A Additional Discussions on the Experiments

In this section, we first delineate a more in-depth discussion and detailed description of the training process for the Pendulum and Cartpole NNDMs. We then showcase our experiments on two of the OpenAI agents by providing simulation results for both the certification and the control of the NNDMs. At last, a detailed description on the training process for the Husky and Acrobot is provided. Table 2 shows the partition width in each dimension for all the NNDMs.

A.1 Pendulum and Cartpole

We trained NNDMs with fixed number of epochs (300) and implemented an early stopping method to avoid over-fitting. The number of data points were increased from 5,000 to 50,000 as the depth of the NNDM increased for a fixed model, ranging from 1 layer to 5 layers. Figure 2 shows the vector fields of the pendulum NNDMs with various architectures.

Pendulum In this case-study, we trained a NNDM for a Pendulum agent which has fixed mass m and length l with actuator limits $u \in [-1, 1]$. We trained the NN under a given controller from [41] that tries to keep the pendulum upright. We modified the original OpenAI gym environment and directly learned the evolution of state variables θ and $\dot{\theta}$. We trained $f^w : \mathbb{R}^2 \to \mathbb{R}^2$ with one to five hidden layers with 64 neurons each. The NNDM is trained in region $\theta \in [-\pi, \pi]$ and $\dot{\theta} \in [-1, 1]$. The safe set is $\theta \in [-\frac{\pi}{15}, \frac{\pi}{15}]$ and $\dot{\theta} \in [-1, 1]$ and the initial set is $\theta \in [-\frac{\pi}{36}, \frac{\pi}{36}]$. The state space X is as mentioned in Table 3. We compute linear bounds for various discretization sizes as discussed in Section 4.1 and Table 2.

The dynamics of the system is

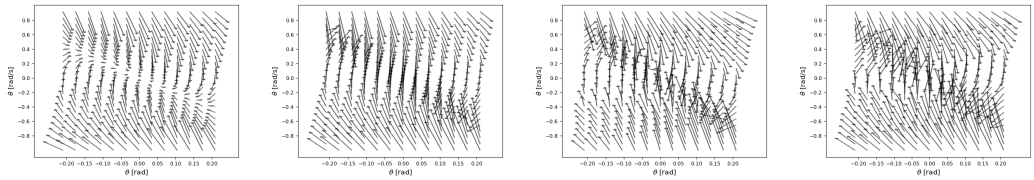
$$\dot{\theta}_{k+1} = \dot{\theta}_k + \frac{3g}{2l}\sin(\theta_k)\delta t^2 + \frac{3}{ml^2}u\delta t^2, \tag{11}$$

$$\theta_{k+1} = \theta_k + \dot{\theta}_{k+1}\delta t \tag{12}$$

For the Pendulum 1-layer NNDM models, we notice that the probability of safety is above δ_s even for coarser discretizations. In general, we observe that for a fixed model, the probability of safety

Table 2: Discretization parameters for each model of dimension n for various partitioning $|Q|$.

Model	n	$\lvert Q\rvert$	Discretization					
			θ				$\dot{\theta}$	
Pendulum	2	120	0.01745329				0.1	
		240	0.00872664				0.1	
		480	0.00872664				0.05	
		960	0.00436332				0.05	
		1920	0.00436332				0.025	
			x	$\dot{x}$	θ		$\dot{\theta}$	
Cartpole	4	960	0.2	0.125	0.01745329		0.125	
		1920	0.2	0.125	0.01745329		0.0625	
		3840	0.1	0.125	0.000872665		0.125	
			x	y	θ		v	
Husky	4	900	0.2	0.2	0.01745329		0.2	
		1800	0.2	0.2	0.00872665		0.2	
		2250	0.2	0.2	0.01745329		0.1	
		4800	0.125	0.125	0.01745329		0.125	
			x	$\dot{x}$	θ	$\dot{\theta}$	ω	
Huksy	5	432	0.2	0.2	0.01745329	0.2	0.2	
		1080	0.2	0.1	0.01745329	0.2	0.2	
		1728	0.2	0.2	0.01745329	0.125	0.125	
			$\cos(\theta_1)$	$\sin(\theta_1)$	$\cos(\theta_2)$	$\sin(\theta_2)$	$\dot{\theta}_1$	$\dot{\theta}_2$
Acrobot	6	144	0.05	0.1	0.05	0.1	0.25	0.25
		288	0.05	0.05	0.05	0.1	0.25	0.25

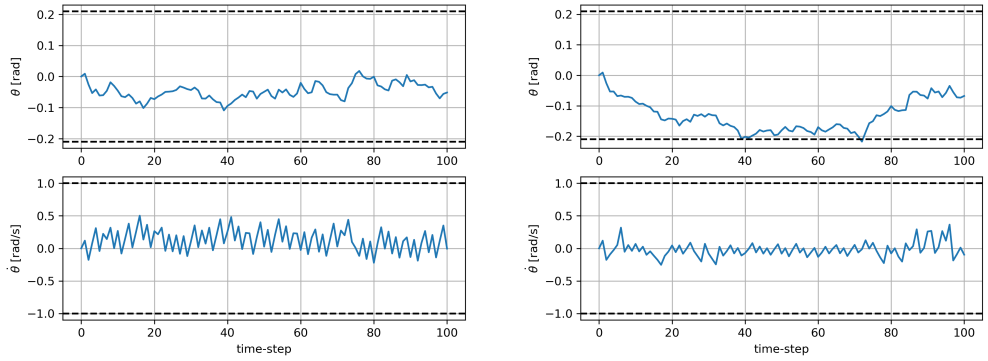


(a) 1 layer (b) 2 layer (c) 3 layer (d) 5 layer

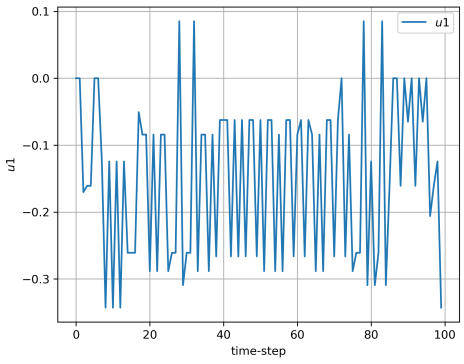
Figure 2: Vector fields of the NNDMs for the Pendulum agent with various architectures.

increases as the discretization becomes finer. For a fixed discretization, the general trend is a decrease in probability of safety as the model gets deeper (more hidden layers).

Figure 3 shows the evolution of the state variables with and without the controller π and the control evolution u under π for the 3-layer NNDM model with noise for one simulation. We observe the NNDM stays within the safe in θ dimension without the controller while the system violates the safety constraint under π. We also note that the controller magnitude varies from 0.1 to -0.4 which is within its bounds of $[-1, 1]$.

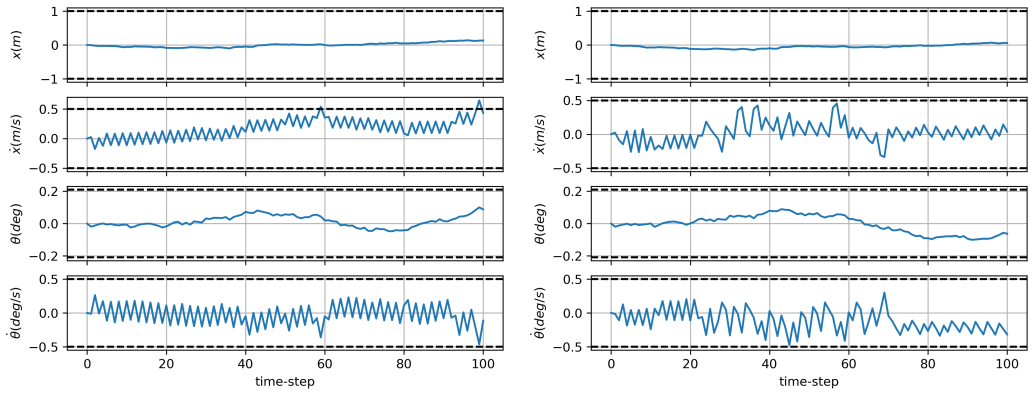


(a) Pendulum without π (b) Pendulum with π



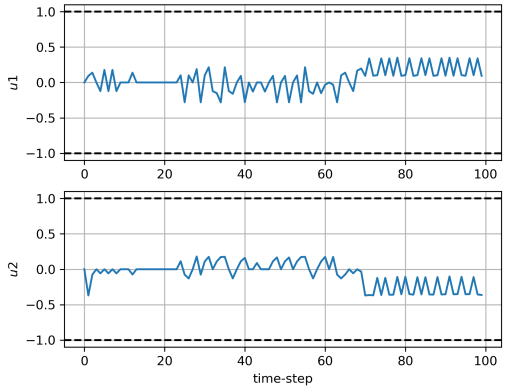
(c) Control evolution under π

Figure 3: State evolution of the Pendulum 3 layer NNDM model for 100 steps with and without safety feedback controller π. Blue line shows state evolution for the system starting at $\theta = 0$ and $\dot{\theta} = 0$. Black dotted line demarcates the safe space (X_{s}) in θ dimension and the state space (X) in $\dot{\theta}$ dimension.



(a) Cartpole without safety controller π

(b) Cartpole with safety controller π



(c) Control evolution under π

Figure 4: State evolution of the cartpole NNDM with 2 layers for 100 steps with and without safety feedback controller π in (a) and (b), respectively. Blue line shows state evolution for the system starting at $x = 0$, $\dot{x} = 0$, $\theta = 0$, and $\dot{\theta} = 0$. Black dotted line demarcates the state space (X) in $\dot{x}$, $\dot{\theta}$, and safe space (X_{s}) in x and θ dimension respectively.

Cartpole For this 4-dimensional agent, we trained a NNDM that predicts the next state of the cartpole given the current state, i.e., $f^w : \mathbb{R}^4 \to \mathbb{R}^4$. For each model, we trained a NNDM, $f^w : \mathbb{R}^4 \to \mathbb{R}^4$, with one to three layers with 128 neurons in each layer. The NNDM is trained in region $x \in [-2.4, 2.4]$, $\dot{x} \in [-0.6, 0.6]$, $\theta \in [-\frac{\pi}{15}, \frac{\pi}{15}]$ and $\dot{\theta} \in [-0.6, 0.6]$. The safe set is $x \in [-1, 1]$ and $\theta \in [-\frac{\pi}{15}, \frac{\pi}{15}]$ with control limits $u_1, u_2 \in [-1, 1]$ in linear and angular acceleration dimensions. The initial set is defined to be $\theta \in [-\frac{\pi}{36}, \frac{\pi}{36}]$.

While the computation time to obtain the linear approximations increases significantly as the number of layers and number of neurons per layer increases, we notice the same trends as observed in the Pendulum agent's experiment, i.e., an increase in P_{s} with finer discretization for a fixed model and a decrease in P_{s} as the model gets deeper (more hidden layers). For all the cartpole models, to achieve the desired probability of safety δ_{s}, the controller needs to be applied in all the regions. Figure 4 shows the evolution of the 2-layer NNDM cartpole with and without the safety controller. We can see that our controller is able to keep the system within the safe set as seen in Figure 4b while it also stays within its allowable range of controller magnitude $[-1, 1]$ as seen in Figure 4c.

A.2 Husky

We trained two Husky NNDMs for a specific task using the approach outlined in [9]. Specifically, a hybrid Model-Based Model-Free (MB-MF) architecture has been used for training NNs that represent the dynamic model of the Husky robot [47]. Further, this NN with a model-based controller, i.e. Model Predictive Controller (MPC), was used to learn tasks that include moving from one point to

Table 3: State Space X for each agent of dimension n in our case-studies over which we compute linear over-and under approximation $\overline{f}_q(x)$ and $\underline{f}_q(x)$ and their corresponding extreme values in region q using the discretization $|Q|$ as mentioned in Table 2.

Model	n	State	Lower	Upper
Pendulum	2	θ	$-\pi/15$	$\pi/15$
		$\dot{\theta}$	-1	1
Cartpole	4	x	-1	1
		$\dot{x}$	-0.5	0.5
		θ	$-\pi/15$	$\pi/15$
		$\dot{\theta}$	-0.5	0.5
Husky	4	x	-0.5	2
		y	-1	1
		θ	$-\pi/12$	$\pi/12$
		v	-0.5	0.5
Husky	5	x	-0.5	2
		y	-0.5	0.5
		θ	$-\pi/18$	$\pi/18$
		v	-0.5	0.5
		ω	-0.5	0.5
Acrobot	6	$\cos(\theta_1)$	-0.1	0.1
		$\sin(\theta_1)$	-0.6	0.6
		$\cos(\theta_2)$	-0.1	0.1
		$\sin(\theta_2)$	-0.6	0.6
		$\dot{\theta}_1$	-0.25	0.25
		$\dot{\theta}_2$	-0.25	0.25

another, and staying in a lane. We use the data generated from this model to initialize a model-free RL algorithm to gain better sample efficiency. This enabled us to learn the unknown dynamic model and a policy for the task with less number of samples compared to a pure model-free approach.

In our case study, we first consider a 4-dimensional Husky model [47], whose states are the x and y position, the orientation θ and the velocity v. The NNDM is trained in region $x \in [-0.5, 2]$, $y \in [-1, 1]$, $\theta \in [-\frac{\pi}{12}, \frac{\pi}{12}]$ and $v \in [-0.5, 0.5]$. The control actions for this model are linear acceleration and angular velocity with control limits $u_1, u_2 \in [-1, 1]$. The initial set is any position with a circle of radius 0.1 around the origin, i.e., $x \in [-0.1, 0.1]$ and $y \in [-0.1, 0.1]$.

Further, we also train a 5-dimensional Husky model in which we observe the angular velocity ω along with linear velocity, orientation, and position as in the 4-dimensional case-study. The state space is $x \in [-0.5, 2]$, $y \in [-0.5, 0.5]$, $\theta \in [-\frac{\pi}{18}, \frac{\pi}{18}]$, $v \in [-0.5, 0.5]$, and $\omega \in [-0.5, 0.5]$. The safe set is defined over y to be $[-0.5, 0.5]$ while the control actions, control limits and the initial set remains the same.

The dynamic model has an input layer with each neuron (six for 4D Husky and seven for 5D Husky in total) representing the individual states and controls of the system. Further, the dynamic model consists of one hidden layer with 500 neurons (ReLU activation function), and the output of the NN is the state difference (4 output neurons for 4D Husky and 5 for 5D Husky) for a defined discretization time ΔT.

Training The data used for training the dynamic model was generated from the described Husky environment modelled in the PyBullet physics simulator [42]. The dynamic model is trained using trajectory data generated from PyBullet, by applying random actions to a set of initial states sampled from a distribution.

Table 4: Parameters for dynamic model and policy network training for the husky systems.

Parameter	**4D Husky**	**5D Husky**
Open-loop Dynamic Model Architecture	6×500(ReLU) $\times 4$	7×500(ReLU) $\times 5$
Policy Network Architecture	4×64(tanh) $\times 64$(tanh) $\times 2$	5×64(tanh) $\times 64$(tanh) $\times 2$
Dynamic Model Training Epoch	200	200
Discretization Time ΔT (secs)	0.1	0.1
MPC Controller Horizon (Timesteps)	1	1
Number of MPC rollouts	100	100
Closed-loop NNDM Architecture 1	4×256(ReLU) $\times 4$	5×512(ReLU) $\times 5$
Closed-loop NNDM Architecture 2	4×256(ReLU) $\times 256$(ReLU) $\times 4$	-

The data is normalized to give equal weights to all the states. We used a MPC controller to generate policies required to complete the specific task. We then generated a set of expert actions from

this MPC controller to train a new policy network π_c. The policy network has one input layer (4 input neurons) representing the states, two hidden layers with 64 neurons each (all tanh activation functions), and one output layer with 2 neurons representing the actions. This policy network π_c was used as an initial policy for the model-free RL. We used the Trust Region Policy Optimization algorithm (TRPO) as a policy gradient method to perform RL. The parameters that we used for our training are described in Table 4.

The trained dynamic model and the policy network π_c were combined to represent the closed-from dynamics of the system. We performed imitation learning on the input-output data obtained from this combined model, to train a NNDM $f^w : \mathbb{R}^n \to \mathbb{R}^n$, where n is the states of the system.

A.3 Acrobot

We train a NNDM to imitate the OpenAI Gym Acrobot model under a given expert controller. The setting we consider is the same as in [48]. It is an underactuated agent (double pendulum) with control applied to the second joint. The NNDM is trained in region from $[-1, 1]$ in the first 4 dimensions, i.e., $\cos(\theta_1)$, $\sin(\theta_1)$, $\cos(\theta_2)$ and $\sin(\theta_2)$. The model was trained over 25000 data points with 300 epochs and validation loss of 10^{-3}. The state space over which the system is verified is mentioned in Table 3.

Here, θ_1 is the angle of the first joint and θ_2 is the angle relative to the angle of the first link. The task for this agent is for the tip of the second link to reach a height of $y = \sin(\theta_1) + \sin(\theta_1 + \theta_2) = 1$. A safety constraint is added such that the tip of the second link should not go beyond $y = 1.2$, i.e., $\sin(\theta_1) \leq 0.6$ and $\sin(\theta_2) \leq 0.6$. Hence, the safe set is defines as $\sin(\theta_1) \in [-0.6, 0.6]$ and $\sin(\theta_2) \in [-0.6, 0.6]$. Note, the maximum height the pendulum can reach is 1.8 m. Finally, the initial set is defined to be any initial point within a radius of 0.1 around the origin in the first 4 dimensions.