Nested Mixture of Experts: Cooperative and Competitive Learning of Hybrid Dynamical System

Junhyeok Ahn   JUNHYEOKAHN91@UTEXAS.EDU
Department of Mechanical Engineering, The University of Texas at Austin, USA

Luis Sentis   LSENTIS@AUSTIN.UTEXAS.EDU
Department of Aerospace Engineering and Engineering Mechanics, The University of Texas at Austin, USA

Abstract

Model-based reinforcement learning (MBRL) algorithms can attain significant sample efficiency but require an appropriate network structure to represent system dynamics. Current approaches include white-box modeling using analytic parameterizations and black-box modeling using deep neural networks. However, both can suffer from a bias-variance trade-off in the learning process, and neither provides a structured method for injecting domain knowledge into the network. As an alternative, gray-box modeling leverages prior knowledge in neural network training but only for simple systems. In this paper, we devise a nested mixture of experts (NMOE) for representing and learning hybrid dynamical systems. An NMOE combines both white-box and black-box models while optimizing bias-variance trade-off. Moreover, an NMOE provides a structured method for incorporating various types of prior knowledge by training the associative experts cooperatively or competitively. The prior knowledge includes information on robots' physical contacts with the environments as well as their kinematic and dynamic properties. In this paper, we demonstrate how to incorporate prior knowledge into our NMOE in various continuous control domains, including hybrid dynamical systems. We also show the effectiveness of our method in terms of data-efficiency, generalization to unseen data, and bias-variance trade-off. Finally, we evaluate our NMOE using an MBRL setup, where the model is integrated with a model-based controller and trained online.

Keywords: Learning of Hybrid Dynamical Systems, System Identification, Model-based Control

1. Introduction

Reinforcement learning algorithms provide an automated framework for decision making by maximizing an objective function and learning a control policy. However, model-free reinforcement learning algorithms require a vast amount of data for learning policies, which often limits their application to simulated domains (Fujimoto et al. (2018); Schulman et al. (2017)). An alternative method for reducing sample complexity is to explore model-based reinforcement learning (MBRL) methods, which involve obtaining a predictive model of the world and then using that model to make decisions (Chua et al. (2018); Nagabandi et al. (2018); Levine and Abbeel (2014)).

In MBRL, model representation is one of the most critical design choices and determines the performance and efficiency of the algorithm. One option is to treat dynamics as a function where machine learning is capable of fitting. For example, Chua et al. (2018); Nagabandi et al. (2018); Raissi et al. (2018) deployed deep neural networks and took a data-driven, black-box approach to approximate dynamics. On the other hand, some works utilize a white-box approach that exploits a priori domain knowledge, make assumptions about kinematic and dynamic properties, and leave a few parameters for identification. For instance, An et al. (1985); Ahn et al. (2019) conducted

classical system identification to learn inertial parameters, and Bai et al. (2013) integrated model learning and planning using Bayesian reinforcement learning. The black-box approach relies on the network's representational power but often requires tremendously large amounts of training data to achieve generalization and suffers from model variance. The assumptions in the white-box approach may not capture hard-to-model effects, leading to model bias, although they enable training with much less data.

Other studies leverage a gray-box method to mitigate the limitations. Lutter et al. (2019); Gupta et al. (2020) and Sutanto et al. (2020) proposed a network topology that encoded the Euler-Lagrange and Newton-Euler equation, respectively, and demonstrated that their model is trained efficiently and extrapolates effectively to unseen samples. Greydanus et al. (2019); Zhong et al. (2020) presented a dynamics model that conformed to the Hamiltonian equation and demonstrated that their model conserves a quantity analogous to the total energy. However, they had to predefine the dynamics equation structure in advance and train the model under the prescribed model capacity, which could still cause a model bias. Ahn et al. (2020); Koller et al. (2019); Carron et al. (2019) fitted residual dynamics with a Gaussian process based upon a prescribed analytic model while accounting for hard-to-model effects and reduced model bias. The aforementioned gray-box approaches are all limited to representing simple mechanical systems, such as a pendulum or robotic arms, where a single dynamic equation prescribes the behaviors over the state space. For hybrid dynamical systems that may include intricate prior knowledge, such as physical contacts with the environment, multiple dynamic models need to be considered instead of a single model.

Thus, the purpose of this paper is to study a gray-box approach that can be used for efficient learning of complex robotic systems, including hybrid systems. To that end, we propose a network structure, dubbed a nested mixture of experts (NMOE), to represent and learn dynamic models. Using this novel network structure, we propose a general method to incorporate various types of physical insights by training local experts to be cooperative or competitive. We demonstrate the effectiveness of our NMOE with useful metrics in various continuous control domains. We further evaluate our NMOE using an MBRL setup, where the model is integrated with a model-based controller and trained online.

2. Preliminaries

A mixture of experts (MOE) is a network composed of many separate feedforward sub-networks and ensembles outputs of the sub-networks to improve model performance in the supervised learning procedure (Jacobs et al. (1991)). It includes a gating network that decides what combination of the experts should be used for each data input. Note that the gating network is also feedforward and typically receives the same input as the expert networks. Figure 1 shows an example of an MOE with N local experts and a gating network g that computes outputs $y_i = \sum_{k=1}^{N} g_k(x_i) f_k(x_i)$,

Figure 1: Mixture of experts.

where $\sum_{k=1}^{N} g_k(x_i) = 1$. Here, x_i and y_i are the input and the output of the MOE given a dataset made of pairs $(x_i, y_{\text{target},i})$. The weight g_k corresponds to the kth element of g and indicates the probability (or so called responsibility) for the expert f_k. Jacobs et al. (1991) showed that each of the experts is trained to either cooperate or compete to produce the output based on a loss function. For example, a loss function with the expression $\sum_{i=1}^{D} \|y_{\text{target,i}} - y_i\|^2$, or equiv-
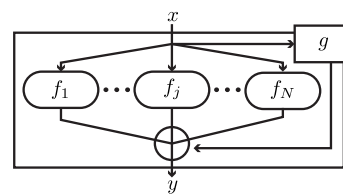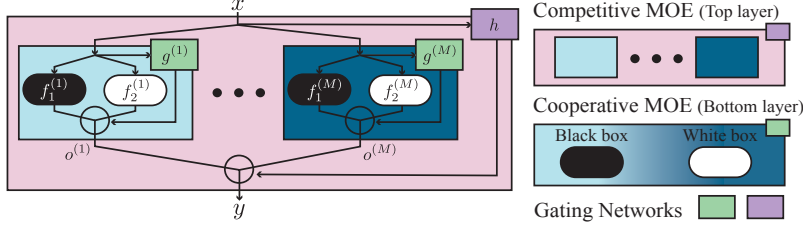
Figure 2: NMOE structure: The top layer forms a competitive MOE, and each of the experts on it are cooperative MOEs.

alently $\sum_{i=1}^{D}\|y_{\text{target,i}} - \sum_{k=1}^{N} g_k(x_i)f_k(x_i)\|^2$, makes the experts contribute to the output linearly, where D is the size of the dataset. With this loss function, each expert is trained to produce an output that eliminates the residual errors left by the other experts. When the local experts cooperate to generate the output, we call this a cooperative MOE. Conversely, another loss function $\sum_{i=1}^{D} \sum_{k=1}^{N} g_k(x_i)\|y_{\text{target},i} - f_k(x_i)\|^2$ makes each local expert produce the whole output rather than a residual, and respond only to a subset of the dataset. If we train the network with gradient descent using this loss function, the MOE tends to devote a single expert to each training case. As a result, each expert is not directly affected by the weights of other experts, which can be useful when we know in advance that a set of training data is naturally classified into particular expert categories. When the local experts compete to generate the output, we call this a competitive MOE.

3. Nested Mixture of Experts

3.1. Network Structure

As depicted in Figure 2, an NMOE has a nested structure of an MOE. The top layer (pink box) forms a competitive MOE with M local experts (blue boxes) and the gating network (purple box). As a nested structure, each of the local experts (blue boxes) forms a cooperative MOE containing two local experts (ellipsoids) and a gating network (green box)[1]. The ith local expert in the top layer (ith blue box), or equivalently ith cooperative MOE, computes an output as follows:

$$o^{(i)}(x) = \sum_{j=1}^{2} g_j^{(i)}(x)f_j^{(i)}(x),$$

$$g^{(i)}(x) = \text{softmax}(W_g^{(i)}x). \tag{1}$$

Here, x is the input, $o^{(i)}(x)$ is the output of the ith cooperative MOE, and $f_j^{(i)}(x)$ is the output of the jth expert in the ith cooperative MOE. $g_j^{(i)}(x)$, the jth element of $g^{(i)}(x)$, indicates the probability (or responsibility) for the expert and $W_g^{(i)}$ is a trainable weight. A competitive MOE (pink box)

---

1. Here, we only consider two local experts in the bottom layer, although an MOE can have any number of local experts.

combines the outputs of M cooperative MOEs (blue boxes) as follows:

$$y = \sum_{i=1}^{M} h_i(x) o^{(i)}(x),$$

$$h(x) = \text{softmax}(W_h x),$$

(2)

where y is the final output of the NMOE, and $h_i(x)$ is the ith element of $h(x)$, indicating the probability (or responsibility) for the output $o^{(i)}$ with a trainable weight W_h. Finally, we define a loss function as follows:

$$\text{loss}(x, y_{\text{target}}) = \frac{1}{M} \sum_{i=1}^{M} h_i(x) \left\| y_{\text{target}} - o^{(i)}(x) \right\|^2 = \frac{1}{M} \sum_{i=1}^{M} h_i(x) \left\| y_{\text{target}} - \sum_{j=1}^{2} g_j^{(i)}(x) f_j^{(i)}(x) \right\|^2,$$

so that the M local experts (blue boxes) in the top layer are trained to compete with each other, whereas the two local experts (ellipsoids) in the bottom layer are trained to cooperate. Due to this loss function, each of the cooperative MOEs (blue boxes) produces the whole of the output and correlates with a subset of the dataset that is determined by their responsibility (i.e., the output of the gating network h). In contrast, the two local experts (ellipsoids) inside the cooperative MOEs contribute to the output linearly, which is determined by their responsibility (i.e., the output of the gating network g). In summary, given the input x, the competitive MOE (pink box) assigns the input to a single cooperative MOE (blue box). The chosen cooperative MOE combines the outputs from its local experts (ellipsoids) to generate the final output y. Note that the input x corresponds to a state and action pair, and the output y corresponds to a next-state prediction when it comes to dynamic model representation.

3.2. Incorporating Prior Knowledge

In this section, we discuss how to constitute an NMOE to incorporate prior knowledge. We consider two different types of prior knowledge. The first type includes information on physical contacts between a system and an environment. In the case of a hybrid dynamical system, we know in advance that the input dataset is naturally divided into subsets based on the contact modes, and each of the subsets is governed by a different physics law. This leads us to set M, the number of experts in the competitive MOE, to be the number of contact modes. The gating network (purple box) divides input data into M subsets and assigns each of them to a local expert associated with the contact mode. A competitive MOE can significantly reduce the model complexity compared to using a single network since each local expert is associated with a subset of the dataset.

The second type of prior knowledge corresponds to system equations describing the robot's behavior with assumptions on rigid body kinematics and dynamics. We then compose a cooperative MOE (blue box) combining the system equations (i.e., the white-box model) and a deep neural network (i.e., black-box model) as local experts (ellipsoids). The white-box model provides a reasonable initial representation of the model and aids in achieving quick convergence of the learning process with low variance. In contrast, the black-box model eliminates the residual errors left by the hard-to-model effects and reduces the bias. The gating network (green box) computes a probability (or responsibility) based on the reliability of the prior knowledge.

Putting these together, we consider a structured method to incorporate prior knowledge into an NMOE. We first estimate the type of robot's contact modes, M. We then define a competitive MOE
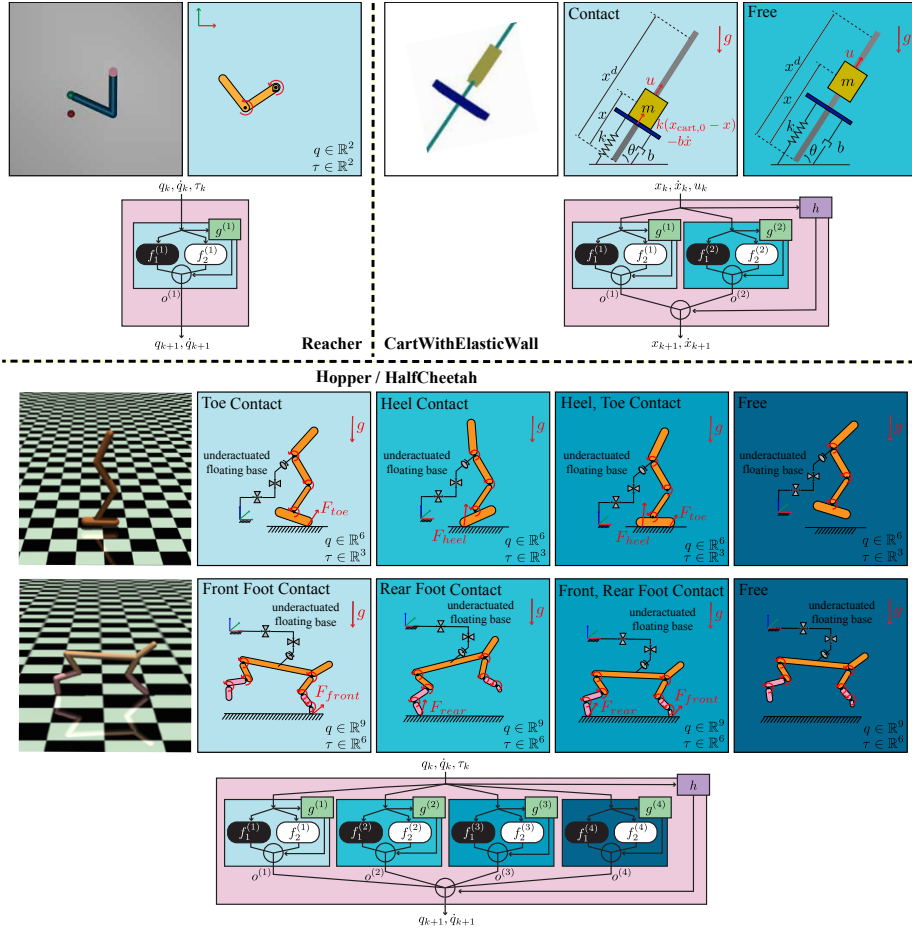
Figure 3: Illustration of physical contact modes and NMOE structures for various domains.

(pink box) with M local experts, each of them being cooperative MOEs (blue boxes). Each of the cooperative MOEs (blue boxes) includes a white-box (white ellipsoid) and a black-box model (black ellipsoid), where the white-box model is a differential equation of motion and the black-box model is a deep neural network.

4. Experimental Results

In this study, we use four different continuous control domains (Reacher, CartWithElasticWall, Hopper, and HalfCheetah) and address the following questions: How do we construct an NMOE for each domain based on prior knowledge? Does our NMOE framework outperform related baseline approaches in dynamic model learning? Does the top layer's gating network (purple box) successfully assign input data to a corresponding contact mode after convergence? Does the bottom layer's gating network (green box) properly combine white-box and black-box models based on the accuracy of the prior knowledge? In an MBRL setup, can we train NMOE online and provide reliable trajectory predictions to the model-based controller? Parameters used in dynamics model represen-

tations, model learning, and model-based controller for each domain can be found in Appendix A in the extended version.

4.1. NMOE Design and Baseline Models

To design an NMOE for each experimental domain, we estimate the number of contact modes of the robot and formulate a competitive MOE. The type of contact modes are illustrated in Figure 3. For each contact mode, we derive a white-box model based on first principles using estimations on the robot kinematics and inertial parameters. Note that these parameters are initialized within a 50 percent margin of error. We derive the equations of motion for the white-box models using the free body diagrams illustrated in Figure 3. We have developed a new rigid body dynamics library using Tensorflow to encode the white-box models (Ahn (2020)). In addition, we build the black-box models with a multi-layer perceptron denoted by MLP. The equations in this section use the nomenclature shown in Figure 3.

Reacher: Reacher has a state $[q; \dot{q}] \in \mathbb{R}^4$ and an action $\tau \in \mathbb{R}^2$ representing joint positions, velocities, and torques. The NMOE for the Reacher uses a single cooperative MOE since there is a single contact mode (a fixed base manipulator). It computes the output with the expression:

$$\begin{bmatrix} q_{k+1} \\ \dot{q}_{k+1} \end{bmatrix} = \sum_{j=1}^{2} g_j^{(1)} f_j^{(1)}(q_k, \dot{q}_k, \tau_k) = g_1^{(1)} \underbrace{\text{MLP}_{\theta_1}(q_k, \dot{q}_k, \tau_k)}_{\text{black-box}} + g_2^{(1)} \underbrace{\begin{bmatrix} q_k + \dot{q}_k dt \\ \dot{q}_k + \left(A^{-1}(\tau_k - b - g) \right) dt \end{bmatrix}}_{\text{white-box}},$$

(3)

where $A \in \mathbb{R}^{2 \times 2}$, $b \in \mathbb{R}^2$, and $g \in \mathbb{R}^2$ represent the mass matrix, centrifugal / coriolis, and gravity vectors, respectively. Trainable variables in the NMOE include the mass and the length of each link as well as the parameters in the MLP and the gating networks.

CartWithElasticWall: This system has a single degree of freedom cart sliding along the inclined pin. The cart has to interact with the elastic wall down below and accumulate energy to reach a desired height accounting for actuator limits. The cart has a state $[x; \dot{x}] \in \mathbb{R}^2$ and an input $u \in \mathbb{R}$ representing the cart's position, velocity, and force. The NMOE includes two cooperative MOEs since there are two possible contact modes: an interacting mode with the elastic wall and a free-floating mode when there is no wall contact. The output is computed as

$$\begin{bmatrix} x_{k+1} \\ \dot{x}_{k+1} \end{bmatrix} = \sum_{i=1}^{2} h_i \left(\sum_{j=1}^{2} g_j^{(i)} f_j^{(i)}(x_k, \dot{x}_k, u_k) \right)$$

$$= h_1 \overbrace{\left(g_1^{(1)} \underbrace{\text{MLP}_{\theta_1}(x_k, \dot{x}_k, u_k)}_{\text{black-box}} + g_2^{(1)} \underbrace{\begin{bmatrix} x_k + \dot{x}_k dt \\ \dot{x}_k + \frac{1}{m} \left(u_k + k(x_{\text{cart},0} - x_k) - b\dot{x}_k - mg\sin\theta \right) dt \end{bmatrix}}_{\text{white-box}} \right)}^{\text{1st cooperative MOE: Contact}}$$

$$+ h_2 \overbrace{\left(g_1^{(2)} \underbrace{\text{MLP}_{\theta_2}(x_k, \dot{x}_k, u_k)}_{\text{black-box}} + g_2^{(2)} \underbrace{\begin{bmatrix} x_k + \dot{x}_k dt \\ \dot{x}_k + \frac{1}{m} \left(u_k - mg\sin\theta \right) dt \end{bmatrix}}_{\text{white-box}} \right)}^{\text{2nd cooperative MOE: Free}},$$

(4)

where m and $x_{\text{cart},0}$ are the mass and initial position of the cart, k and b are the spring constant and the damping of the elastic wall, respectively, g is the gravitational acceleration, θ is the angle of the rail, and dt is the integration step. Training variables in the NMOE include m, k, b, θ, the parameters in the MLPs, and the gating networks.

Hopper: The Hopper has a state $[q; \dot{q}] \in \mathbb{R}^{12}$ and an input $\tau \in \mathbb{R}^3$ representing joint positions, velocities, and torques. The NMOE includes four cooperative MOEs since there are four possible contact modes: toe contact mode, heel contact mode, heel-toe contact mode, and contact free mode. The output is computed as

$$
\begin{bmatrix} q_{k+1} \\ \dot{q}_{k+1} \end{bmatrix} = \sum_{i=1}^{4} h_i \left(\sum_{j=1}^{2} g_j^{(i)} f_j^{(i)}(q_k, \dot{q}_k, \tau_k) \right)
$$

$$
= h_1 \left(g_1^{(1)} \underbrace{\text{MLP}_{\theta_1}(q_k, \dot{q}_k, \tau_k)}_{\text{black-box}} + g_2^{(1)} \underbrace{\begin{bmatrix} q_k + \dot{q}_k dt \\ \dot{q}_k + A^{-1}(N_{\text{T}}^{\top}(S^{\top}\tau_k - b - g) - J_{\text{T}}^{\top}\Lambda_{\text{T}}\dot{J}_{\text{T}}\dot{q})dt \end{bmatrix}}_{\text{white-box}} \right)
$$

$$
\overbrace{\phantom{= h_1 \left(g_1^{(1)} \text{MLP}_{\theta_1}(q_k, \dot{q}_k, \tau_k) + g_2^{(1)} \right)}}^{\text{1st cooperative MOE: T(oe) Contact}}
$$

$$
\overbrace{+ h_2 \left(g_1^{(2)} \underbrace{\text{MLP}_{\theta_2}(q_k, \dot{q}_k, \tau_k)}_{\text{black-box}} + g_2^{(2)} \underbrace{\begin{bmatrix} q_k + \dot{q}_k dt \\ \dot{q}_k + A^{-1}(N_{\text{H}}^{\top}(S^{\top}\tau_k - b - g) - J_{\text{H}}^{\top}\Lambda_{\text{H}}\dot{J}_{\text{H}}\dot{q})dt \end{bmatrix}}_{\text{white-box}} \right)}^{\text{2nd cooperative MOE: H(eel) Contact}}
$$

$$
\overbrace{+ h_3 \left(g_1^{(3)} \underbrace{\text{MLP}_{\theta_3}(q_k, \dot{q}_k, \tau_k)}_{\text{black-box}} + g_2^{(3)} \underbrace{\begin{bmatrix} q_k + \dot{q}_k dt \\ \dot{q}_k + A^{-1}(N_{\text{HT}}^{\top}(S^{\top}\tau_k - b - g) - J_{\text{HT}}^{\top}\Lambda_{\text{HT}}\dot{J}_{\text{HT}}\dot{q})dt \end{bmatrix}}_{\text{white-box}} \right)}^{\text{3rd cooperative MOE: H(eel), T(oe) Contact}}
$$

$$
\overbrace{+ h_4 \left(g_1^{(4)} \underbrace{\text{MLP}_{\theta_4}(q_k, \dot{q}_k, \tau_k)}_{\text{black-box}} + g_2^{(4)} \underbrace{\begin{bmatrix} q_k + \dot{q}_k dt \\ \dot{q}_k + A^{-1}(S^{\top}\tau_k - b - g)dt \end{bmatrix}}_{\text{white-box}} \right)}^{\text{4th cooperative MOE: Contact free}}, \tag{5}
$$

where $A \in \mathbb{R}^{6\times6}$, $b \in \mathbb{R}^6$, and $g \in \mathbb{R}^6$ represent the mass matrix, and centrifugal / coriolis, and gravity vectors, respectively. $S \in \mathbb{R}^{3\times6}$ is the selection matrix that maps joint space to actuator space. J_{T} and J_{H} are both in $\mathbb{R}^{3\times6}$, denoting the jacobian matrices associated with the toe and heel contact frames with respect to the world. $J_{\text{HT}} \in \mathbb{R}^{6\times6}$ is a matrix vertically concatenating J_{H} and J_{T}. $\Lambda_* = (J_* A^{\top} J_*^{\top})^{\top}$ is a mass matrix in frame $*$, $\overline{J}_* = A^{-1} J_*^{\top} \Lambda_*$ is a dynamically consistent pseudo-inversion of J_*, and $N_* = I - \overline{J}_* J_*$ is a null-space projection operator in frame $*$, where $* \in \{\text{T}, \text{H}, \text{HT}\}$. Trainable variables in the NMOE include the mass of each link as well as the parameters in the MLP and the gating networks. Readers can refer to Appendix B in the extended version for more details about white-box modeling.

HalfCheetah: The HalfCheetah has a state $[q; \dot{q}] \in \mathbb{R}^{18}$ and an input $\tau \in \mathbb{R}^6$ representing joint positions, velocities, and torques. The NMOE includes four cooperative MOEs since there are four possible contacts with the ground: front-foot contact mode, rear-foot contact mode, front- and rear-foot contact mode, and contact free mode. The structure and equations of the NMOE are identical to Eq. (5), with a few modifications (i.e., the dimensions of the kinematic and dynamic operators and the contact frames).
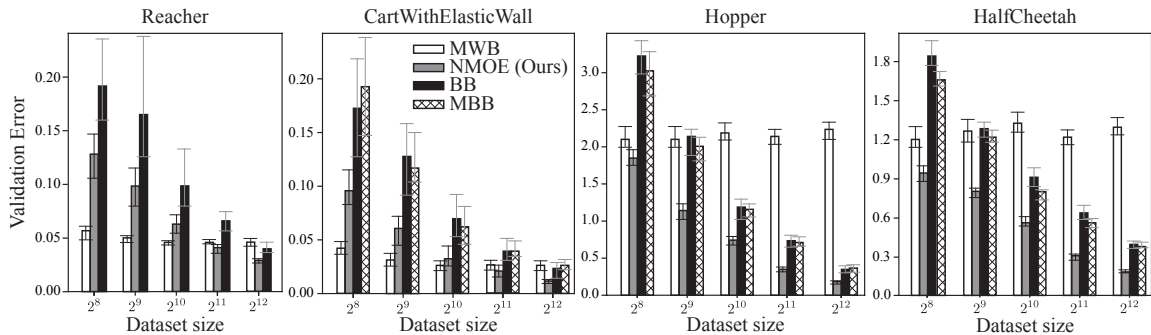
Figure 4: Validation error obtained using our NMOE setup and compared to baseline models for various training data sizes. We plot the median values with interquartile ranges of five runs.

We compare our NMOE models to the following baseline approaches:

- **Single black-box model (BB)**, which involves a single large size neural network implemented using an `MLP`.

- **Mixture of black-box models (MBB)**, being equivalent to our NMOE if the cooperative MOEs include only black-box models. In other words, it is identical to our NMOE when the white-box models are ignored (i.e., $g^{(i)} = [1, \ 0]$, $f_2^{(i)} = 0$). Note that in the case of the Reacher, the BB model and the MBB are equivalent since it only has one mode (i.e., $M = 1$).

- **Mixture of white-box models (MWB)**, which is equivalent to our NMOE when the cooperative MOEs has only white-box models. In other words, it is identical to our NMOE when the black-box models are ignored (i.e., $g^{(i)} = [0, \ 1]$, $f_1^{(i)} = 0$).

4.2. Performance Analysis

We investigate performance of our NMOE models in terms of data-efficiency, generalization to unseen data, and bias-variance trade-off. For all experimental domains, we uniformly sample state, input, and next-state tuples to build training and test datasets. We then train the NMOE and the baseline models and measure a root mean squared error using the test set. We train the models with various size of training sets ranging from 2^8 to 2^{12} sizes and evaluate them using a test set with 2^{12} tuples. Figure 4 shows the validation error using our NMOE compared to the baseline models demonstrating our NMOE's ability to generalize from limited data. For the cases involving the Hopper and HalfCheetah systems, the MWB model suffers from model bias due to hard-to-model effects and low model capacity, resulting in large generalization errors. The BB and the MBB models show worse data-efficiency and higher model variance in the learning procedure than the other models. We therefore conclude that our NMOE outperforms the baseline models for learning various complex dynamical systems in terms of data-efficiency, generalization, and bias-variance trade-off.

4.3. Gating Network Evaluation

We study how the gating network in the top layer, h, in our NMOE assigns input data to local experts associated with the contact modes. Let's consider the NMOE for the CartWithElasticWall domain. After the training converges, we collect trajectories from three randomly sampled initial states with zero inputs and evaluate them. Figure 5a shows the outputs of the gating network displayed in the phase plot. Note that h_1 and h_2 represent the responsibility to the first and second contact modes which correspond the contact and contact-free modes, respectively, as in Eq. (4). The phase plot demonstrates that the NMOE assigns inputs which have negative displacement and negative velocity, where the cart is most likely contacting the elastic wall, to the first cooperative MOE. Similarly, the NMOE assigns inputs to the second cooperative MOE when the cart is in contact-free mode.

We further demonstrate how the NMOE optimizes the gating network in the bottom layer, g, depending on the accuracy of the prior knowledge. In the Reacher domain, we prepare multiple training datasets uniformly sampled from different simulation setups. For example, we set different joint friction and damping coefficients, μ and b, for each dataset. Note that the accuracy of the white-box model in the NMOE decreases as the simulation is setup with large friction and damping since the white-box model is derived under the assumption of zero-friction and zero-damping. Figure 5b summarizes the average of the responsibility evaluated over the state-space (i.e., $\mathbb{E}\left[g^{(1)}(q, \dot{q}, \tau)\right]$) for each training case after convergence. Note that the first and the second element of vector g corresponds to the black-box and the white-box model, respectively, as in Eq. (3). It shows that the NMOE relies more on the white-box model to generate the output when the prior knowledge is more reliable.

(a) CartWithElasticWall

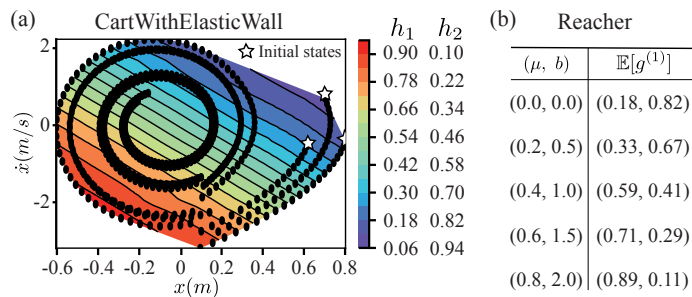| $(\mu,\ b)$	$\mathbb{E}[g^{(1)}]$
(0.0, 0.0)	(0.18, 0.82)
(0.2, 0.5)	(0.33, 0.67)
(0.4, 1.0)	(0.59, 0.41)
(0.6, 1.5)	(0.71, 0.29)
(0.8, 2.0)	(0.89, 0.11)

(b) Reacher

Figure 5: Gating network output after convergence.

4.4. Reliability of Model-based Control

We evaluate our NMOE using an MBRL setup where the network is linked to a model-based controller and is trained online. Here, we devise a model predictive path integral controller (Williams et al. (2017)) using our NMOE model and compared with other baseline models. The reward function for each experimental domain is described in Appendix C in the extended version. Figure 6 summarizes the performance of the model-based controllers. It includes the cumulative rewards in the training procedure and the final performance of the controllers after convergence. The final performance is measured with the mean and std of rewards collected in one episode after the models converge. Note that our NMOE achieves a similar cumulative reward gradient to the ground truth model at the earliest timestep compared to the baseline methods. The MWB model provides a reliable model capacity for the Reacher and the CartWithElasticWall but fails to represent the Hopper and HalfCheetah. The BB model and the MBB model require much more data to converge and show higher variance in the final performance compared to our NMOE.

9

(b) **Mean and Std of Rewards Collected in One Episode (After Convergence)**

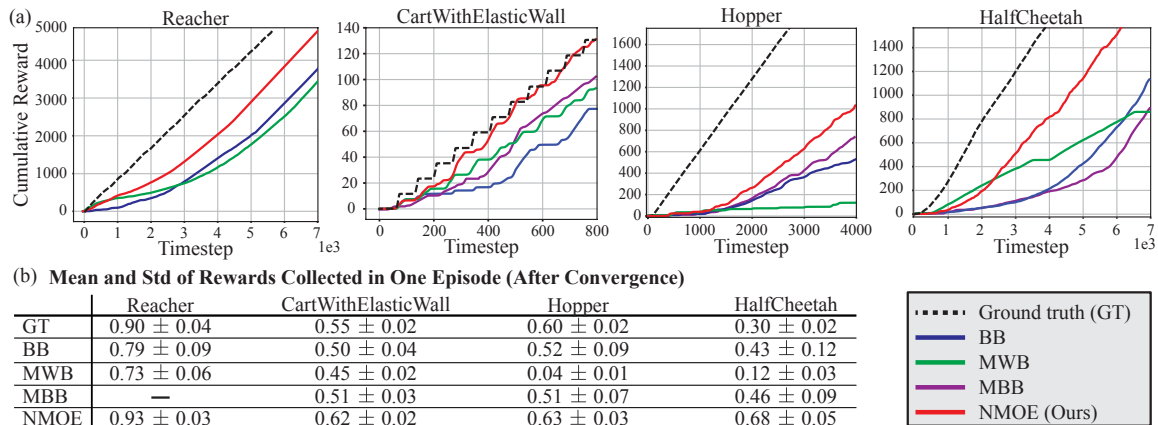|	Reacher	CartWithElasticWall	Hopper	HalfCheetah
GT	0.90 ± 0.04	0.55 ± 0.02	0.60 ± 0.02	0.30 ± 0.02
BB	0.79 ± 0.09	0.50 ± 0.04	0.52 ± 0.09	0.43 ± 0.12
MWB	0.73 ± 0.06	0.45 ± 0.02	0.04 ± 0.01	0.12 ± 0.03
MBB	—	0.51 ± 0.03	0.51 ± 0.07	0.46 ± 0.09
NMOE	0.93 ± 0.03	0.62 ± 0.02	0.63 ± 0.03	0.68 ± 0.05

Figure 6: (a) illustrates the average cumulative rewards during the training. (b) represents the final performance of the model-based controllers. We run five experiments with random seeds.

5. Conclusion

In this study, we introduce a novel network structure, called a nested mixture of experts, to represent and learn complex robotic systems and behaviors including hybrid interactions such as contact changes. Our network structure provides a tool for incorporating prior knowledge about the robot's interactions with the environment as well as kinematic and dynamic properties. We devise a loss function to train the local experts to be competitive or cooperative, which enables our NMOE to be trained quickly, to generalize to unseen data from limited data, and to optimize bias-variance trade-off. In various continuous control domains, we evaluate our NMOE with related baseline models both in a supervised learning and an MBRL setup.

Acknowledgments

References

J. Ahn, D. Kim, S. Bang, N. Paine, and L. Sentis. Control of a high performance bipedal robot using viscoelastic liquid cooled actuators. In *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, pages 146–153, 2019.

J. Ahn, J. Lee, and L. Sentis. Data-efficient and safe learning for humanoid locomotion aided by a dynamic balancing model. *IEEE Robotics and Automation Letters*, 5(3):4376–4383, 2020.

Junhyeok Ahn. tf_rbdl: Rigid Body Dynamics Library using Tensorflow, May 2020. URL https://doi.org/10.5281/zenodo.3780707.

C. H. An, C. G. Atkeson, and J. M. Hollerbach. Estimation of inertial parameters of rigid body links of manipulators. In *1985 24th IEEE Conference on Decision and Control*, pages 990–995, 1985.

H. Bai, D. Hsu, and W. S. Lee. Planning how to learn. In *2013 IEEE International Conference on Robotics and Automation*, pages 2853–2859, 2013. doi: 10.1109/ICRA.2013.6630972.

A. Carron, E. Arcari, M. Wermelinger, L. Hewing, M. Hutter, and M. N. Zeilinger. Data-driven model predictive control for trajectory tracking with a robotic arm. *IEEE Robotics and Automation Letters*, 4(4):3758–3765, 2019.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pages 4754–4765, 2018.

Scott Fujimoto, Herke Van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems 32*, pages 15379–15389. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/9672-hamiltonian-neural-networks.pdf.

Jayesh K. Gupta, Kunal Menda, Zachary Manchester, and Mykel Kochenderfer. Structured mechanical models for robot learning and control. volume 120 of *Proceedings of Machine Learning Research*, pages 328–337, The Cloud, 10–11 Jun 2020. PMLR. URL http://proceedings.mlr.press/v120/gupta20a.html.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Torsten Koller, Felix Berkenkamp, Matteo Turchetta, Joschka Boedecker, and Andreas Krause. Learning-based model predictive control for safe exploration and reinforcement learning, 2019. URL https://arxiv.org/abs/1906.12189.

Sergey Levine and Pieter Abbeel. Learning neural network policies with guided policy search under unknown dynamics. In *Advances in Neural Information Processing Systems*, pages 1071–1079, 2014.

M. Lutter, C. Ritter, and J. Peters. Deep lagrangian networks: Using physics as model prior for deep learning. In *7th International Conference on Learning Representations (ICLR)*, May 2019. URL https://openreview.net/pdf?id=BklHpjCqKm.

Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7559–7566. IEEE, 2018.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Multistep neural networks for data-driven discovery of nonlinear dynamical systems. *arXiv preprint arXiv:1801.01236*, 2018.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Giovanni Sutanto, Austin Wang, Yixin Lin, Mustafa Mukadam, Gaurav Sukhatme, Akshara Rai, and Franziska Meier. Encoding physical constraints in differentiable newton-euler algorithm. volume 120 of *Proceedings of Machine Learning Research*, pages 804–813, The Cloud, 10–11 Jun 2020. PMLR. URL http://proceedings.mlr.press/v120/sutanto20a.html.

G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou. Information theoretic mpc for model-based reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1714–1721, 2017.

Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Symplectic ode-net: Learning hamiltonian dynamics with control. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ryxmb1rKDS.