MeLON: Mixed Integer Latent Observer Networks

Abhishek Cauligi
Department of Aeronautics & Astronautics
acauligi@stanford.edu

Benoit Landry
Department of Aeronautics & Astronautics
blandry@stanford.edu

Abstract

Planning and control for discontinuous, nonlinear systems using output feedback control is a limited area of study. However, such approaches are crucial for enabling many robotic tasks in environments where full-state information is unavailable. In this work, we introduce Mixed Integer Latent Observer Networks (MeLON), an approach that learns piecewise-affine dynamics directly from images. Our algorithm leverages results in learning of latent dynamics using variational autoencoders, and introduces a new parameterization of the latent space that opens the door to a richer set of feedback control techniques without sacrificing its expressiveness. We demonstrate our algorithm on three different systems, and show that MeLON can match predictions from a state-of-the-art model, opening the door to exciting future work that leverages MeLON along with control in latent space based on mixed-integer programming.

1. Introduction

Planning and control for systems with contact is a challenging problem and one that has been extensively studied within the fields of robotics and controls. While most of the proposed approaches consider full state-information, such information may not be available for use in field or remote applications. Instead, such applications operate using partial state or output feedback using sensors such as RGB cameras, LIDAR, or force/torque sensors. Thus, planning and control techniques for contact systems must be capable of incorporating feedback information.

However, output feedback control for hybrid or discontinuous systems faces its own set of challenges. The functional mappings from an observation to state may be highly nonlinear, discontinuous, or both. Further, approaches using local linearizations using analytical models may not be sufficiently expressive, making it difficult to capture long-horizon behavior and predictions for such systems. Thus, in this work, we are interested in exploring data-driven approaches to approximate such functional mappings for use in the control of hybrid and discontinuous systems.

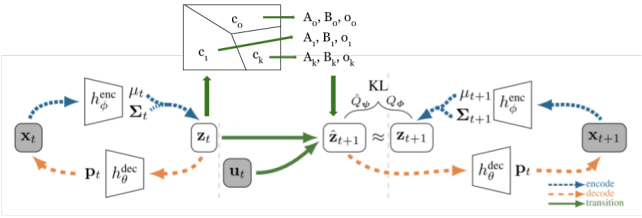


Figure 1: Illustration of the proposed architecture (figure adapted from [20]).

Specifically, we consider such a problem of learning image-conditioned dynamics models for underactuated, contact-rich systems with the goal of using such models with downstream feedback controllers. To this end, we propose mixed-integer latent observer networks (MeLON) as an approach to learn encodings of images into a latent observation space in which the observed dynamics behave like a piecewise-affine system. The key contributions of MeLON are two-fold:

1. We propose a novel formulation for learning piecewise-affine dynamics for nonlinear and discontinuous systems conditioned on images.

2. We demonstrate the efficacy of our proposed approach on a set of problems with hybrid dynamics.

2. Related Work

Systems with contact appear in a wide variety of domains and applications within the field of robotics. However, the inherently combinatorial nature of these problems make them $\mathcal{NP}$-hard and challenging to solve in real-time. Although a variety of problem formulations have been proposed in the areas of trajectory optimization and control for systems with contact [2, 10], relatively little attention has been paid towards control of discontinuous and hybrid systems using partial state or output feedback.

Recently, learning-based approaches have also been used to accelerate finding solutions for model-based, mixed-integer controllers. Both [3, 9] train a classifier to pre-

dict mode sequences for robotics tasks and [14] proposes a similar binary regression-based approach. In [7], heuristics for solving a non-convex mixed-integeer program are learned for a manipulation task encoded on observations of the scene. However, in all of these approaches, the underlying controller relies on full state information and disregards the sensor observation or output.

Learning image-conditioned dynamics models for continuous systems is a nascent area of research. To the authors knowledge, one of the first such approaches was presented in [11], where the authors use deep autoencoders to train a Q-learning algorithm for driving a model racecar around a track. More recently, Suh et al. [18] learn a set of linear dynamical models directly in pixel space for robot pushing tasks. However, rather than considering a mixed-integer control formulation, Suh et al. use a greedy approach to satisfy the sufficient decrease condition for a Lyapunov-based controller. Closest to our approach, [20] proposed to leverage variational autoencoders to learn linear dynamics in an embedded space. Our proposed algorithm can be understood as a direct extension of this work, where the embedded dynamics are piecewise-affine instead of linear, enabling the use of a richer set of control methodologies in latent space.

Finally, our proposed approach of learning a piecewise affine partition in latent space also draws inspirations from the field of non-parametric meta-learning. In [17], the authors learn a Voronoi partitioning of the latent space using an encoder and demonstrate its efficacy in generalizing to new classification tasks. [12] similarly learns a linear classifier in latent space. Both approaches rely on the idea of constructing a linear partition in a learned feature space.

3. Methods

3.1. Notation

We briefly review the notation used through the remainder of this report. We use v_k to denote a vector v at time step k and v_k^i for the i'th component of v_k. For a dynamical system, we refer to the underlying state representation as $s_k \in \mathbb{R}^{n_s}$. The image x_k is generated using the state vector s_k and is of dimension n_x. Using a slight abuse of notation, we also consider the image x_k as a tensor of dimension $\mathbb{R}^{C\times W\times H}$, where C, W, and H are the number of channels, width, and height of the image, respectively. Typically, $C = 3$ for an RGB camera input and $C = 4$ for an RGB-D input. Finally, u_k and z_k are the control input for the dynamical system and latent vector, respectively.

3.2. Piecewise-Affine Systems

For our approach, we propose explicitly learning piecewise-affine dynamical systems [4] in a semi-supervised learning fashion. Piecewise-affine dynamical (PWA) systems partition the state space into polytopes in which the dynamics are constrained to be linear:

$$z_{k+1} = A_i z_k + B_i u_k \text{ if } \begin{pmatrix} z_k \\ u_k \end{pmatrix} \in \mathcal{D}_i$$

where $z_k \in \mathbb{R}^{n_z}$ and $u_k \in \mathbb{R}^{n_u}$ are the state and control, respectively, and $\mathcal{D} = \cup_{i=1}^{N_d} \mathcal{D}_i$ is a polyhedral partition of the state and action space. Each polytope $\mathcal{D}_i$ is defined according to a series of linear inequality constraints:

$$\mathcal{D}_i = \{(z, u) \in \mathbb{R}^{n_z+n_u} \mid G_i^z z + G_i^u u \leq c_i\}$$

where the affine inequality constraints defined by G_i^z, G_i^u, and c_i implicitly encode linear state and control constraints $z \in \mathcal{Z}$ and $u \in \mathcal{U}$ present in the physical system.

3.3. Variational Autoencoders

The field of generative modeling has seen an increase in popularity for modeling and predicting robot dynamics [13, 16]. For this work, we use variational autoencoders (VAE) as our learning framework to find expressive encodings of the images [6]. Given an image $x \in \mathbb{R}^{n_x}$, we seek to learn a latent representation $z \in \mathbb{R}^{n_z}$ that simultaneously captures a low-dimensional, task-relevant feature encoding and is expressive enough to learn transition dynamics in the latent space $\mathcal{Z}$. Here, we briefly review the theory and notation of VAEs, but refer the reader to [8] for a more thorough review.

In accordance with the standard approaches for using VAEs, we seek to estimate a posterior distribution $p(z|\mathcal{D}$ given data $\mathcal{D}$, where $\mathcal{D}$ consists of observed images x. However, computing the true posterior $p(z|x)$ is intractable as it would entail integrating over the entire latent space z. Instead, a new distribution $q(z|x)$ is introduced to approximate the posterior, with the use of the KL-divergence as a pseudo-metric to find a valid approximation of $p(z|x)$. Finally, we assume that the inference and generative distributions belong to a parametrized set of distributions such that we can use neural networks with parameters ϕ and ψ to approximate $q_\phi(z|x)$ and $p_\theta(x|z)$.

For learning transition dynamics, we also consider the transition distribution at the next time step $\hat{q}_\psi(z_{k+1}|\mu_k, u_k)$. In this work, $q_\phi(z|x)$ and $\hat{q}_\psi(z_{k+1}|\mu_k, u_k)$ are assumed to be Gaussian distributions with means μ_k and μ_{k+1} and diagonal covariances Σ_k and Σ_{k+1}, respectively.

3.4. Transition Dynamics

Given a latent representation $z_k \in \mathbb{R}^{n_z}$ of our image x_k, we seek to learn transition dynamics of the form:

$$z_{k+1} = A^{(i)} z_k + B^{(i)} u_k + o^{(i)} \quad i \in \{1, \dots N_d\} \qquad (1)$$

Because this update rule is not a differentiable operation, in order to perform backpropagation over our loss term, we

instead consider a convex combination of the N_d transition mappings:

$$z_{k+1} = \sum_{i=1}^{N_d} \alpha^{(i)}(A^{(i)}z_k + B^{(i)}u_k + o^{(i)})$$

where $\alpha^{(i)} \geq 0$ and $\sum_{i=1}^{N_d} \alpha^{(i)} = 1$. We compute the values $\alpha^{(i)}$ by performing a softmax operation on the output of a linear classifier operating on z_k:

$$\alpha^{(i)} = \frac{\exp\left(\beta w^{(i),T} z_k\right)}{\sum_{j=1}^{N_d} \exp\left(\beta w^{(j),T} z_k\right)}$$

where $w^{(i)}$ are the rows of a matrix $W \in \mathbb{R}^{N_d \times n_z}$. Note that we also introduce the scalar parameter β, which corresponds to the inverse of the *temperature*

$$\beta = \frac{1}{T + \epsilon},$$

where T is the temperature and ϵ is a small constant introduced for numerical stability (in practice we train the square root of the temperature to enforce positivity). We then add the temperature of the softmax function to the loss function to encourage the piecewise-affine model to learn strict boundaries between each mode during training. Importantly, at test time, we replace the softmax of the latent classifier by a hard max, propagating a single mode at a time as defined by equation (1).

Note that many different classifiers could be used in latent space. However, our specific choice of a linear classifier is motivated by the fact that we can later recover the explicit description of the polyhedral partitions corresponding to modes in $\mathcal{Z}$. Thus, a mode $i \in \{1, \ldots, N_d\}$ is "active" when:

$$w^{(i),T} z_k \geq w^{(j),T} z_k$$

for all $j = 1, \ldots, N_d, j \neq i$. This explicit partitioning of the latent space will later enable the straightforward use of control approaches based on mixed-integer programming.

3.5. Algorithm Overview

The offline portion of MeLON is described in Algorithm 1. Given a dataset $\mathcal{D} = \{(x_k, u_k, x_{k+1})\}$, where each tuple consists of a pair of images x_k, x_{k+1} and u_k, the complete loss function we seek to optimize is [20]:

$$\mathcal{L}(\mathcal{D}) = \sum_{x_k, u_k, x_{k+1}} \mathcal{L}^{\text{gen}}(x_k, u_k, x_{k+1})$$
$$+\lambda_{\text{trans}} \operatorname{KL}(\hat{q}_\psi(z_{k+1}|\mu_k, u_k) \| q_\phi(z_k|x_k)) + \lambda_{\text{temp}} T \quad (2)$$

where λ_{trans} and λ_{temp} are parameters that trade-off the three loss terms. The first loss $\mathcal{L}^{\text{gen}}$ is:

$$\mathcal{L}^{\text{gen}}(x_k, u_k, x_{k+1}) = \mathbb{E}_{z_k \backsim q_\phi,\, z_{k+1} \backsim \hat{q}_\psi}\big[-\log p_\theta(x_k|z_k)$$
$$-\log p_\theta(x_{k+1}|z_{k+1})\big] + \operatorname{KL}(q_\phi \| p(z_k))$$

Algorithm 1 MeLON Offline

Require: Batch of training data $\mathcal{D} = \{(x_k, u_k, x_{k+1})\}_{i=1}^{N_s}$ and batch size $N_b \leq N_s$

1: Initialize network weights θ, ϕ and ψ for VAE and transition dynamics
2: **for** each batch $\mathcal{B} = \{(x_k, u_k, x_{k+1})\}_{i=1}^{N_b}$ **do**
3: Compute loss $\mathcal{L}(\mathcal{B})$ using (2)
4: Update weights θ,ϕ, $\psi \leftarrow \text{ADAM}(\mathcal{L}(\mathcal{B}), \theta, \phi, \psi)$
5: **end for**
6: **return** Trained network parameters θ,ϕ, ψ

where $\log p_\theta(x_k|z_k)$ is the mean squared error loss from the output of the generative model p_θ.

The second term of this loss function is a KL-divergence term that induces finding an inference model $q_\phi(z_k|x_k)$ that matches the analogous distribution $\hat{q}_\psi(z_{k+1}|\mu_k, u_k)$ at the next time step. Finally, the third loss penalizes T, the non-negative temperature term described in 3.4

4. Dataset

For simplicity, our preliminary analysis of MeLON considers only autonomous systems without control inputs. The three systems considered are: the half-space single integrator for ball, the bouncing ball, and cart-pole with contact. Table 1 provides an overview of the dimensionality of the three systems considered.

System	n_x	n_z	N_d
Half-Space Integrator	$3 \times 32 \times 32$	6	2
Bouncing Ball	$3 \times 128 \times 128$	6	4
Cart-Pole	$3 \times 64 \times 64$	8	3

Table 1: Overview of system dimensions in numerical experiments.

4.1. Half-Space Single Integrator

The half-space single integrator system consists of a ball in 2D moving with constant velocity. The state $s_k \in \mathbb{R}^4$ for this system consists of position coordinates p_k and velocity v_k. The workspace bounds are $p_{\min} \leq p_k \leq p_{\max}$. We denote this system half-space single integrator because it obeys single integrator dynamics but with the update condition that the horizontal velocity $v_k^{(1)}$ is negated if $p_k^{(1)} \leq 0.5(p_{\min}^{(1)} + p_{\max}^{(1)})$. The velocity is held constant $v_{k+1} = v_k$ and the position update is $p_{k+1} = p_k + \Delta h v_k$, where the sign of v_k is updated according to position bounds. For this system, we construct a training set of 500 transition tuples and a test set of 100 transition tuples.

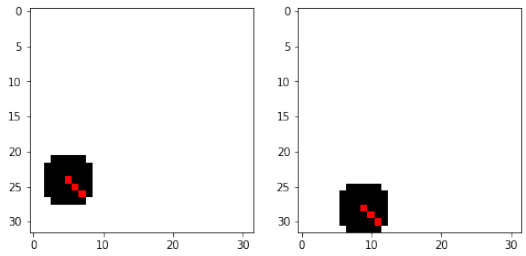


Figure 2: Example transition step for the single integrator with constant velocity. The red pixels are used to indicate the constant velocity vector.

4.2. Bouncing Ball

The bouncing ball system is similar to the half-space single integrator in that it consists of a ball moving in 2D with a state $s_k \in \mathbb{R}^4$. However, rather than considering PWA single integrator dynamics in the interior of the workspace, we use the same single integrator model in the interior and are only interested in simulating collision with the four walls of the workspace. For this system, we construct a training set of 500 transition tuples and a test set of 100 transition tuples.

4.3. Cart-Pole with Contact

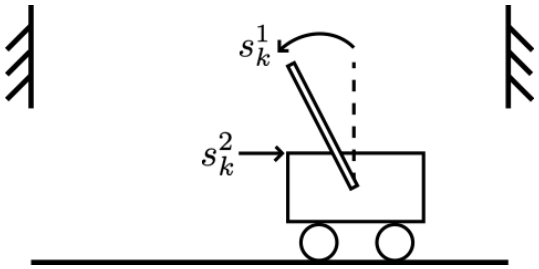


Figure 3: 4D cart-pole with wall system.

The state $s_k \in \mathbb{R}^4$ for the cart-pole with contact system consists of the position of the cart s_k^1, angle of the pole s_k^2, and their derivatives s_k^3 and s_k^4, respectively. For this system, contact occurs when the pole makes contact with the left or right wall as shown in Figure 3.

We model this system using `PyBullet` [5], which can simulate the rigid bodies as well as the elastic collision between the wall and the pole. We include a virtual camera in the simulation that produces images which we use as dataset. A single sample consists of three, 64 by 64, consecutive images taken at 100ms of each other. We generated a total of 3000 samples for the cart-pole system and retained 10 percent of those as test set.

5. Results

Our machine learning models are implemented in `PyTorch` [15]. Our code builds upon an existing implementation found at `https://github.com/ethanluoyc/e2c-pytorch`. However, we extend the fully connected neural network implementation to accommodate convolutional networks for better performance. The code for our project is available at `https://github.com/acauligi/cs231n-project`.

5.1. Benchmarks

In addition to the transition dynamics from (1), we also implement two baselines. The first benchmark uses the transition dynamics from [20] of the form:

$$z_{k+1} = (I + r_{\phi_r}(z_k) v_{\phi_v}(z_k)^T) z_k + B_{\phi_B} u_k + o_{\phi_o} \quad (3)$$

where $r_{\phi_r}(z_k) : \mathbb{R}^{n_z} \to \mathbb{R}^{n_z}$ and $v_{\phi_v}(z_k) : \mathbb{R}^{n_z} \to \mathbb{R}^{n_z}$ are fully connected neural network function approximators. Further, $B_{\phi_B} \in \mathbb{R}^{n_z \times n_u}$ and $o \in \mathbb{R}^{n_z}$ are additional learned parameters. The second benchmark entails learning a single set of linear dynamics:

$$z_{k+1} = A_{\phi_A} z_k + B_{\phi_B} u_k + o_{\phi_o}, \quad (4)$$

where $A_{\phi_A} \in \mathbb{R}^{n_z \times n_z}$, $B_{\phi_B} \in \mathbb{R}^{n_z \times n_u}$ and $o_{\phi_o} \in \mathbb{R}^{n_z}$ are parameters that are learned directly. Note that this latter linear approximation differs from learning PWA dynamics with $N_d = 1$ due to learning an additional α^1 parameter.

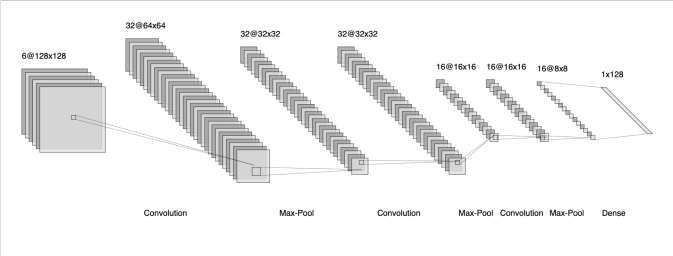
5.2. Numerical Experiments

As the velocity for the half-space single integrator system is held-constant, each transition tuple is simply $\{x_k, x_{k+1}\}$. However, because the velocity of a system cannot be determined from a static image, for the bouncing ball and cart-pole with contact systems, each data tuple also constains the image at a previous time step i.e. $\{x_{k-1}, x_k, x_{k+1}\}$. Thus, the input to the encoder network is a six channel RGB image for the bouncing ball and cart-pole with contact systems.

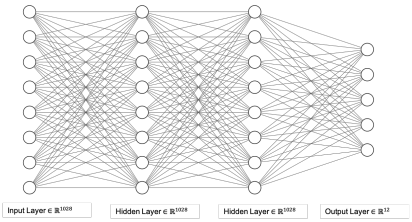
5.2.1 Hyperparameter Tuning

For this algorithm, we have two sets of design choices: the hyperparameters for MeLON from Algorithm 1 and the neural network architectures for the inference, generative, and transition models.

For optimizing the network, we use Adam with Pytorch's default parameters. We used encoder and decoder architectures similar to the ones presented in the original E2C paper. Moreover, we used batch normalization where applicable in order to reduce our models' dependencies to weight initialization. In general, we tuned the remainder of the hyperparameters by first overfitting a small dataset, and then in-



(a) Convolution component of encoder network



(b) Feedforward component of encoder network

Figure 4: The encoder used for the single integrator consists of a series of convolutional and max-pool operations. The flattened output is then passed to a feedforward neural network that has an output dimension of $\mathbb{R}^{2n_z}$.

creased the training set size as well as regularization terms while monitoring performance on the validation set.

Figure 4 depicts the encoder network used for the half-space single integrator. In general, for each system, we follow a similar approach of first using convolution, activation, max-pool, and batch norm operations before flattening the output. Therafter, we use a dense feedforward network with activations in each layer with an output dimension of $2n_z$, as we predict both the mean and covariance of $q(z|x)$. The decoder network used for each system is identical to the encoder except for the use of the convolutional transpose operator in place of a convolution operation.

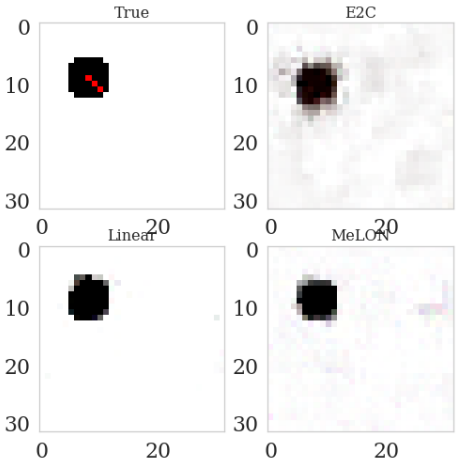


Figure 5: An example of reconstructing the transition image. We see that the although a $\max$ term is enforced in the MeLON prediction, training may not be leading to sparsity in the softmax terms.

Figure 8 shows the reconstruction ℓ_2-norm loss for the three models on the test set. Figure 7 also shows a predicted samples for each of the models.

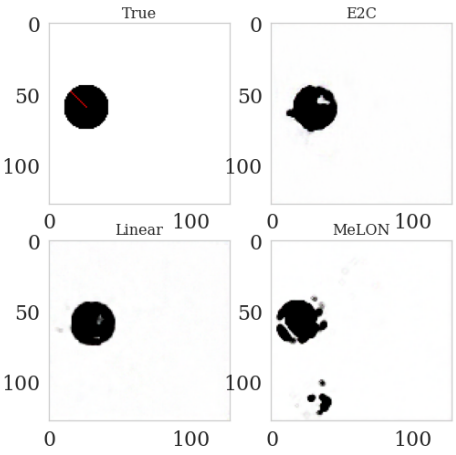


Figure 6: Predictions for the bouncing ball system also demonstrate the issue of discrepancies in using softmax for training and max at test time

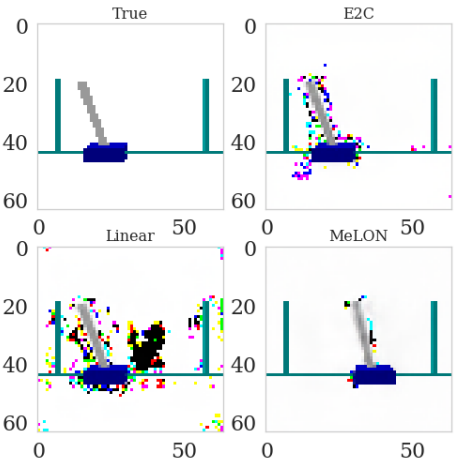
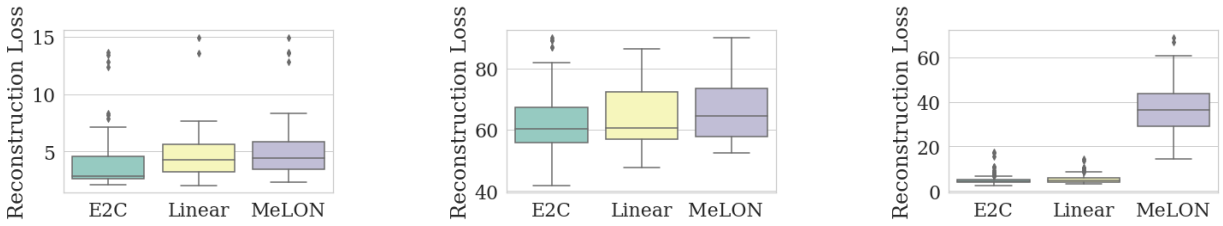


Figure 7: Predictions for the cartpole system.

6. Discussion

Numerical results seem to indicate that for simple systems (i.e. the half-space single integrator and the bouncing ball), MeLON is able to match the prediction quality of a state-of-the-art model such E2C. However, the ability of the linear latent space model to also approach this level of reconstruction fidelity might suggest that piecewise nature of the latent space only provides marginal improvement. We



(a) Half-space single integrator (b) Bouncing ball (c) Cart-pole

Figure 8: MeLON performs comparably to the benchmarks for half-space single integrator and bouncing ball, but less well on cart-pole.

also note that predictions for the more complicated system (the cartpole) are significantly worse using MeLON.

However, inspection of the failure cases of MeLON on the more challenging task provides important insight. Indeed, many of the predictions seem to correspond to valid future states, were the underlying system to be in a different discrete mode.

Moreover, evaluating the test set without enforcing hard boundaries between each mode (i.e. using the same architecture for testing than was used for training) produces results of high quality (Figure 9 and 10). This seems to suggest that it might be possible to greatly improve MeLON by investigating better strategies to enforce hard boundaries in latent space. Those can include, for example, a curriculum on the weight given to the temperature in the loss function. Moreover, it would be worth investigating applying the linear classifier in latent space not to the sample z, but to the *distribution* $q(z|x)$ from which z is sampled, in order to prevent the sampling from switching the discrete mode.

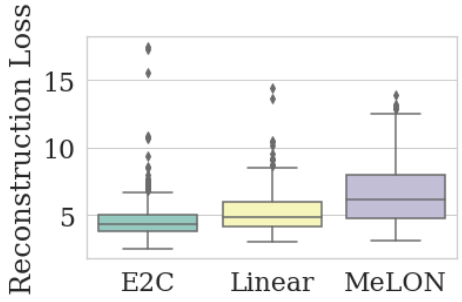


Figure 9: Predictions errors for the cartpole system using soft boundaries between the modes in latent space.

7. Conclusion and Future Work

In this work, we proposed MeLON, an algorithm that learns PWA-dynamics in latent space given image observations of the system. As we demonstrate, MeLON is a promising framework for developing dynamics models in latent space in a manner that is amenable for use with a rich set of tools from optimal control theory. For the simpler half-space single integrator and bouncing ball systems, MeLON performs comparably to the state-of-the-art E2C framework. However, we find that MeLON falls short for the more complex cart-pole with contact system. We believe that the discrepancy between training and test time is due to the use of a max operation at test time. Thus, we would like to investigate further a curriculum on the temperature T and training a linear classifier on the distribution from which z is sampled such that the classification boundaries in latent space are more strictly enforced.

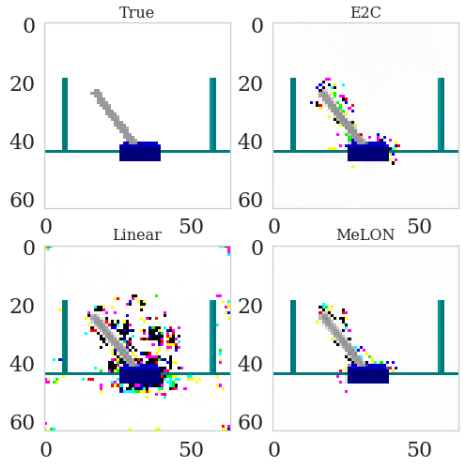


Figure 10: Predictions for the cartpole system using soft boundaries between the modes in latent space.

Even though the results presented here focus on the ability of MeLON to predict future observations, the main strength of our proposed approach lies in the rich family of feedback controllers its latent space is suited for. As a part of our future work, we propose integrating the learned PWA-affine dynamics with a hybrid MPC controller in latent space. This can be achieved by introducing Boolean variables $\delta^{(i)} \in \{0, 1\}$ for each of the N_d partitions and using big-M notation to enforce that only one mode is active at each time step [1]. We note that a complicating factor for control in latent space is that the decoder is not necessarily an injective mapping. That is, given a desired goal state s_g,

there may exist a set of latent states $\{z_g\}$ for which the image decoding has the equivalent ℓ_2-distance with the goal image x_g. Specifying the "correct" z_g for a controller may require further consideration than simply encoding x_g. To tackle the issue of long-horizon prediction errors, we would also like to augment the loss term with a longer horizon as described in [19].

Acknowledgements

We would like to thank Karen Leung, James Harrison, and Edward Schmerling of the Autonomous Systems Lab for their feedback during the development of this project.

References

[1] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 1999.

[2] J. Carius, R. Ranftl, V. Koltun, and M. Hutter. Trajectory optimization with implicit hard contacts. *IEEE Robotics and Automation Letters*, 3(4):3316–3323, 2018.

[3] A. Cauligi, P. Culbertson, B. Stellato, D. Bertsimas, M. Schwager, and M. Pavone. Learning mixed-integer convex optimization strategies for robot planning and control. In *Proc. IEEE Conf. on Decision and Control*, 2020.

[4] F. Christophersen. *Optimal Control of Constrained Piecewise Affine Systems*. Springer Berlin Heidelberg, first edition, 2007.

[5] E. Coumans and Y. Bai. Pybullet, a Python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2019.

[6] C. Doersch. Tutorial on variational autoencoders, 2016. Available at `https://arxiv.org/abs/1606.05908`.

[7] D. Driess, O. Oguz, J.-S. Ha, and M. Toussaint. Deep visual heuristics: Learning feasibility of mixed-integer programs for manipulation planning. In *Proc. IEEE Conf. on Robotics and Automation*, 2020.

[8] S. Ermon. CS 236: Deep generative models, 2020. Available at `https://deepgenerativemodels.github.io/notes/index.html`.

[9] F. R. Hogan, E. R. Grau, and A. Rodriguez. Reactive planar manipulation with convex hybrid MPC. In *Proc. IEEE Conf. on Robotics and Automation*, 2018.

[10] B. Landry, Z. Manchester, and M. Pavone. A differentiable augmented lagrangian method for bilevel nonlinear optimization. In *Robotics: Science and Systems*, 2019.

[11] S. Lange and M. Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In *IEEE Int. Joint Conference on Neural Networks*, 2020.

[12] K. Lee, S. Maji, A. Ravichandran, and S. Soatto. Meta-learning with differentiable convex optimization. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2019.

[13] M. A. Lee, Y. Zhu, P. Zachares, M. Tan, K. Srrinivasan, S. Savarese, F.-F. Li, and J. Bohg. A sampling-based tree planner for systems with complex dynamics. *IEEE Transactions on Robotics*, 2020.

[14] D. Masti and A. Bemporad. Learning binary warm starts for multiparametric mixed-integer quadratic programming. In *European Control Conference*, 2019.

[15] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. In *Conf. on Neural Information Processing Systems - Autodiff Workshop*, 2017.

[16] E. Schmerling, K. Leung, W. Vollprecht, and M. Pavone. Multimodal probabilistic model-based planning for human-robot interaction. In *Proc. IEEE Conf. on Robotics and Automation*, 2018.

[17] J. Snell, J. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *Conf. on Neural Information Processing Systems*, 2017.

[18] H. J. T. Suh and R. Tedrake. The surprising effectiveness of linear models for visual foresight in object pile manipulation, 2020. Available at `https://arxiv.org/abs/2002.09093`.

[19] Y. Wang, K. Caluwaerts, A. Iscen, T. Zhang, J. Tan, and V. Sindhwani. Data efficient reinforcement learning for legged robots. In *Conf. on Robot Learning*, 2019.

[20] M. Watter, J. T. Springenberg, J. Boedecker, and M. A. Riedmiller. Embed to Control: A locally linear latent dynamics model for control from raw images. In *Conf. on Neural Information Processing Systems*, 2015.