the training of DNN, the observables in state-space (x) are segmented into data pairs to guide the automated training process, whereas the output from the DNN is the representation of the lifting basis function for all data pairs ($\psi(x)$). This method is completely data-driven, and basis functions are learned directly from training datasets.

The input dataset is split into three sets, whereas 70% of the data are used as training samples, 15% as validation, and another 15% as testing cases. For the problems in this study, we choose a Multilayer Perceptron (MLP) module with four hidden layers to be used in between the neural network's input and output layers. At each layer in between the input and output layers, we apply hyperbolic-tangent (tanh) non-linear activation functions with biases at each neuron. In this study, all training processes are done using ADAM optimizer [26] for neural networks parameters' backpropagation based on Pytorch 1.4 platform with NVIDIA V100 GPUs on an NVIDIA DGX-2 GPU supercomputer cluster. The deployment of the code has been tested on personal computers with NVIDIA GTX 10-series GPU cards.

During training, the DNN will iterate through epochs to minimize a loss term, where differences between two

Authorized licensed use limited to: Korea Advanced Inst of Science & Tech - KAIST. Downloaded on January 14,2023 at 10:15:59 UTC from IEEE Xplore. Restrictions apply.

elements in lifted data pairs will be recorded and minimized in batch. At the convergence, we obtain a DNN, which automatically finds optimal lifting functions to ensure both the Koopman transformation (Equation 1) and linearity (Equation 3) by tuning the DNN's loss function to a minimum. A schematic illustration of the Deep Koopman Representation learning can be found in Figure 2.
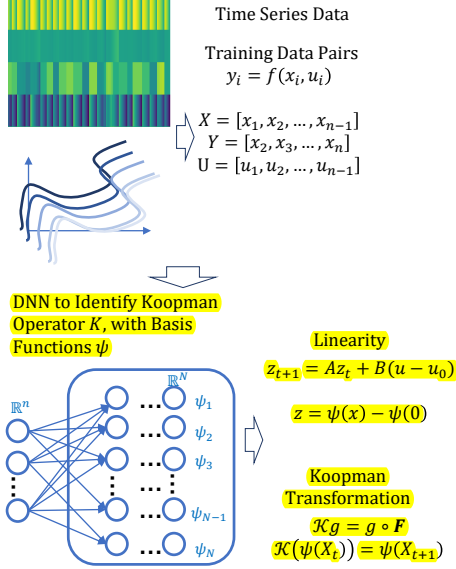

Fig. 2.   Koopman Operator Learning using Neural Network Scheme

The mapping from state-space to higher-dimensional space requires a modification to the existing control design governing equation, which is equivalent to regulation around a non-zero fixed point for non-linear systems. The affine transformation can be introduced without changing A and B matrices, as shown in Equation 4 below:

$$\mathbf{z} := \Psi(\boldsymbol{x}) - \Psi_0, \quad \boldsymbol{v} := \boldsymbol{u} - u_0 \qquad (4)$$

Here, we use Ψ_0 to represent this fixed point (in this case, goal position x=0) in the higher-dimensional space. To approximate u_0, it requires one additional auxiliary network for this constant. The network hyperparameters will then be trained together with the DNN.

After applying these changes, the Equation 3 becomes the final expression of the high-dimensional linear model of the system in lifted space:

$$\mathbf{z}_{t+1} = A\mathbf{z}_t + B\mathbf{v}_t \qquad (5)$$

Correspondingly, the problem we are solving becomes:

$$\min_{A,B,u_0} \sum_k \| \mathbf{z}_{t+1} - A\mathbf{z}_t - B(\mathbf{u}) - \mathbf{u_0} \|_F \qquad (6)$$

The problem we are dealing with is non-convex. There is no one-shot solution to solve for multiple unknowns in the problem. We propose the following way to solve for the lifting function, with assumed A and B values at the beginning. We start the training with randomly initialized

$A \in \mathbb{R}^{N \times N}$ and $B \in \mathbb{R}^{N \times m}$. The ψ functions are learned within the DNN training, whereas the matrices A and B are kept frozen during each training epoch.

In particular, the following loss function, $\mathcal{L}$ in Equation 7, is used in the training of DNN.

$$\mathcal{L} := \sum_{t=1}^{T} \| \mathbf{z}_{t+1} - A\mathbf{z}_t - B(\mathbf{u} - \mathbf{u_0}) \|_F \qquad (7)$$

At the end of each epoch, the A and B can be updated analytically by adding new information (with a learning rate of 0.5) based on the Equation 8 as follows:

$$[A, B] = \mathbf{z}_{t+1} \begin{bmatrix} \mathbf{z}_t \\ U \end{bmatrix} \left(\begin{bmatrix} \mathbf{z}_t & U \end{bmatrix} \begin{bmatrix} \mathbf{z}_t \\ U \end{bmatrix} \right)^{\dagger} \qquad (8)$$

where $\dagger$ denotes pseudo-inverse. In addition to finding system A and B matrix, following optimization problem is solved to compute the C matrix mapping states from lifted space back to state space.

$$\min_C \sum_t \| x_t - C\Psi(x_t) \|_F, \quad \text{s.t.} \quad C\Psi_0 = 0 \qquad (9)$$

III. Koopman-based Control

Based the insights gained from the linearized system in the higher-dimensional space, we can proceed to solve discrete LQR problem for the linear system with a quadratic cost:

$$J(V) = \sum_t z_t^\top C^\top QC z_t + v_t^\top R v_t \qquad (10)$$

Model Predictive Control (MPC) can also be implemented to accommodate problem constrains. Given a time horizon of L, we can solve for $V \in \mathbb{R}^{m \times L}$ by minimizing the following cost function, as illustrated in Equation 11.

$$J(V) = \sum_{t=0}^{L-1} (z_t^\top C^\top QC z_t + v_t^\top R v_t) + z_L^\top C^\top QC z_L$$
$$\text{subject to} \begin{cases} v_t \leq u_{max} - u_0 \\ v_t \geq u_m in - u_0 \end{cases} \qquad (11)$$

To summarize, the algorithm of Deep Koopman Representation for Control (DKRC) is listed in Algorithm 1, where $L1(\theta)$ ensures a high dimension linear system, $L2(\theta)$ handles the controllability of the system.

To interpret the control strategy generated by DKRC, we proceed with the spectral analysis of the finite dimensional approximation. In particular, the eigenfunctions of the Koopman operator are approximated using the identified A matrix.

The eigenfunctions learned from the neural network can be viewed as intrinsic coordinates for the lifted space and provide insights into the transformed dynamics in the higher dimensional space. We use a classic inverted pendulum problem to demonstrate in this section. The pendulum system setup can be found in Figure 3. We introduce a 3-D contour graph setting that uses the original state space observables (angular position & angular velocity of the pendulum) as the basis and plot the eigenfunction values in the third dimension

Algorithm 1 Deep Koopman Representation for Control (DKRC)

Input: observations: x, control: u

Output: Planned trajectory and optimal control inputs: (z_{plan}, v_{plan})

- Initialization
  1) Set goal position x^*
  2) Build Neural Network: $\psi_N(x_t; \theta)$
  3) Set $z(x_t; \theta) = \psi_N(x_t; \theta) - \psi_N(x^*; \theta)$
- Steps
  1) Set $K = z(x_{t+1}; \theta) * z(x_t; \theta)^\dagger$
  2) Set the first loss function $L1$
     $$L_1(\theta) = \frac{1}{L-1} \sum_{t=0}^{L-1} \| z(x_{t+1}; \theta) - K * z(x_t; \theta) \|$$
  3) Set the second loss function L_2
     $$[A, B] = \mathbf{z}_{t+1} \begin{bmatrix} \mathbf{z}_t \\ U \end{bmatrix} \left(\begin{bmatrix} \mathbf{z}_t & U \end{bmatrix} \begin{bmatrix} \mathbf{z}_t \\ U \end{bmatrix} \right)^\dagger$$
     $$L_2(\theta) = (N - rank(controllability(A, B))) + \|A\|_1 + \|B\|_1$$
  4) Train the neural network, updating the complete loss function
     $$L(\theta) = L_1(\theta) + L_2(\theta)$$
  5) After converging, We can get system identity matrices A, B, C
     $$C = \boldsymbol{X}_t * \psi_N(\boldsymbol{X}_t)^\dagger$$
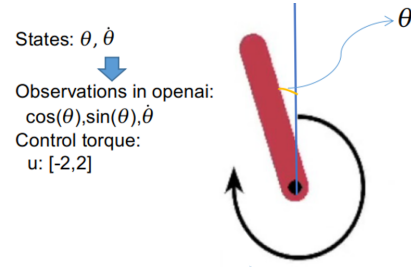 6) Apply LQR or MPC control with constraints
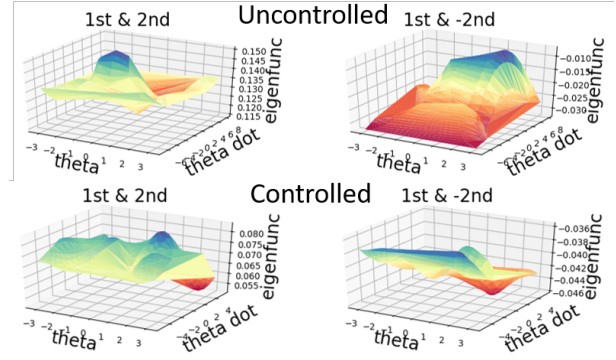

Fig. 3.   Inverted Pendulum

Fig. 4.   Eigenfunctions learned from controlled inverted pendulum case. Left: real part of eigenfunctions, Right: imaginary part of the eigenfunctions. The dominant eigenfunction has zero value imaginary part.

An example of 3d plots for eigenfunctions obtained from DNN can be seen in Figure 4. The two images from the first row represent the first dominant eigenfunctions for uncontrolled pendulum case, whereas the second row represents the controlled case.

As can be seen, the first dominant eigenfunctions from both the two cases exhibit a very similar double-pole structure. The uncovered intrinsic coordinates exhibit a trend that is related to the energy levels within the system as base physical coordinates change, i.e., $\dot{\theta}$ is positively related to the kinetic energy, whereas the θ is proportional to the potential energy. Towards the center of the 3d plot, both kinetic energy and potential energy exhibit declined absolute magnitudes. In the controlled case, torque at the joint is being added into the system's total energy count, which seeks a path to bring the energy of the system to the maximum in potential energy and lowest absolute value in kinetic energy. Due to the kinetic energy part ($\frac{1}{2}\dot{\theta}^2$) is dominating the numerical value of the energy, this effect has been reflected in the distribution of eigenfunction along the $\dot{\theta}$ axis. This comparison figure shows the feasibility of using eigenfunctions from learned Koopman operator to gain insight of the dynamical system, especially for the control purpose.

IV. SIMULATION RESULTS

We demonstrate the proposed continuous control method based on deep learning and Koopman operator (DKRC) on several example systems offered in OpenAI Gym [15]. In this paper, we deploy model predictive control(MPC) on Inverted Pendulum, and adopt LQR for Lunar Lander (Continuous Control).

A. Testing Environment

The OpenAI Gym utilizes a Box2D physics engine, which provides a dynamical system for the testing. Gravity, object interaction, and physical contact behavior can be specified in the world configuration file. The dynamics of the simulation environment are unknown to the model. The model needs to learn the dynamics through data collected during training. Each simulation game is randomly initialized to generate initial disturbance.

B. Reinforcement Learning

The OpenAI Gym provides open-source environments primarily for Reinforcement Learning (RL) algorithm development purposes. Although RL shares the same model-free learning basis with our proposed DKRC method, the training processes are different. DKRC requires segmentation of data pairs from one batch of training data, whereas the RL will require a time series of data streams for model learning. Therefore it is hard to compare the learning process of the two algorithms side-by-side directly. In this study, we are using a state-of-art Reinforcement Learning algorithm called Deep Deterministic Policy Gradient (DDPG) [27] to compare with results obtained using DKRC. The detailed comparison of the DKRC with controller designed using pre-determined choice of basis function such as using Extended Dynamic Mode Decomposition (EDMD) or DMD will be a subject of our future investigation.

C. Inverted Pendulum

The inverted pendulum environment is a classic 2-dimensional state space environment with one control input, as shown in Equation 12. A visualization of the simulated environment is also shown in Figure 3. The game starts with the pendulum pointing down with a randomized initial disturbance. The goal is to invert the pendulum to its upright position. In order to finish the game, the model needs to apply control input to drive the pendulum from start position ($\theta \in (-\pi, 0) \cup (0, \pi), \dot{\theta} = 0$) to goal position ($\theta = 0, \dot{\theta} = 0$).

$$\boldsymbol{\chi} = [\cos\theta, \sin\theta, \dot{\theta}], \quad where\ \theta \in [-\pi, \pi], \dot{\theta} \in [-8, 8]$$
$$\boldsymbol{U} = [u], \quad where\ u \in [-2, 2]$$

(12)

In this case, the recorded θ orientation angle is in the form of a pair of cosine and sine values to ensure a smooth transition at $-\pi$ and π. The dynamical governing equation can be shown as follows:

$$ml^2\ddot{\theta} = -\gamma\dot{\theta} + mgl\sin(\theta) + u \qquad (13)$$

This governing equation is unknown to our model, and the model needs to recover the physics-based on limited time-series data only. Moreover, the angular velocity ($\dot{\theta}$) and the torque input (u) at the shaft are also limited to a small range to add difficulties to finish the game. In most of the testing cases, we find that the pendulum needs to learn a swing back-and-forth motion before being able to collect enough momentum to swing up to the final vertical upright position. Although optimal control is possible, the trained model needs to adapt to the limitation imposed by the testing environment and make decisions solely based on state observations.
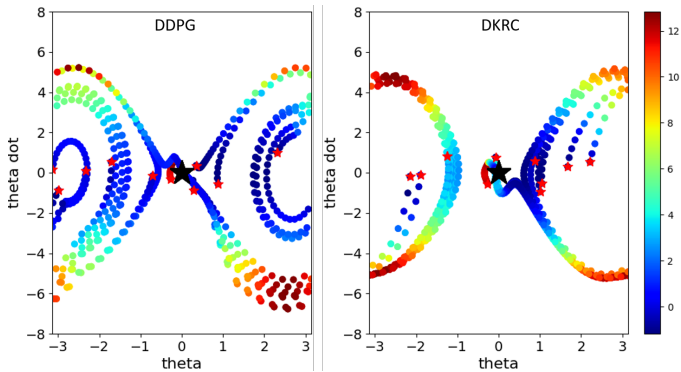

Fig. 5. Trajectories of swinging pendulum under control using DDPG reinforcement learning model (left) and the proposed DKRC algorithm (right). The figure exhibits the trajectories of the tip of the pendulum breaking through Hamiltonian energy level lines to arrive at the goal position ($\theta = 0, \dot{\theta} = 0$)

We train the DDPG reinforcement learning model first for benchmark purposes. Figure 5 shows the trajectory of the pendulum in 2D (θ, $\dot{\theta}$) space based on trained DDPG algorithm. The black star in the center of Figure 5 is the goal position, and the concentric lines on both sides correspond to Hamiltonian total energy levels. It is clear that due to the implicit model learned during the reinforcement learning

process, the RL method cannot finish the game within a short time and therefore left many failed trails.

Besides the 2D comparison shown in Figure 5, we are also presenting the state vs. time plots for the games played by DKRC model in 3D plot, as shown in Figure 6. In each deployment, the game starts with different initial conditions. The 3D plot is color-mapped by Hamiltonian energy levels. As indicated by the figure, the control generated by DKRC effectively minimized the system energy level within a short amount of deployment time.
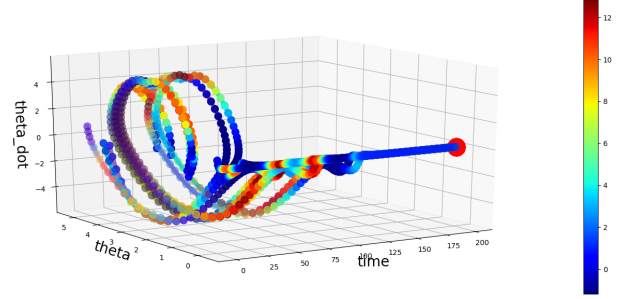

Fig. 6. Example 3D trajectories from 10 games played by DKRC, each game is reset by random initialization, 3D plot color-mapped by Hamiltonian energy level. Red Point: Goal states

D. Lunar Lander

As depicted in Figure 7, the lunar lander environment is a 6-dimensional state space with 2 control inputs. The state observables involve more terms and more closely related to a real-world game environment than previous classic dynamics examples.

$$\boldsymbol{\chi} = [x, y, \theta, \dot{x}, \dot{y}, \dot{\theta}]$$
$$\boldsymbol{U} = [u_1, u_2], \quad where,\ u_1 \in [0, 1], u_2 \in [-1, 1]$$

(14)

where x, y are the Cartesian coordinates of the lunar lander, and θ is the orientation angle measured clockwise from the upright position. u_1 represents the main engine on the lunar lander, which only takes positive input. In contrast, the u_2 is the side engine that can take both negative (left engine firing) and positive (right engine firing) values. The goal of this game is to move lunar lander from initial position ($x = 10$, $y = 13$) to the landing zone ($x = 10$, $y = 4$), subject to randomized initial gust disturbance and dynamics specified in the Box2D environment. The data collection can be obtained by running a un-trained reinforcement learning algorithm with random policy together with a random noise generated by the Ornstein–Uhlenbeck process during the data collection procedure. These randomization treatments are implemented to ensure enough exploration vs. exploitation ratio from the Reinforcement Learning model. In this study, we use 1876 data pairs obtained from playing five games to train our DKRC model.

The exact model for this system cannot be extracted directly from the OpenAI Gym environment but has to be identified either through a data-driven model-based approach (this paper) or model-free approach (e.g., reinforcement learning). A game score is calculated by OpenAI Gym where the system will reward smooth landing within the landing
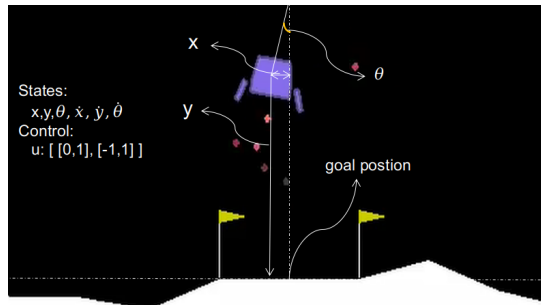
1894

Fig. 7. Lunar Lander

zone (area in between double flags) while penalizing the fuel consumption by engines (u_1, u_2). The proposed method is first to generate the model of system identification. Assuming after the Koopman operator transform, the dynamical system can be considered to be linear. Therefore, model predictive control can be applied. The trajectories of the lunar lander successfully finishing the game are shown in Figure 8. Note that the trajectories for the ten games are spread out in the simulation environment and returning back to the same goal position. The reason for the spread-out behavior is that the initialization of each game will randommly assign an initial velocity and the control algorithm need to apply control to offset the drift while keeping balance in the air. In this simulation environment, the DKRC model was demonstrated to be able to learn the dynamics and cope with the unforeseen situation.
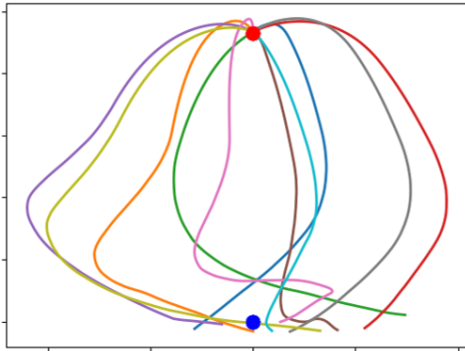

Fig. 8. Example 2D trajectories from 10 games played by DKRC. Red Point: Initial states, Blue Point: Goal states

V. CONCLUSIONS

The proposed MPC controller designed by Deep Koopman Representation for Control (DKRC) has the benefit of being completely data-driven in learning, whereas it remains to be model-based in control design. The DKRC model is efficient in training and sufficient in the deployment in two OpenAI Gym environments compared to a state-of-the-art Reinforcement Learning algorithm (DDPG).

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT press, 2018. 1

[2] I. Mezić, "Spectral properties of dynamical systems, model reductions and decompositions," *Nonlinear Dynamics*, 2005. 1

[3] U. Vaidya and P. G. Mehta, "Lyapunov measure for almost everywhere stability," *IEEE Transactions on Automatic Control*, vol. 53, pp. 307–323, 2008. 1

[4] U. Vaidya, P. Mehta, and U. Shanbhag, "Nonlinear stabilization via control Lyapunov measure," *IEEE Transactions on Automatic Control*, vol. 55, pp. 1314–1328, 2010. 1

[5] A. Raghunathan and U. Vaidya, "Optimal stabilization using lyapunov measures," *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1316–1321, 2014. 1

[6] M. Budisic, R. Mohr, and I. Mezic, "Applied koopmanism," *Chaos*, vol. 22, pp. 047 510–32, 2012. 1

[7] M. Korda and I. Mezić, "Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control," *Automatica*, vol. 93, pp. 149–160, 2018. 1, 2

[8] Y. Susuki and I. Mezic, "Nonlinear koopman modes and coherency identification of coupled swing dynamics," *IEEE Transactions on Power Systems*, vol. 26, no. 4, pp. 1894–1904, 2011. 1

[9] E. Kaiser, J. N. Kutz, and S. L. Brunton, "Data-driven discovery of koopman eigenfunctions for control," *arXiv preprint arXiv:1707.01146*, 2017. 1, 2

[10] A. Surana and A. Banaszuk, "Linear observer synthesis for nonlinear systems using koopman operator framework," in *Proceedings of IFAC Symposium on Nonlinear Control Systems*, Monterey, California, 2016. 1

[11] S. Peitz and S. Klus, "Koopman operator-based model reduction for switched-system control of pdes," *arXiv preprint arXiv:1710.06759*, 2017. 1

[12] A. Mauroy and I. Mezić, "Global stability analysis using the eigenfunctions of the koopman operator," *IEEE Transactions on Automatic Control*, vol. 61, no. 11, pp. 3356–3369, 2016. 1

[13] B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," *Nature communications*, vol. 9, no. 1, pp. 1–10, 2018. 1

[14] E. Yeung, S. Kundu, and N. Hodas, "Learning deep neural network representations for koopman operators of nonlinear dynamical systems," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 4832–4839. 1

[15] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016. 1, 4

[16] B. O. Koopman, "Hamiltonian systems and transformation in hilbert space," *Proceedings of the national academy of sciences of the united states of america*, vol. 17, no. 5, p. 315, 1931. 1

[17] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, "A data–driven approximation of the koopman operator: Extending dynamic mode decomposition," *Journal of Nonlinear Science*, vol. 25, no. 6, pp. 1307–1346, 2015. 1

[18] M. O. Williams, M. S. Hemati, S. T. Dawson, I. G. Kevrekidis, and C. W. Rowley, "Extending data-driven koopman analysis to actuated systems," *IFAC-PapersOnLine*, vol. 49, no. 18, pp. 704–709, 2016. 1

[19] M. O. Williams, C. W. Rowley, and I. G. Kevrekidis, "A kernel-based approach to data-driven koopman spectral analysis," *arXiv preprint arXiv:1411.2260*, 2014. 1

[20] J. L. Proctor, S. L. Brunton, and J. N. Kutz, "Dynamic mode decomposition with control," *SIAM Journal on Applied Dynamical Systems*, vol. 15, no. 1, pp. 142–161, 2016. 2

[21] E. Kaiser, J. N. Kutz, and S. L. Brunton, "Data-driven approximations of dynamical systems operators for control," in *The Koopman Operator in Systems and Control*. Springer, 2020, pp. 197–234. 2

[22] A. Broad, T. Murphey, and B. Argall, "Learning models for shared control of human-machine systems with unknown dynamics," *arXiv preprint arXiv:1808.08268*, 2018. 2

[23] P. You, J. Pang, and E. Yeung, "Deep koopman controller synthesis for cyber-resilient market-based frequency regulation," *IFAC-PapersOnLine*, vol. 51, no. 28, pp. 720–725, 2018. 2

[24] X. Ma, B. Huang, and U. Vaidya, "Optimal quadratic regulation of nonlinear system using koopman operator," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 4911–4916. 2

[25] B. Huang, X. Ma, and U. Vaidya, "Feedback stabilization using koopman operator," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 6434–6439. 2

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014. 2

[27] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015. 4