Supplementary Information for: Distilling Governing Laws and Source Input for Dynamical Systems from Videos

Lele Luan, Yang Liu, Hao Sun

Contents

This supplementary document provides a detailed description of the proposed deep learning scheme and its testing examples in the manuscript submitted to IJCAI-2022: Distilling Governing Laws and Source Input for Dynamical Systems from Videos.

1 Video Dataset

The videos studied here are synthesized by MATLAB to imitate the real dynamical systems being recorded by video cameras. At first, the dynamical system is pre-defined and its trajectory is simulated with function ode113, a embedded ordinary differential equation(ODE) solver in MATLAB. The moving object is simulated by a constant circle marker in 2D coordinate system with the physical trajectory coming from the ODE solver. The original colorful image with moving object only has a size of 256×256. The object image is downsampled into 64×64 later. Then an image with size of 64×64 is randomly selected to simulate the background in real videos. In the background image, the pixels at the location of moving object are removed and replaced by the downsampled object. The generated videos are colorful videos with RGB channels for each frame.

2 Governing Equation Discovery from Videos

2.1 Network architecture

In this work, we present a novel end-to-end unsupervised deep learning paradigm to uncover the governing equations of dynamical systems recorded by the video camera. In the studied dynamical systems, the physical laws are presented by the moving object(s) in the coordinate system which is parallel to the image space. Our goal is to simultaneously extract the physical states of moving object(s), and identify their governed parsimonious closed-form equation and the unknown input applied to the dynamical system. The high-level view of the designed network is shown in Figure. 2(A) which involves a complete autoencoder to condense the video frame into physical states and the forward propagation of physics between two consecutive frames. It should be noted that, the two consecutive video frames enables the unknown external input to be considered in the designed neural network. The high-level view network consists of four basic components which are Encoder φ, Coordinate-Consistent Decoder ψ, Spatial-Physical Transformation (spatial to physical transformation $\mathcal{T}$ and physical to spatial transformation $\tilde{\mathcal{T}}$), and Sparse Regression on physical states. All components are discussed as follows.

Because the input data is high-dimensional video images, a complete autoencoder is employed to condense the images into low-dimensional latent variables. The complete autoencoder here is built by the Encoder φ and Coordinate-Consistent Decoder ψ. In the Encoder, each video frame is fed into a U-Net which constitutes several convolution layers and max pooling layers at first. Because U-Net has been proved efficient on extracting the multi-scale features from high-dimensional image data. The output of Encoder is the unnormalized masks which have the same size as the input frame and contain the location information of the moving object. The unnormalized object masks are then stacked with the constant masks of background and then passed through a Softmax to generate the masks each of which is assigned to a pixel in the video frame. Those masks are multiplied with the input video frame pixel-wisely and the multiplication is flatten and fed into a localization network which is made of 3 fully-connected layers (200 units in hidden layers and 2 units in the output layer). Note that the number of nodes in the output layer depends on the number of elements needed to determine the locations of moving object(s) in the scene. The output elements from the dense network are then passed through a saturating non-linear activation function $H/2 \cdot \tanh(\cdot) + H/2$ to produce the pixel coordinate of the moving object at two dimensions in the video frame with size of $H \times H$. Here, the volume of U-Net can be fine-tuned by changing the number of layers and number of channels in each layer.

The decoder takes the spatial coordinate of moving object $\mathbf{x}_s$ as input given by the Physical-Spatial Transformation of physical state and outputs a reconstructed image. Because in the studied dynamical systems, the physical law is presented by the moving object(s). The designed encoder-decoder should have the capacity to determine the locations (relative locations) of moving object(s) at different time steps. Instead of using conventional decoders (like convolutional decoder, the failure of conventional autoencoder will be discussed in the baseline), the Coordinate-Consistent Decoder is employed here to reconstruct the video fame from the spatial coordinate. In the Coordinate-Consistent Decoder, a fixed relationship between the latent spatial coordiante and pixel-coordiante correspondence is built based on the spatial transformer (ST) network. In the conventional decoders, the input is reconstructed by several fully connected or convolutional layers. While the condensed latent variable by these conventional encoders may not be the physical states representing the location of the moving object(s) in image space. Instead, in the Coordinate-Consistent Decoder, the contents and masks of moving object(s) and the background are learned by independent neural networks at first and the video images are reconstructed by moving the object(s) according to the

input spatial coordinates. The ST based decoder not only guarantees that the extracted physical states represent the locations of moving object(s) in image space, but also enables the fixed relationship between the latent spatial coordiante and pixel-coordiante correspondence. If the input spatial coordinate to decoder is $\mathbf{x}_s = (x_s, y_s)$, the parameters ω of ST is to place the center of the writing attention window for moving object at this point. The Coordinate-Consistent Decoder proposed in [1] uses ST with inverse transformation parameters to represent the translation, scale or angle of the writing attention window. For a general affine transformation with translation (t_x, t_y), angle ϑ and scale s, the source image coordinates can be modified according to:

$$\begin{pmatrix} x_s^o \\ y_s^o \\ 1 \end{pmatrix} = \begin{pmatrix} s \cdot \cos\vartheta & s \cdot \sin\vartheta & t_x \\ -s \cdot \sin\vartheta & s \cdot \cos\vartheta & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_s^s \\ y_s^s \\ 1 \end{pmatrix} \tag{1}$$

where (x_s^o, y_s^o) and (x_s^s, y_s^s) are the output and source image coordinates. This transformation can be obtained with a ST by inverting Equation 1:

$$\begin{pmatrix} x_s^s \\ y_s^s \\ 1 \end{pmatrix} = \frac{1}{s} \begin{pmatrix} \cos\vartheta & -\sin\vartheta & -t_x\cos\theta + t_y\sin\vartheta \\ \sin\vartheta & \cos\vartheta & -t_x\sin\vartheta - t_y\cos\vartheta \\ 0 & 0 & s \end{pmatrix} \begin{pmatrix} x_s^o \\ y_s^o \\ 1 \end{pmatrix} \tag{2}$$

In the Coordinate-Consistent Decoder, the image can be reconstructed by using the ST with parameters ω given in Equation 2. Each video frame can be reconstructed by the combination of moving object(s) and background which are learned by the independent neural networks from constant inputs. The ith moving object is represented by a learnable content $\mathbf{c}^i \in [0,1]^{H\times H\times C}$ and mask tensor $\mathbf{m}^i \in \mathbb{R}^{H\times H\times 1}$. The background is also represented by a content $\mathbf{c}^b \in [0,1]^{H\times H\times C}$, but a mask $\mathbf{m}^b$ which is a tensor with ones. The content(s) and masks of the moving object(s) are transformed with ST as $[\hat{\mathbf{c}}^i, \hat{\mathbf{m}}^i] = \mathrm{ST}([\mathbf{c}^i, \mathbf{m}^i], \omega^i)$. The resulting logit masks for moving object(s) are stacked with the constant background mask $\mathbf{m}^b$ and passed into a Softmax, $[\tilde{\mathbf{m}}^i, \tilde{\mathbf{m}}^{i+1}, ..., \tilde{\mathbf{m}}^b] = \mathrm{softmax}(\mathbf{m}^i, \mathbf{m}^{i+1}, ..., \mathbf{m}^b)$. Then the output image reconstruction can be represented as the summation of the multiplications of contents and masks for moving object(s) and background:

$$\tilde{\mathbf{I}} = \tilde{\mathbf{m}}^b \odot \mathbf{c}^b + \sum_{i=1}^{J} \tilde{\mathbf{m}}^i \odot \hat{\mathbf{c}}^i \tag{3}$$

where J denotes the number of moving object(s) in the scene. In the Coordinate-Consistent Decoder, the fully-connected networks which learn the the contents and masks of moving object(s) and the background have a constant array of 1s as input. The number of layers and number of nodes for each layer of these independent neural networks can be tuned based on the training. The outputs of these networks are activated with tanh and reshaped into 2D content and mask.

The learned spatial coordinate of moving object is then passed through a spatial to physical coordinate transformer to produce the expected physical states which may be responsible to the parsimonious closed-form governing equations. If we assume the dynamic system is built in the 2D plane image space, the transformation between spatial and physical coordinate can be implemented by standard 2D Cartesian coordinate transformation. In coordinate transformation, the position of moving object is kept fixed while the coordinate system is transformed relative to the object. The coordinate transformation here involves translation and scaling. If the transformation has translation vector $\tilde{\mathbf{t}} = (\tilde{t}_x, \tilde{t}_y)^T$ and scaling factor $\tilde{s}$, the physical coordinate can be expressed from spatial coordinate as

$$\mathbf{x}_p^T = \mathcal{T}(\mathbf{x}_s^T) = \tilde{s}(\mathbf{x}_s^T - \tilde{\mathbf{t}}) \tag{4}$$

Then the transformation can be expanded as

$$\begin{pmatrix} x_p \\ y_p \end{pmatrix} = \mathcal{T}(\mathbf{x}_s^T) = \tilde{s} \begin{pmatrix} x_s - \tilde{t}_x \\ y_s - \tilde{t}_y \end{pmatrix} \tag{5}$$

Likewise, the physical to spatial transformation can be expressed as

$$\mathbf{x}_s^T = \tilde{\mathcal{T}}(\mathbf{x}_p^T) = (1/\tilde{s}\mathbf{x}_p^T) + \tilde{\mathbf{t}} \tag{6}$$

Here $\mathcal{T}$ and $\tilde{\mathcal{T}}$ share the same transformation factors $\tilde{s}$ and $\tilde{\mathbf{t}}$. Those factors are trainable variables and will be optimized in the whole network training.

2.2 Loss Functions

The total loss function used in training is a weighted sum of four loss terms: autoencoder reconstruction loss $\mathcal{L}_{recon}$, physical state derivative loss $\mathcal{L}_{d\mathbf{x}/dt}$, forward video frame reconstruction loss from integral physical state $\mathcal{L}_{int}$, and candidate function coefficients regularization $\mathcal{L}_{reg}$. For one single input video I with N frames, each loss is explicitly defined as follows:

$$\mathcal{L}_{recon} = \frac{1}{N} \sum_{k=1}^{N} \left\| I_k - \psi(\tilde{\mathcal{T}}(\mathcal{T}(\varphi(I_k)))) \right\|_2^2 \tag{7}$$

$$\mathcal{L}_{\dot{\mathbf{x}}_p} = \frac{1}{N} \sum_{k=1}^{N} \left\| \nabla_t(\mathcal{T}(\varphi(I_k)) - \boldsymbol{\Theta}((\mathcal{T}(\varphi(I_k)))^T)\boldsymbol{\Xi} + g_k \right\|_2^2 \tag{8}$$

$$\mathcal{L}_{int} = \frac{1}{N} \sum_{k=1}^{N} \left\| I_{k+1} - \psi(\tilde{\mathcal{T}}(\mathbf{RK4}(\mathcal{T}(\varphi(I_k)), g_k, g_{k+1})) \right\|_2^2 \tag{9}$$

$$\mathcal{L}_{reg} = \frac{1}{2n} \|\boldsymbol{\Xi}\|_{0.5} \tag{10}$$

The total loss function is

$$\mathcal{L}_{total} = \mathcal{L}_{recon} + \mathcal{L}_{\dot{\mathbf{x}}_p} + \lambda_2 \mathcal{L}_{int} + \lambda_3 \mathcal{L}_{reg} \tag{11}$$

$\mathcal{L}_{recon}$ ensures that the autoencoder can accurately reconstruct the video frame from the physical states of the moving object. $\mathcal{L}_{\dot{\mathbf{x}}_p}$ and $\mathcal{L}_{int}$ ensure that the discovered physical model captures the dynamics of the physical states extracted from autoencoder incorporated with spatial to physical and its inverse transformation. Here, **RK4** denotes the 4th-order Runge-Kutta (RK4) numerical method to calculate the physical state one-step ahead of current video frame. g_k represents the unknown input at time step k. $\mathcal{L}_{reg}$ promotes the sparsity of dynamic model. The choice of loss weights λ_1, λ_2 and λ_3 are discussed in Section 2.4.

2.3 Training Procedure

There are four steps in the network training for uncovering the closed-form governing equations of dynamical systems with unknown inputs: pre-training, total loss optimization, sequential thresholding and refinement. In the whole network, the trainable variables include the weights in Encoder W_φ, weights in Coordinate-Consistent Decoder W_ψ, transformation factors in Spatial-Physical Transformation $\mathcal{T}$ (same as in Physical-Spatial Transformation $\tilde{\mathcal{T}}$), physical library function coefficient $\boldsymbol{\Xi}$ and system unknown input array $\mathbf{g}$. In the pre-training, the network is trained by only

optimizing the autoencoder reconstruction loss $\mathcal{L}_{recon}$, and only W_φ and W_ψ are updated. The pre-training ends up with a small value for autoencoder reconstruction loss. The pre-trained model is then loaded for the following total loss training in which all trainable variables are optimized. After total loss training, both $\mathcal{L}_{\dot{\mathbf{x}}_p}$ and $\mathcal{L}_{int}$ are reduced to small values and the candidate function coefficient matrix shows high sparsity. Later, the total trained model is loaded for the following sequential thresholding. In sequential thresholding, the network is trained along thresholding the candidate function terms whose coefficients are smaller than the given threshold. Finally, the sequential thresholding model is loaded for the final refinement. In the refinement, the network is trained by only updating the coefficients of the candidate function terms left from sequential thresholding. It should be noted that, in the sequential thresholding, once the function terms are filtered out for their smaller coefficient than threshold, the coefficients for those terms are set as 0 and will not be trained in the following epochs. After refinement training, the dynamic model is discovered from the candidate function coefficients in sparse regression.

2.4 Choice of Hyperparameters

The training procedure described above requires the choice of several hyperparameters: chiefly the order of discovered ODEs, candidate function polynomial order for sparse regression, initial values for coordinate transformation factors, and the loss weight penalties λ_1, λ_2, λ_3. The choice of these parameters greatly impacts the success of governing equation discovery. Here we outline some guidelines for choosing these parameters.

The most important hyperparameter choice is the order of discovered ODEs d, as this impacts the interpretation of the extracted physical states and the associated dynamical model. In this paper we built a general method to discover the governing equations for both first and second-order dynamical systems while only the second-order dynamical systems are discussed as examples. Nevertheless, the order may not be obvious in real system. Here we suggest using the sparsity of discovered candidate function coefficients to determine the order d. Because choosing d that is not correct to the real system may still result in a valid model for the dynamics; however, obtaining a sparse model may be more difficult. One may want to train the models with order $d = 1$ and $d = 2$. The order leading to much more obvious sparsity in the coefficient matrix will be the determined order d.

Since the equations discovered here are identified by sparse regression, this method requires a choice of library functions to interpret the extracted physical states. All the examples shown here use polynomial library terms. But for general systems, the best library functions to use may be unknown, and choosing the wrong library functions can leads to the failure discovery. A recommended practice is to start with polynomial models, as many common physical models are represented by polynomials. In addition, the order of polynomial library is still a hyperparameter to tune. On one hand, we prefer to make the discovery more general by using more diverse polynomial candidate terms. On the other hand, balancing the increasing theoretical and computational complexity is crucial for applications. Although the higher order of the polynomial is, the more likely the exact terms will be uncovered from the video. Nevertheless, too large scale candidate library may obscure the sparsity of physical model and leads to increasing difficulty on simplest model discovery. As simple library function is typically easy to train, one may want to start by using lower polynomial order like 2. Once the sparsity has been found, the model can be trained by fine-tuning other hyperparameters while keeping the library functions.

The choice of initial values of spatial physical transformation also affects the discovery. After pre-training, Encoder captures the spatial coordinate of moving object which is then converted into physical states. With other loss terms being added, the Encoder and Spatial-Physical Transformer

are constrained by the physical law for physical states extraction. If the initial transformation factors for spatial physical transformation have huge discrepancy to the ground truth, physical law will not provide proper constraint on the physical states. A recommended way to initialize the Spatial-Physical Transformation is to determine the transformation factors based on the spatial coordiante of moving object after pre-training. Take the studied videos in this paper for example, the initialized translation factor enables the origin of physical coordinate system at the centre of spatial coordiante represented trajectories. For the scaling factor, since the discovered physical trajectories and governing equations have a scaled relationship with the ground truth, the choice of scaling factor may not be a big issue. Because the physical coordinate is typically smaller than pixel coordinate, one may want to try the scaling factor with multiplier of 10.

The choice of loss weight penalties λ_1, λ_2, λ_3 also has a significant impact on the success of the governing equation discovery. The loss weight λ_2 can be determined at first. If the governing equation is discovered, the reconstructed video frames from the forward and backward integral physical states should have the same error as the autoencoder reconstruction loss, the first term in total loss function, so the loss weight λ_2 is chosen as 1.0. Then loss weight λ_1 is chosen to be three orders of magnitude less than 1.0, around 10^{-3}. Although the derivative of physical states loss promotes the physical constraint to the physical states extraction and the governing equation discovery, λ_1 cannot be too large. Because too large λ_1 will destroy the autoencoder pre-trained for spatial coordiante of moving object. Furthermore, too large λ_1 encourages shrinking of the magnitude of extracted physical states. The third parameters λ_3 determines the strength of the regularization of the sparse regression model coefficients $\mathbf{\Xi}$ and thus affects the sparsity of the resulting models. If this value is too large, the model will be too simple and achieve poor prediction; if it is too small, the modes will be nonsparse and prone to overfitting. The coefficients regularization loss weight requires the most tuning and should typically be chosen by trying a range of values and assessing the level of sparsity in the discovered equation.

2.5 Discovery on State-space Modeling

The proposed method is applicable to the discovery for second-order dynamical systems by transferring the systems into state-space modeling. Discovery on the second order systems shares the same network with the first order but slight modification in the physical constraint and equation identification. In the network, Encoder still condenses the video frame into spatial coordinate of moving object (x_s, y_s), which is followed by Spatial-Physical Transformation to get physical coordinate (x_p, y_p). If we define the variables $x_1 = x_p$ and $y_1 = y_p$ and calculate the first order derivatives of x_1 and y_1 as $\dot{x}_1 = x_2$ and $\dot{y}_1 = y_2$. Then the second order equation group is transferred into 4 first order equations as

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{y}_1 \\ \dot{y}_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ f(x_1, x_2, y_1, y_2) \\ y_2 \\ g(x_1, x_2, y_1, y_2) \end{pmatrix} \tag{12}$$

where $f(x_1, x_2, y_1, y_2)$ and $g(x_1, x_2, y_1, y_2)$ are the equations needed to be discovered. Noteworthy, the state-space model transfer also make the Runge-Kutta method applicable to the integral of physical states. Although both physical coordinate and its derivative are obtained in latent space, only physical coordinate is feed into decoder for video reconstruction. In addition, in order to simplify the discovery, the equation $\dot{x}_1 = x_2$ and $\dot{y}_1 = y_2$ are given in the discovery.

3 Source Code

We use the Python API for TensorFlow to implement and train our network. Our code will be publicly available at github.com after the manuscript is published.

4 Example Systems

This section provides the detailed description of the studied dynamical systems, especially the parameters in neural network training and the discovered result for each system.

4.1 Single mass single degree of freedom (SMSD) system

The ground truth governing equation of the studied SMSD system is

$$\ddot{x} = -0.1\dot{x} - 1.0x + g \tag{13}$$

where g is the unknown input. To generate the video set, the ground truth physical trajectories for this dynamical system are simulated from random initial conditions with $x(0), \dot{x}(0) \in [-1.0, 1.0]$. In order to increase the diversity of the training dataset, the trajectories with $max|x, \dot{x}| < 0.70$ are removed with 256 trajectories left in total. Hyperparameters used for training this dynamical system are shown in Table 1. First of all, the network is trained with 200 epochs for the pre-training with only parameters of encoder and decoder being updated. After the pre-training, the autoencoder reconstruction loss is very small, around 9.0×10^{-6}, which means the designed network has the capacity to extract the latent space and reconstruct the video frames from latent space. The forward and backward frame reconstruction loss is 9.5×10^{-6} and physical derivative loss is 0.037. Then the network was kept training with the total loss for 300 epochs. In the total loss training, the network decreases the physical loss and reduces the error of forward and backward reconstructed frames at the same time by updating sparse regression coefficients. During the total loss training, the autoencoder reconstruction loss is very small, less than 1.0×10^{-5} along with the whole training. After total loss training, the autoencoder reconstruction loss is 9.0×10^{-6}, forward and backward frame reconstruction loss 9.0×10^{-6}, and physical states derivative loss 8.1×10^{-5}. For sparse regression coefficients, only 2 expected terms have a coefficient larger than 0.02. The trained total loss model is then loaded for the following sequential thresholding training. In sequential thresholding, the threshold value is 0.02 and sparse regression coefficients are thresholded every 100 epochs. After 200 epochs, the redundant terms are filtered out and only 2 terms are kept. Finally, the network is kept training for 200 epochs with the left 2 candidate terms. The discovered closed-form equation for this dynamical system is

$$\ddot{x} = -0.1\dot{x} - 1.0x + g' \tag{14}$$

where g' is the identified external input.

4.2 Single mass two degrees of freedom (SMTD) system

The ground truth governing equation of the studied SMTD system is

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} = \begin{pmatrix} -0.1\dot{x} - 1.0x + g_1 \\ -0.2\dot{x} - 2.0x + g_2 \end{pmatrix} \tag{15}$$

where g_1 and g_2 are the unknown inputs applied to the moving mass in two directions. To generate the video set, the ground truth physical trajectories are simulated from random initial conditions

Table 1: Hyperparameter values for the SMSD system

Parameter	Value
Video number	256
Video length (unknown input length)	300
Time step	1/10.0
Batch size	4
Learning rate	5×10^{-4}
Dynamic model order	2
Candidate function polynomial order	2
Physical states derivative loss $\mathcal{L}_{\dot{\mathbf{x}}_p}$ weight λ_1	1×10^{-3}
Forward/backward frame reconstruction loss $\mathcal{L}_{int}$ weight λ_2	1.0
Candidate function coefficient regularization loss $\mathcal{L}_{reg}$ weight λ_3	1×10^{-5}

with $x(0), \dot{x}(0), y(0), \dot{y}(0) \in [-1.0, 1.0]$. The trajectories with $max|x| < 0.80$ and $max|y| < 0.80$ are removed with 256 trajectories left in total. Hyperparameters used for discovering this dynamical system are shown in Table 2. First of all, the network was trained with 300 epochs for the pre-training with only parameters of encoder and decoder being updated. After the pre-training, the autoencoder reconstruction loss is around 7.7×10^{-6}, the forward and backward frame reconstruction loss is 8.0×10^{-6} and physical derivative loss is 0.178. Then the network was kept training with the total loss for 400 epochs. During the total loss training, the autoencoder reconstruction loss is very small, less than 1.0×10^{-5} along with the whole training. After total loss training, the autoencoder reconstruction loss is 7.7×10^{-6}, the forward and backward frame reconstruction loss 7.7×10^{-6}, and the physical states derivative loss 0.0007. For sparse regression coefficients, only 4 expected terms have a coefficient larger than 0.02. The trained total loss model is then loaded for the following sequential thresholding training. In sequential thresholding, the threshold value is 0.02 and sparse regression coefficients are thresholded every 100 epochs. After 200 epochs, the redundant terms are filtered out and only 4 terms are kept. Finally, the network is kept training for 200 epochs with the left 4 candidate terms. The discovered closed-form equation for this SMTD system is

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} = \begin{pmatrix} -0.1\dot{x} - 1.0x + g_1' \\ -0.2\dot{x} - 2.0x + g_2' \end{pmatrix} \tag{16}$$

where g_1' and g_2' are the identified external inputs.

4.3 Two masses two degrees of freedom (TMTD) system

The ground truth governing equation of the studied TMTD system is

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} = \begin{pmatrix} -0.15\dot{x}_1 + 0.1\dot{x}_2 - 1.5x_1 + 1.0x_2 \\ 0.10\dot{x}_1 - 0.1\dot{x}_2 + 1.0x_1 - 1.0x_2 + g \end{pmatrix} \tag{17}$$

where g is the unknown input applied to the right moving mass. To generate the video set, the physical trajectories are simulated from random initial conditions with $x(0), \dot{x}(0), y(0), \dot{y}(0) \in [-1.0, 1.0]$. The trajectories with $max|x| < 0.80$ and $max|y| < 0.80$ are removed with 256 trajectories left in total. Hyperparameters used for discovering this dynamical system are shown in Table 3. First of all, the network was trained with 300 epochs for the pre-training with only parameters of encoder and decoder being updated. After the pre-training, the autoencoder reconstruction loss is around

Table 2: Hyperparameter values for the SMTD system

Parameter	Value
Video number	256
Video length (unknown input length)	320
Time step	1/16.0
Batch size	4
Learning rate	5×10^{-4}
Dynamic model order	2
Candidate function polynomial order	2
Physical states derivative loss $\mathcal{L}_{\dot{\mathbf{x}}_p}$ weight λ_1	1×10^{-3}
Forward/backward frame reconstruction loss $\mathcal{L}_{int}$ weight λ_2	1.0
Candidate function coefficient regularization loss $\mathcal{L}_{reg}$ weight λ_3	1×10^{-5}

1.1×10^{-5}, the forward and backward frame reconstruction loss is 1.2×10^{-5} and physical derivative loss is 0.027. Then the network was kept training with the total loss for 500 epochs. During the total loss training, the autoencoder reconstruction loss is very small, less than 1.2×10^{-5} along with the whole training. After total loss training, the autoencoder reconstruction loss is 1.1×10^{-5}, the forward and backward frame reconstruction loss 1.1×10^{-5}, and the physical states derivative loss 0.0001.

Because, in this dynamical system, the physical states for two moving masses are not independent (physical states for one moving object should be considered in building the sparse regression for the other moving object), the latent dimension in sparse regression for governing equation discovery is 8 (with state-space modeling). After total loss training, there are 12 terms having a coefficient larger than 0.05 in sparse regression coefficients. In the following sequential thresholding and refinement training, all these 12 terms are kept. Fig.1 shows the learned candidate function coefficients for $\ddot{x}_1$, $\ddot{y}_1$, $\ddot{x}_2$ and $\ddot{y}_2$ after refinement training. Here post-processing is needed to achieve the correct discovery. In this dynamical system, the objects are moving along the horizontal direction and physical states at vertical direction is a constant value. Furthermore, the learned equations for both $\ddot{y}_1$ and $\ddot{y}_2$ are 0. Here, if assume $y_1 = -0.1422$ and $y_2 = -0.1576$, The discovered equation for this TMTD can be re-written as

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} = \begin{pmatrix} -0.15\dot{x}_1 + 0.1\dot{x}_2 - 1.5x_1 + 1.0x_2 \\ 0.10\dot{x}_1 - 0.1\dot{x}_2 + 1.0x_1 - 1.0x_2 + g' \end{pmatrix} \tag{18}$$

where g' is the identified external input. The discovered equation after post-processing is exactly the same as the ground-truth.

4.4 Nonlinear dynamical system

In addition to the linear dynamical systems discussed above, our method was also tested on discovering the governing equation of nonlinear dynamics. Here, we did not consider source input in this case because the solution to the problem easily become non-unique when a small number of video measurements are used. Our object here is to simply test whether our proposed method is applicable to discovery of nonlinear governing equations. The tested nonlinear dynamics is Cubic Oscillator whose ground truth governing equation is

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} p_1 x^3 + p_2 y^3 \\ p_3 x^3 + p_4 y^3 \end{pmatrix} \tag{19}$$

Table 3: Hyperparameter values for the TMTD system

Parameter	Value
Video number	256
Video length (unknown input length)	320
Time step	1/16.0
Batch size	4
Learning rate	5×10^{-4}
Dynamic model order	2
Candidate function polynomial order	2
Physical states derivative loss $\mathcal{L}_{\dot{\mathbf{x}}_p}$ weight λ_1	5×10^{-3}
Forward/backward frame reconstruction loss $\mathcal{L}_{int}$ weight λ_2	1.0
Candidate function coefficient regularization loss $\mathcal{L}_{reg}$ weight λ_3	1×10^{-5}

	$\ddot{x}_1$	$\ddot{y}_1$	$\ddot{x}_2$	$\ddot{y}_2$
x_1	−1.096	0	1.003	0
y_1	0	0	0	0
x_2	0.999	0	−0.776	0
y_2	0	0	0	0
$\dot{x}_1$	−0.151	0	0.101	0
$\dot{y}_1$	0	0	0	0
$\dot{x}_2$	0.101	0	−0.101	0
$\dot{y}_2$	0	0	0	0
x_1^2	0	0	0	0
x_1y_1	1.373	0	0	0
x_1x_2	0	0	0	0
x_1y_2	1.322	0	0	0
$x_1\dot{x}_1$	0	0	0	0
$x_1\dot{y}_1$	0	0	0	0
⋮	⋮	⋮	⋮	⋮
x_2y_1	0	0	0.763	0
⋮	⋮	⋮	⋮	⋮
x_2y_2	0	0	0.744	0
⋮	⋮	⋮	⋮	⋮
$\dot{y}_2^2$	0	0	0	0

Figure 1: Sparse regression coefficients after refinement training for TMTD system. 12 terms are left after refinement training and post-processing is needed to achieve the discovery.

with $p_1 = -0.1$, $p_2 = 2$, $p_3 = -2$ and $p_4 = -0.1$. The tested Cubic Oscillator is a free oscillator and has no external input. In the data set, the simulated Cubic Oscillators have initial conditions with $x(0), y(0) \in [-1.2, 1.2]$ and the trajectories with larger absolution value of $x(0), y(0)$ less than 1.1 are removed with 64 trajectories left in total. Fig. 2 shows the snapshots of a video example. Hyperparameters used for training Cubic Oscillator model are shown in Table 4. After pre-training, the autoencoder reconstruction loss is 5.5×10^{-6}, the forward and backward frame reconstruction loss 1.8×10^{-4}, and physical state derivative loss 0.15. Then the network was kept training with total loss for 300 epochs. After total loss training, the autoencoder reconstruction loss is 3.7×10^{-6}, forward and backward frame reconstruction loss 5.9×10^{-6} and physical state derivative loss 0.0051. For sparse regression coefficients, there are 6 values larger than 0.1. The trained total loss model

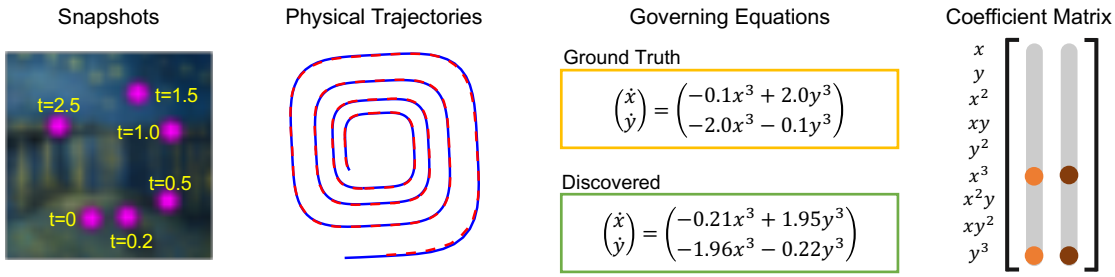

Figure 2: Testing on discovery of nonlinear dynamics (Cubic Oscillator). The ground truth physical trajectory is presented by solid lines, while the learned is presented by dash lines.

Table 4: Hyperparameter values for the Cubic Oscillator

Parameter	Value
video number	64
video length	1001
time step	0.05
q	3
batch size	2
learning rate	10^{-3}
Dynamic model order	1
Candidate function polynomial order	3
Physical states derivative loss $\mathcal{L}_{\dot{\mathbf{x}}_p}$ weight λ_1	1×10^{-3}
Forward/backward frame reconstruction loss $\mathcal{L}_{int}$ weight λ_2	1.0
Candidate function coefficient regularization loss $\mathcal{L}_{reg}$ weight λ_3	2×10^{-3}

is then loaded for the following sequential thresholding training. In sequential thresholding, the threshold value is 0.1 and sparse regression coefficients are thresholded every 100 epochs. After 300 epochs, the redundant terms are filtered out and only 4 terms kept. Finally, the network was kept training for 200 epochs with the left 4 candidate terms. The discovered closed-form equation for Cubic Oscillator is

$$\begin{pmatrix}\dot{x}\\ \dot{y}\end{pmatrix} = \begin{pmatrix}-0.33x^3 + 3.05y^3\\ -3.07x^3 - 0.35y^3\end{pmatrix} \tag{20}$$

Then the variable transformation is applied to the discovered equation to eliminate the effect of scaling. After transformation with scaling factor 1.28, the discovered equation can be rewritten as

$$\begin{pmatrix}\dot{x}\\ \dot{y}\end{pmatrix} = \begin{pmatrix}-0.21x^3 + 1.95y^3\\ -1.96x^3 - 0.22y^3\end{pmatrix} \tag{21}$$

5 Baseline Comparison

In this paper, two baselines are applied to discover the governing equations of the studied dynamical systems from videos. The first baseline is the coordinate and governing equation discovery method developed in [2]. Since this method is proposed to distill the governing equation from data without external input, a slight update is made to make this method applicable to the discovery

Table 5: Hyperparameters for SMSD system with the coordinate and governing equation discovery method

Parameter	Value
n	64×64
d	2
training samples	300×256
batch size	300×10
activation function	sigmoid
Encoder layer widths	512, 128, 32
Encoder layer widths	32, 128, 512
learning rate	1×10^{-3}
SINDy model order	1
SINDy library polynomial order	2
SINDy library includes sine	no
SINDy loss weight $\dot{x}$, λ_1	1×10^{-2}
SINDy loss weight $\dot{z}$, λ_2	1×10^{-3}
SINDy regularization loss weight, λ_3	1×10^{-5}

from data with input. In the update, the external unknown input is defined as a trainable array, and then the loss function is re-defined as

$$\mathcal{L} = \|\mathbf{x} - \psi(\mathbf{z})\|_2^2 + \lambda_1 \left\|\dot{\mathbf{x}} - (\nabla_{\mathbf{z}}\psi(\mathbf{z}))(\boldsymbol{\Theta}(\mathbf{z}^T)\boldsymbol{\Xi})\right\|_2^2 + \lambda_2 \left\|(\nabla_{\mathbf{x}}\mathbf{z})\dot{\mathbf{x}} - \boldsymbol{\Theta}(\mathbf{z}^T)\boldsymbol{\Xi} - \mathbf{g}\right\|_2^2 + \lambda_3 \|\boldsymbol{\Xi}\|_1 \quad (22)$$

where $\|\mathbf{x} - \psi(\mathbf{z})\|_2^2$ is the reconstruction loss, $\left\|\dot{\mathbf{x}} - (\nabla_{\mathbf{z}}\psi(\mathbf{z}))(\boldsymbol{\Theta}(\mathbf{z}^T)\boldsymbol{\Xi})\right\|_2^2$ is the SINDy loss in $\dot{\mathbf{x}}$, $\left\|(\nabla_{\mathbf{x}}\mathbf{z})\dot{\mathbf{x}} - \boldsymbol{\Theta}(\mathbf{z}^T)\boldsymbol{\Xi} - \mathbf{g}\right\|_2^2$ is the SINDy loss in $\dot{\mathbf{z}}$, and $\|\boldsymbol{\Xi}\|_1$ is the SINDy regularization. $\mathbf{g}$ is the defined trainable array representing the unknown external input. Besides, the original colorful video is converted into gray scale at first to feed the data into the network. The hyperparameters for testing this method on SMSD system is given in Table 5. The network was trained for 300 iterations with total loss function. After total loss training, the reconstruction loss and SINDy loss in $\dot{\mathbf{z}}$ are optimized into small values which are 2×10^{-4} and 4×10^{-7}. while the SINDy loss is 0.34 and it is almost constant along the training. The learned candidate function coefficient matrix given in Fig. 5 does not show sparsity.

The second baseline is from the designed method in which the video frame is condensed by the conventional convolutional encoder-decoder. The encoder has several convolutional layers and pooling layers to condense the high-dimensional image into low-dimensional latent variable. The decoder is made up of several transpose convolutional (or deconvolutional) layers to reconstruct the video frame. Because in this baseline, the encoder-decoder is expected to extract the physical states directly, the module for spatial to physical transformation is removed. The physical law constrain is still imposed to the latent variable. The network training is still conducted in 4 steps, which follows the same produce as our method. Hyperparameters used to test this baseline on discovering the governing equation of SMSD system are given in Table 6. In the pre-training, the autoencoder reconstruction loss is reduced to 3.3×10^{-5}, which means that the convolutional encoder-decoder is able to condense the video frame into latent variables. Because the physical states derivative loss after pre-training is quite large, about 650.00, in order to keep the learned encoder-decoder, the physical states derivative loss weight λ_1 is set from 1×10^{-5} to 5×10^{-3} gradually with hundreds of training iteration for each weight. After total loss training with the physical states derivative loss weight of 5×10^{-3}, the autoencoder loss, physical states derivative

Table 6: Hyperparameter values for the convolutional encoder-decoder baseline

Parameter	Value
Video number	256
Video length (unknown input length)	300
Time step	1/10.0
Batch size	4
Learning rate	1×10^{-3}
Dynamic model order	2
Candidate function polynomial order	2
Physical states derivative loss $\mathcal{L}_{\dot{\mathbf{x}}_p}$ weight λ_1	10^{-5}, 10^{-4}, 10^{-3}, 5×10^{-3}
Forward/backward frame reconstruction loss $\mathcal{L}_{int}$ weight λ_2	1.0
Candidate function coefficient regularization loss $\mathcal{L}_{reg}$ weight λ_3	1×10^{-5}

loss and forward frame reconstruction loss are 2.0×10^{-5}, 0.0020 and 2.0×10^{-5}. While the learned candidate function coefficient matrix given in Fig. 5 has no sparsity. Because the latent variable extracted from convolutional encoder-decoder does not represent the location-based physical states that present the underlying physical law. Moreover, the relationship between the physical states and real locations of the moving object(s) cannot be guaranteed by the conventional autoencoder.

Supplementary References

[1] Miguel Jaques, Michael Burke, and Timothy Hospedales. Physics-as-inverse-graphics: Unsupervised physical parameter estimation from video. In *International Conference on Learning Representations*, 2020.

[2] Kathleen Champion, Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45):22445–22451, 2019.