2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Learned Regions of Attraction for Safe Motion Primitive Transitions

Wyatt Ubellacker and Aaron D. Ames

***Abstract*— Estimating regions of attraction (ROAs) of dynamical systems is critical for understanding the operational bounds within which a system will converge to a desired state. In this paper, we introduce a neural network-based approach to approximating ROAs that leverages labeled data generated by offline sampling and simulation of initial conditions, with labels determined by flow membership in an "explicit region of attraction." This framework is designed to estimate ROAs with a level of precision suitable for integration into a motion primitive transition framework as conditions to switch between candidate primitive behaviors. To account for gaps between the simulated environment and the real world, online learning is employed; this refines the offline-learned model of the ROA based on observed discrepancies between predicted and actual system behaviors. We validate this methodology on a quadrupedal robot, demonstrating that our ROA estimates can effectively model regions of attraction for a high-dimensional system. We show this for multiple primitive behaviors and in environments different from the training data. The outcomes highlight the usefulness of our method in estimating regions of attraction and informing transition conditions between primitive behaviors.**

I. Introduction

The concept of a region of attraction (ROA) plays an important role in the analysis of dynamical systems—it can provide essential insights in predicting system behavior. In systems with multiple desired behaviors and corresponding controllers, understanding and approximating these regions is fundamental for determining when a behavior is suitable for use, and reasoning on the regions of attraction can enable transitions between these behaviors, as is done in a motion primitive transition framework [1]. This information is useful in scenarios where the robot must transition between different modes of operation to reach a desired behavior or to react to disturbances and uncertainties that are too challenging for a single behavior to manage effectively.

Studying regions of attraction has important ramifications that extend beyond robotics. It has its origins going back to the seminal work of Lyapunov and Lasalle [2], [3]. This has led to a concerted attempt to calculate (approximations) of the ROA. For practical uses, numerical approximations of ROA have proven to be effective in achieving reasonable characterizations. One of the earliest numerical techniques is Zubov's Method [4], [5], which describes the region of attraction with a Lyapunov function derived from the solution of a first-order partial differential equation. Other numerical methods iteratively estimate the region of attraction via sublevel sets of local Lyapunov functions [6], [7], [8], [9]. Hamilton-Jacobi-based methods have also shown promise by leveraging the relationship between the region of attraction and the backwards reachable set [10], [11]. Work by [12], [13] directly computes the boundary of the regions of attraction, and sum of squares optimization techniques are effective in identifying Lyapunov functions that describe the regions of attraction for systems governed by polynomial and rational dynamics [14], [15], [16].

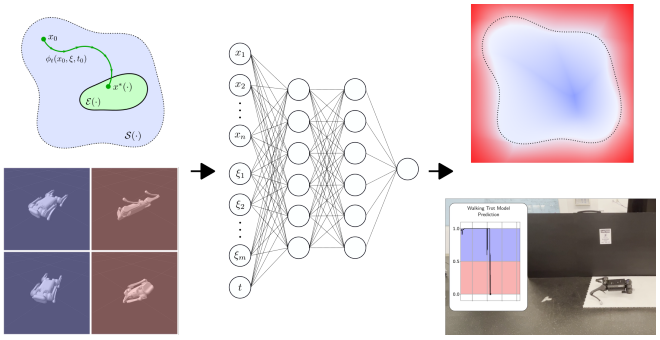


Fig. 1: Training data is sampled and labeled via membership of the flow in an *explicit region of attraction*. A binary classifier is trained to predict the region of attraction.

With the rise of machine learning as a data-driven solution to traditionally intractable problems, learning regions of attraction has seen several recent advancements. Some techniques include utilizing Gaussian processes to learn regions of attraction for uncertain and nonlinear systems [17]. In [18], neural networks are used to produce approximations for a particular class of non-linear systems, and [19] introduced a model-free approach based on the recurrent property of system trajectories.

This paper presents a neural network-based method for estimating regions of attraction in nonlinear dynamical systems under mild assumptions. We generate training data by sampling initial conditions from the state and parameter space of our system. The system is simulated from this point, and samples are labeled based on whether the system flow enters an "explicit region of attraction" or not. This is then posed as a straightforward binary classification problem. As this is intended for use on a robotic system where the real world may not match the simulation environment, we include an online learning component to correct for mispredictions of the resulting model. Our method is showcased on a Unitree A1 quadruped for a variety of behaviors as part of a motion primitive framework [1].

This research is supported by Dow (#227027AT).

Authors are with the Departments of Control and Dynamical Systems, Mechanical and Civil Engineering, California Institute of Technology, Pasadena, CA, USA. wubellac, ames@caltech.edu

II. Preliminaries

We will reason in the context of nonlinear dynamical systems of the form

$$\dot{x} = f(x, u) \tag{1}$$

with state $x \in \mathcal{X}$, input $u \in \mathcal{U}$, and function $f : \mathcal{X} \times \mathcal{U} \mapsto T\mathcal{X}$, assumed to be locally Lipschitz continuous. Control input is assumed to be defined by a feedback control law such that $u = k(x, \xi, t)$, where ξ represents continuous arguments from a bounded argument set, $\Xi \subset \mathbb{R}^a$, and $t \in \mathbb{R}$ denotes time.

Assuming forward completeness, the *flow*, $\phi_T(x_0, \xi, t_0)$, of this system is the solution to (1) with initial condition x_0, arguments and phasing time to the control law, ξ and t_0.

In this work, we are interested in systems with an *asymptotically stable equilibrium*, a point x^* such that $f(x^*) = 0$ and $\forall \epsilon > 0,\ \exists \delta > 0$ such that:

$$||x_0 - x^*|| < \delta \ \Rightarrow \ \begin{cases} \|\phi_T(x_0, \xi, t_0) - x^*\| < \epsilon & \forall T \geq 0 \\ \lim\limits_{T \to \infty} ||\phi_T(x_0, \xi, t_0) - x^*|| = 0 \end{cases}$$

A. Regions of Attraction

For general nonlinear systems, including systems of the form in (1), we can only expect stability properties to hold locally, and it is this region of attraction that we are interested in determining.

Definition 1. The *region of attraction (ROA)* of an asymptotically stable equilibrium point $x^* \in \mathcal{X}$ is given by:

$$\mathcal{S} \triangleq \{x \in \mathcal{X} : \lim_{T \to \infty} ||\phi_T(x, \xi, t_0) - x^*|| = 0\}.$$

A useful construct for this work will be an *explicit region of attraction*, $\mathcal{E} \subset \mathcal{S}$, of an asymptotically stable equilibrium point x^* that satisfies $x^* \in \text{Int}(\mathcal{E})$.

$\mathcal{E}$ can be produced from methods described in Section I, but notably, as long as the assumptions on $\mathcal{E}$ are met, the accuracy of $\mathcal{E}$ compared to the true value of $\mathcal{S}$ is unimportant. This allows us to still utilize methods that may be overly conservative or intractable for our systems of interest.

We will use the characterization of explicit regions of attraction to determine points that lie within the true region of attraction $\mathcal{S}$. Consider system (1) with associated flow $\phi_T(x_0, \xi, t_0)$ as in [20]:

Theorem 1. *For any point $x_0 \in \mathcal{X}$, control law arguments ξ, initial phasing t_0, and fixed horizon $T \in \mathbb{R}_{\geq 0}$, we have that $\phi_T(x_0, \xi, t_0) \in \mathcal{E} \implies x_0 \in \mathcal{S}$.*

Proof. By virtue of $\mathcal{E} \subset \mathcal{S}$, we have that $\phi_T(x_0, \xi, t_0) \in \mathcal{E}$ implies $\phi_T(x_0, \xi, t_0) \in \mathcal{S}$. From the definition of $\mathcal{S}$, we have that $\phi_T(x_0, \xi, t_0) \in \mathcal{S}$ implies $x_0 \in \mathcal{S}$. □

B. Neural Networks

Neural networks are pivotal in machine learning for their ability to model complex functions that would otherwise be difficult or impossible to determine by other means. Modeling regions of attraction, especially in higher dimensions, is especially difficult, and neural networks present an effective mechanism to characterize these regions.

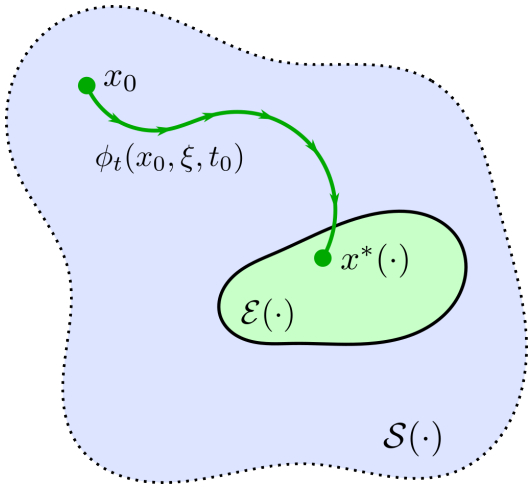


Fig. 2: $\mathcal{E}$ is an overly conservative, but known subset of the region of attraction, $\mathcal{S}$. Membership of $x_0 \in \mathcal{S}$ can be determined by membership of $\phi_T(x_0, \xi, t_0) \in \mathcal{E}$ for some $T \geq 0$ [20].

Definition 2. A *neural network*, $N : \mathbb{R}^i \mapsto \mathbb{R}^o$, is a series of layers defined by:

$$a_l = \sigma(W_l a_{l-1} + b_l)$$

where $l = 1, \ldots, L$ indexes the layer, L is the number of layers, $a_l \in \mathbb{R}^{n_l}$ is the output of the l-th layer. $a_0 \in \mathbb{R}^i$ and $a_L \in \mathbb{R}^o$ are the input and output vectors of the network, respectively. The matrix W_l is referred to as the *weight*, b_l is the *bias*, and σ_l is the *activation function*.

For the networks used in this work, each layer will use the sigmoid activation function.

Definition 3. The *sigmoid function*, $\sigma : \mathbb{R} \mapsto (0, 1)$, is defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

It is continuous, differentiable, and non-constant, and is a popular activation function in neural networks.

The power of neural networks is highlighted by the following core theorem:

Theorem 2 (Universal Approximation Theorem [21])**.** *A neural network with a single hidden layer and sufficient neurons, employing a non-constant, bounded, and continuous activation function, can approximate any continuous function on compact subsets of $\mathbb{R}^n$ to any desired degree of accuracy.*

That is to say, there exists a neural network that will approximate any region of attraction defined over a compact input set. However, there is still the issue of determining appropriate parameters for such a network.

C. Training Neural Networks

Generally, the parameters for a given neural network are trained by minimizing the value of some loss function with respect to the weights and biases via iterative gradient descent. While many algorithms for this exist, we will use the ADAM method in this work.

Definition 4 (ADAM Method for Gradient Descent [22])**.** The ADAM (Adaptive Moment Estimation) optimization algorithm for gradient descent is defined by:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t, \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \\ \hat{m}_t &= \frac{m_t}{1-\beta_1^t}, \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t}, \\ \theta_{t+1} &= \theta_t - \frac{\gamma \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}. \end{aligned}$$

Here, m_t and v_t are estimates of the first and second moments of the gradients, g_t is the gradient at time step t, β_1 and β_2 are decay rates, and η is the learning rate. θ corresponds to the parameters of the network.

Loss functions measure the discrepancy between the training data and the model predictions and serve as a metric to minimize in the optimization. Since we will cast learning the region of attraction as a binary classification problem, a key loss function is the weighted Binary Cross-Entropy Loss.

Definition 5. Given class labels "0" and "1", the *Weighted Binary Cross-Entropy* loss is given by:

$$\text{WBCE} = -w_1 y \log(p) - w_0 (1-y) \log(1-p)$$

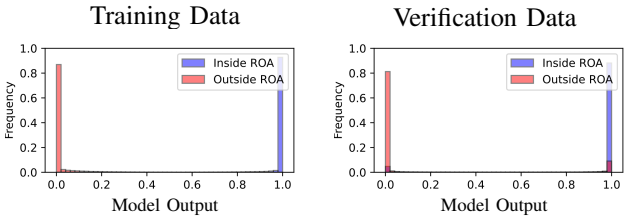
where y is the true label, p is the predicted probability that the class is labeled as 1, and w_0 and w_1 are the weights for the "0" and "1" classes, respectively.

Definition 6. The *spectral norm* of a matrix $W \in \mathbb{R}^{n_{l+1} \times n_l}$, denoted as $\|W\|_2$, is defined as:

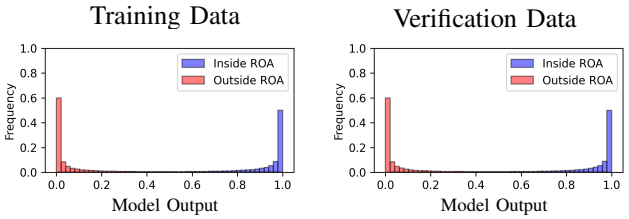
$$\|W\|_2 = \sqrt{\lambda_{\max}(W^\top W)}$$

where $\lambda_{\max}$ represents the largest eigenvalue of $W^T W$.

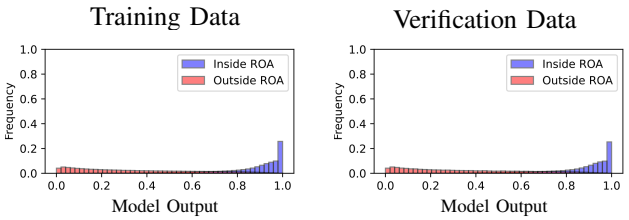
Limiting the spectral norm of each weight matrix allows us to control the Lipschitz constant of the resulting function [23], and will be useful as we are concerned with modeling regions of attraction of Lipschitz continuous dynamical systems. Note that explicitly restricting the spectral norm of the model may not be essential if our data comprehensively covers the input space, but achieving such dense coverage can be challenging in systems with a large number of dimensions. Manually limiting the spectral norm will be valuable in our applications.



(a) **Spectral norm limit that is too high/no spectral norm limit** leads to overfitting to the training data, poor generalization, and mislabeling of data outside the training set.



(b) **Appropriate spectral norm limit** allows for good distinction between labels and generalization to data not in the training set.



(c) **Spectral norm limit that is too low** leads to poor distinction between different label groups.

Fig. 3: Comparison of spectral norm limiting on model training result. As seen, if the spectral norm limit is too high, or no limit is used, then the model can overfit the training data and result in poor generalization. If the limit is too low, then the model is unable to properly capture the distinctions in underlying data. A properly chosen limit can provide a balanced model.

III. Contribution

In this work, we aim to determine models for bounded regions of attraction for systems of the form described in (1) via neural networks. These predictions are to be used in a motion primitive transition framework as conditions for switching between primitive behaviors, both in nominal conditions and in an online adaptive capacity to effectively handle unknowns. Though accuracy is a goal, completely accurate characterization is not necessary for this use case, a fact we will take advantage of in the implementation.

A. Generating Training Data

To generate training data, we begin by constructing the conservative, explicit region of attraction using a suitable method as described in Section I. The best method may depend on the particular system, but recall that the method only needs to be tractable and satisfy the definition in Section II for our purposes. Second, we consider a bounded set $\overline{\mathcal{S}}$,

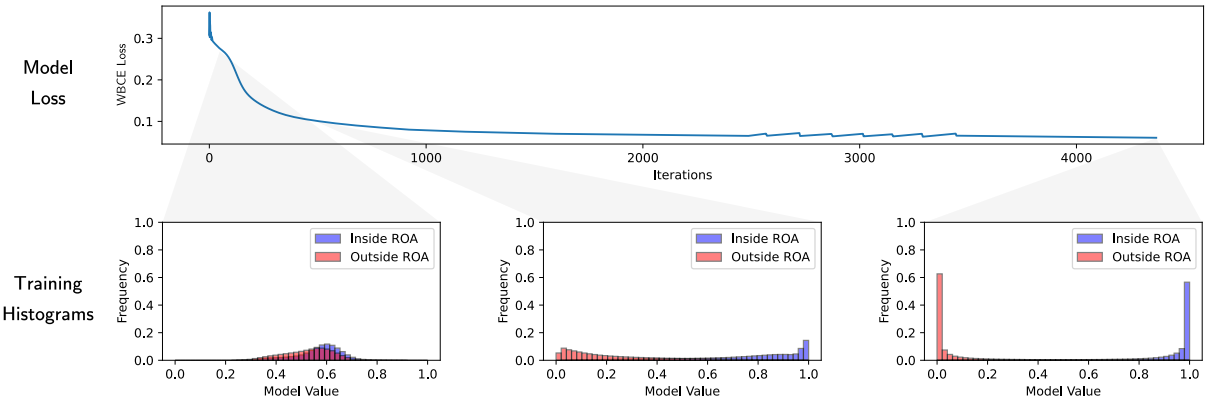


Fig. 4: As more iterations are executed in the training process, the weighted binary-cross entropy loss decreases, indicating that the model is predicting the data labels with higher accuracy. This is elucidated with histogram snapshots. Early in the training, there is poor accuracy in the model prediction, but as training progresses, the "Outside ROA" samples are predicted to be close to zero, and the "Inside ROA" samples are predicted to be close to one.

as the set where we wish the model inference to be valid. The choice of this is user-defined, and a typical choice is the bounded set of joint and velocity limits for robotic systems. We then draw the following random samples uniformly:

$$\begin{aligned} x_0 &\sim U(\overline{\mathcal{S}}) \\ \xi_0 &\sim U(\Xi) \\ t_0 &\sim U([0, t_{\max}]). \end{aligned}$$

We then have a dynamical system:

$$\dot{x} = f(x, k(x, \xi_0, t_0))$$

To label the data, we numerically compute the flow $\phi_T(x_0, \xi_0, t_0)$ over some finite time $t \in [0, T]$ and leverage the ideas behind Theorem 1:

$$\text{Label} = \begin{cases} 1, & \text{if } \phi_T(x_0, \xi_0, t_0) \in \mathcal{E}(x_0, \xi_0, t_0) \\ & \quad \text{and } \phi_t(x_0, \xi_0, t_0) \in \overline{\mathcal{S}} \\ 0, & \text{if } \phi_T(x_0, \xi_0, t_0) \notin \mathcal{E}(x_0, \xi_0, t_0) \\ & \quad \text{or } \phi_t(x_0, \xi_0, t_0) \notin \overline{\mathcal{S}} \end{cases}$$

Note that these labels will only coincide with the true region of attraction as $T \to \infty$, but with large enough finite T, we can achieve sufficient accuracy. Additionally, if $\mathcal{E}$ is forward invariant, we can relax the first condition from $\phi_T(x_0, \xi_0, t_0) \in \mathcal{E}(x_0, \xi_0, t_0)$ to $\phi_t(x_0, \xi_0, t_0) \in \mathcal{E}(x_0, \xi_0, t_0)$, potentially allowing us to terminate the numerical integration earlier than T and improve sampling performance. In addition to the sampled data, we also include the sampled equilibrium points, $x^*(x_0, \xi_0, t_0)$, in the training data set, as this must lie within the region of attraction.

B. Training the Model

Consider a neural network $N : \mathbb{R}^n \times \mathbb{R}^a \times \mathbb{R} \to [0, 1]$ with input (x, ξ_0, t) and output a prediction that indicates the likelihood of $x \in \mathcal{S}(x, \xi_0, t)$. The network has L layers and the number of nodes in each layer is denoted by $\{n_1, n_2, \ldots, n_L\}$.

Additionally, given that our dynamical system is assumed to be Lipschitz continuous, we expect discrepancies in the finite-time flow to be bounded. That is:

$$\|\phi_T(x_0, \xi_0, t_0) - \phi_T(x_0', \xi_0, t_0)\| \leq L\|x_0 - x_0'\|, L > 0$$

where x_0, x_0' are initial conditions and ξ_0, t_0 are arguments and initial time to the control law. This leads us to expect that initial conditions that are close will result in similar flows and, thus similar labels. Since our neural network outputs a continuous label prediction between 0 and 1, we expect the model to also be Lipschitz continuous. To enforce this Lipschitz continuity, we utilize spectral norm limiting:

$$\|W_l\|_2 \leq C, \quad l = 1, ..., L$$

on the weight matrix W_l of each layer. The constant C could potentially be inferred if the collected data is dense enough, but in practice becomes a hyperparameter.

To train the model, we use the ADAM algorithm with the weighted binary cross entropy loss function. With s_0 and s_1 representing the number of samples labeled "0" and "1", respectively, the weights in the loss function are chosen so that labels are normalized:

$$\begin{aligned} w_0 &= \frac{s_1}{s_0 + s_1} \\ w_1 &= \frac{s_0}{s_0 + s_1} \end{aligned}$$

C. Online Adaptation

While the offline training procedure described in the previous section can provide effective results, it fails when the environment used to numerically compute the flow differs substantially from the operation environment. To combat this, an online adaptation procedure is included to correct for prediction errors during operation.

When compared to the offline procedure, there are two additional challenges we must consider: 1) we will have limited data, and 2) we must adapt quickly. Our strategy will be to rely on the offline model as much as possible but

still allow for data collected online to meaningfully correct our predictions.

As before, to label data, we first must determine the flow $\phi_t(x_0, \xi_0, t_0)$. Rather than computing it numerically, we measure it directly from data by storing a buffer of length T_{buff} seconds and mapping state to its initial condition $t \in [0, T_{\text{buff}}]$ seconds earlier. In contrast to the offline data generation, we will be very data-sparse, since we will not be able to explore the full state space. Thus, we will label points all along the flow, as we declined to do offline to favor more informative samples. Additionally, we must use a much more restrictive description of $\mathcal{E}$, typically a small norm bound about the setpoint, as differences in the environment can invalidate the previous approximations. We will only retrain when necessary – when the offline model mispredicts behavior. Generally, this could be due to a transient disturbance or an environmental change, and some tuning is required to balance the sensitivity of the retraining to these effects. The retraining process is identical to the training from the previous section, except for an additional loss term:

$$\mathcal{R} = \alpha \sum_{l=1}^{L} (\|W_l - W_{l,\text{offline}}\|^2 + \|b_l - b_{l,\text{offline}}\|^2) \quad (2)$$

This penalizes deviations from the original, offline trained weights, and the hyperparameter, α, allows us to tune the importance of the online update. To speed up adaptation, the retraining will typically be done with much fewer iterations.

IV. Application to a Quadruped

To validate our method in an experimental setting, we apply this procedure to several motion primitives on a Unitree A1 quadruped. The primitives include "Lie", "Stand", a 'Quasistatic Walk", "Walking Trot" behaviors and are implemented in a motion primitive transition framework [1]. This framework uses attributes of the motion primitives, including regions of attractions and setpoints of controllers to create an implicit "motion primitive graph". Searching this graph finds an appropriate sequence of behaviors to drive the system to a desired motion primitive, in a manner that is robust to initial conditions and disturbances. However, estimating regions of attraction for a high dimensional system, such as the quadruped, is difficult, and represents a good application for our proposed method.

A. Input Space

We start with the quadruped model: we have n-dimensional configuration $q \in \mathcal{Q}$ giving overall state $x = (q, \dot{q}) \in \mathcal{X} = \mathcal{Q} \times T\mathcal{Q}$ with $n = 18$. We have $m = 12$ actuated degrees of freedom for control input $u \in \mathcal{U} \subset \mathbb{R}^m$. The *hybrid-dynamic* nature of the system leads to several domains of operation to be considered. These domains are marked by the contact state of each foot, denoted by a contact vector $c \in \{0,1\}^{|\mathcal{N}_c|}$ where $\mathcal{N}_c = \{1,2,3,4\}$ the set of considered contacts, in this case the quadruped's feet. We consider and model no-slip via *holonomic constraints* dependent on the number of active contacts, i.e. the hybrid domain. We have our system dynamics for a specific domain in control affine form as:

$$\dot{x} = \underbrace{\begin{bmatrix} \dot{q} \\ -D(q)^{-1}(H(q,\dot{q}) - J_c(q)^\top F) \end{bmatrix}}_{f(x)} + \underbrace{\begin{bmatrix} 0 \\ D(q)^{-1}B \end{bmatrix}}_{g(x)} u,$$

where $D(q) \in \mathbb{R}^{n \times n}$ is the mass-inertia matrix, $H(q, \dot{q}) \in \mathbb{R}^n$ accounts for the Coriolis and gravity terms, $B \in \mathbb{R}^{n \times m}$ is the actuation matrix, $J_c(q)$ is the contact Jacobian for the specific hybrid domain, and $F \in \mathbb{R}^h$ is the constraint wrench.

For each primitive, there is an associated control law, $u = k(x, \xi, t)$ to arrive at the closed loop dynamics for each motion primitive on the quadruped. The specific details of the control laws are not important in the context of this paper, but details can be reviewed in [1].

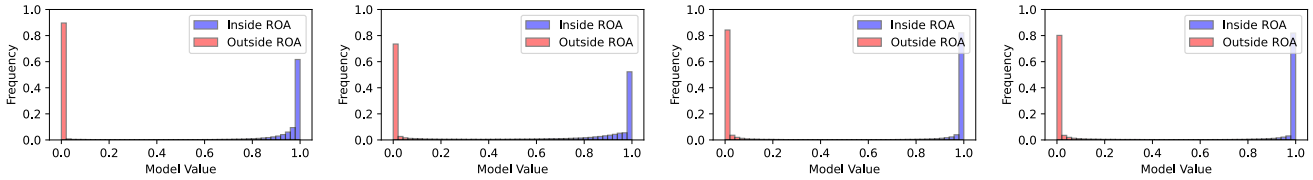
In addition to the state space x, we will also include the contact state, c, as an input to the neural network to capture the different hybrid domains. The final input is then (x, c, ξ, t) with network $N : \mathcal{X} \times \{0,1\}^{|\mathcal{N}_c|} \times \Xi \times \mathbb{R} \mapsto [0,1]$

B. Generating the Samples

The setpoints $x^*(x, \xi, t)$ are self-reported by each primitive implementation, and the explicit region of attraction $\mathcal{E}(x, \xi, t)$ is taken to be a small norm bound around the setpoint. The bounds on the state space are taken to be the physical joint and velocity limits, and reasonable values for body motion in the world frame. We consider the task space of the feet, using kinematic bounds to prevent samples where the ground plane is violated. We then follow the procedure as described above, drawing initial condition samples uniformly from their bounded spaces. The flow is calculated using the Pinocchio rigid body dynamics library [24] to model the quadrupedal dynamics and Boost's odeint with a `runge_kutta_cash_karp54` integration scheme [25]. In total, we collected 1.5M samples for each primitive: 500K samples outside, 500K inside, and 500K setpoints as functions of the randomized initial conditions.

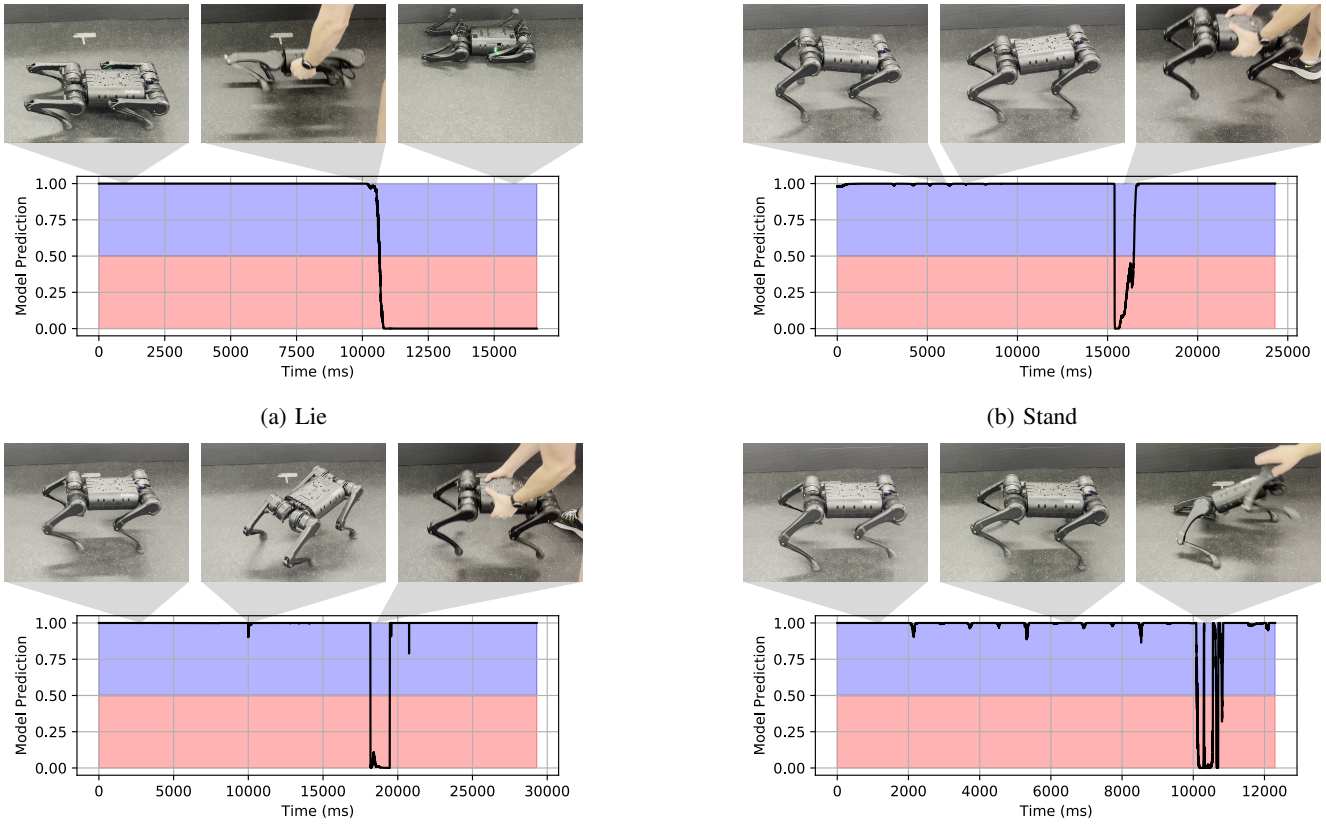
TABLE I: Performance Metrics for Region of Attraction Models

	Inside ROA			Outside ROA			
Model	Precision	Recall	F1	Precision	Recall	F1	MCC
Lie	97.48%	98.18%	97.83%	96.32%	94.93%	95.62%	0.9345
Stand	95.41%	94.95%	95.18%	89.99%	90.87%	90.43%	0.8561
Walking Trot	93.92%	98.58%	96.19%	96.85%	87.24%	91.80%	0.8827
Quasistatic Walk	97.80%	97.98%	97.89%	95.94%	95.59%	95.77%	0.9366



(a) Lie (b) Stand (c) Walking Trot (d) Quasistatic Walk

Fig. 5: Normalized Verification Data Histograms for the Motion Primitive Region of Attraction Models.



(a) Lie (b) Stand

(c) Walking Trot (d) Quasistatic Walk

Fig. 6: Model predictions for various motion primitives, under nominal operation and manually forced into invalid states.

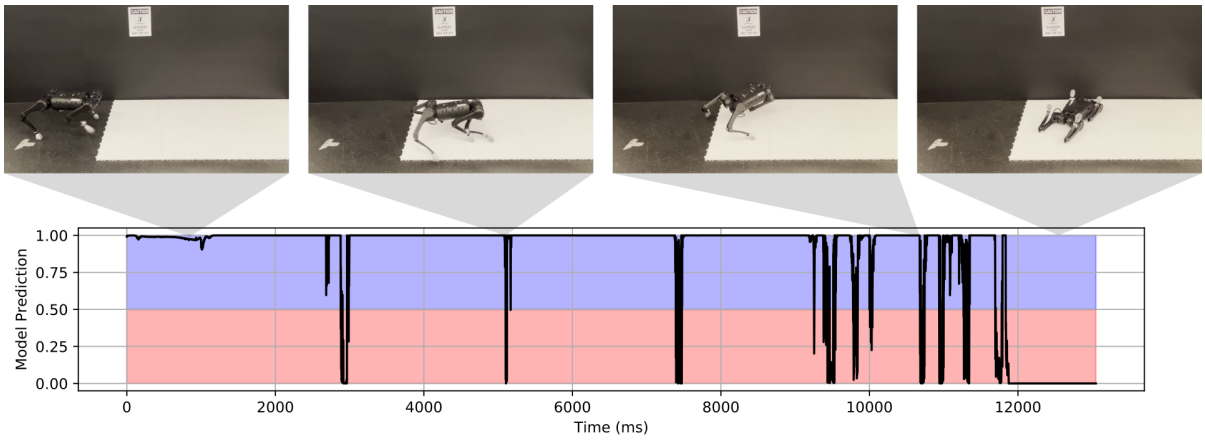
C. Training the Models

To train the models, we use the PyTorch python machine learning library [26] using the ADAM optimizer and implementation of spectral norm limiting described in Section II. We found that a network with a single hidden layer of 256 nodes gave a qualitatively good balance between accuracy and online inference speed. When training offline, we determined $\gamma = 0.01$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$ to be suitable parameters for ADAM and $C = 50$ to be an effective spectral norm limit for each layer by a basic parameter sweep. The offline model was trained for 20,000 iterations on a workstation with an Intel i7-8700K CPU, 64GB RAM, and an Nvidia Titan V, 12GB.

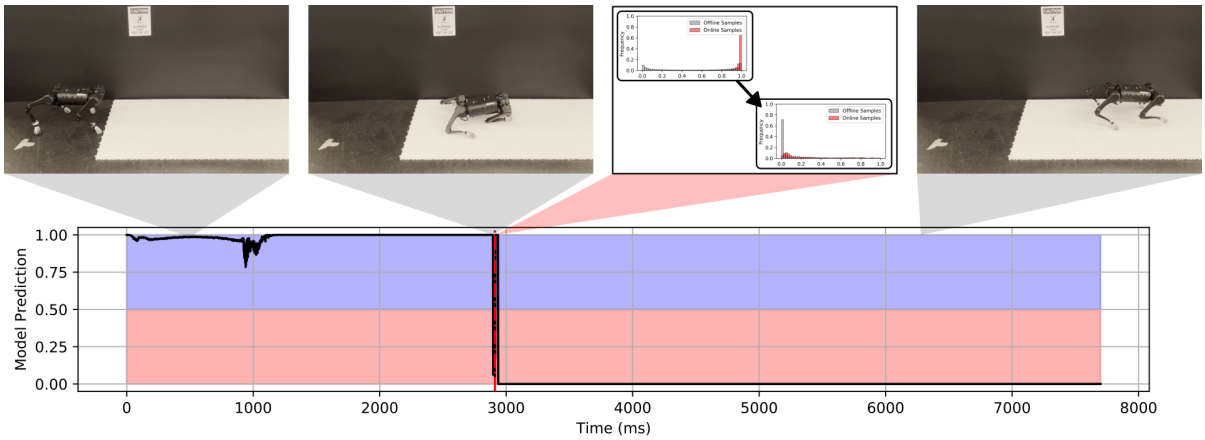
When retraining the model online in response to mispredictions, we modified the learning rate to $\gamma = 0.005$ and used a regularization loss factor $\alpha = 0.25$. We found retraining for 10 iterations was sufficient to account for the new data.

D. Verifying the Models

To evaluate the performance of models, we generate a set of verification data in the same manner as the training data, but with 50K samples for each category. Note the model is not trained on this set. We then compute typical metrics for binary classification—precision, recall, and F1 score—for both inside ROA and outside ROA classes. Precision measures the accuracy of positive predictions made by a model, while recall indicates the model's ability to identify all actual positives. The F1 score provides a balance between precision and recall via the harmonic mean. We use the Matthews Correlation Coefficient (MCC) as a combined measure, where a value of 1 indicates perfect prediction, 0 is no better than random chance, and -1 indicates opposite prediction compared to actual values. These results are summarized in Table I for each primitive and histograms on the verification datasets can be seen in Figure 5.



(a) **Region of Attraction Model without adaptation:** The model correctly predicts the region of attraction when trotting on the rubberized floor, but fails when on the synthetic ice until slipping manifests a large enough disturbance and the quadruped falls. When the disturbance from the slip subsides, the model incorrectly believes the walking trot can be executed again, and the misprediction leads to a cycle of slips and reattempts until ultimately entering an unrecoverable state.



(b) **Region of Attraction Model with adaptation:** In this case, when the first slip is encountered, the misprediction is recognized, and a buffer of online data is relabeled and used to retrain the model online. This allows the quadruped to correctly predict that the walking trot can no longer be performed, and the motion primitive transition framework selects the lowest cost feasible behavior, the quasistatic walk. The quadruped can then progress along the synthetic ice.

Fig. 7: Quadrupedal walking trot experiments on unmodeled "synthetic ice" environment, with and without online adaptation.

E. Hardware Experiments

To demonstrate our method on hardware, we conducted two types of experiments using the learned regions of attraction. The first aims to test the offline model's performance in conditions similar to the training environment, and the second explores the online model retraining in response to an environment the model was not previously trained on.

F. Representative Real-World Environment

We first test the performance of our models in a real-world environment that is consistent with the simulated training environment – that is, flat ground with a consistent, reasonable ground friction coefficient. For this, we used our indoor lab environment with a flat floor covered in rubber tiles. Here the model performed as expected by simulation, and was able to predict successful nominal operation for the "Lie", "Stand", "Walking Trot", and "Quasistatic Walk" motion primitives with varying arguments. Additionally, when manually forced into invalid states by an external disturbance, the model correctly predicts that the state is outside of the region of attraction. These tests are shown in Figure 6 and the supplementary video [27].

G. Unrepresentative Real-World Environment

To test in an environment that differs from the training environment, we set up a scenario where the quadruped would encounter low ground friction. The low-friction environment was achieved by using high-density polyethylene "synthetic ice" [28] and low-density polyethylene booties on the feet of the quadruped. The quadruped was commanded to perform a walking trot that encountered this slick surface, but since the model was not trained with such conditions, it incorrectly predicts that the state is within the region of attraction for the walking trot. Without any online adaptation, the motion primitive transition framework uses this incorrect

prediction to generate a plan that is doomed to fail and causes the quadruped to continually slip and retry the walking trot behavior. Eventually, it falls on its back – a state outside of the region of attraction regardless of friction.

When adaptation is enabled, the quadruped again slips on the slick surface, but the mispredicted data is used to retrain the model online. Using the updated model, the motion primitive transition search can no longer find a valid sequence to reach the region of attraction for "walking trot", and instead selects the closest behavior according to search cost function, "quasistatic walk". This behavior is capable of walking on the slick surface, and its model predictions are correct despite the difference from the training environment. The quadruped is able to proceed with walking. This experiment can be seen in Figure 7 and the supplementary video [27].

V. CONCLUSIONS

In this work, we focused on estimating regions of attraction in dynamical systems. Our proposed method utilizes neural networks to model an estimate of the region of attraction. To train the model, data is generated offline by uniformly sampling initial conditions from a bounded state space, including arguments and time for systems with closed-loop control laws, and labeled based on their finite time flow and entry into a conservative, but tractable "explicit region of attraction". The model is primarily trained offline, but to account for differences between the modeled dynamics and the real world, online retraining of the model is used to correct for mispredictions. We applied this procedure to several closed-loop primitive behaviors on a quadrupedal robot as part of a motion primitive transition framework, successfully demonstrating its capability in estimating regions of attraction to enable effective motion primitive transitions across various behaviors and environmental settings.

For future work, the authors would like to explore training the offline model on more varied environments, potentially using techniques like domain randomization and including additional sensor data in the model input to reduce reliance on online retraining in new environments. Despite this, we still expect some online retraining to be necessary and would like to study improving the online retraining process and making more effective use of limited data available. Finally, we would like to expand our experimental results to encompass more motion primitives and other robotic systems.

REFERENCES

[1] W. Ubellacker and A. D. Ames, "Robust locomotion on legged robots through planning on motion primitive graphs," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[2] A. Lyapunov, "The general problem of the stability of motion," *International Journal of Control*, vol. 55, no. 3, pp. 531–534, 1992. [Online]. Available: https://doi.org/10.1080/00207179208934253

[3] J. P. LaSalle, *The Stability and Control of Discrete Processes*. Springer, 1968.

[4] V. Zubov, *Methods of A.M: Lyapunov and their Application*. P. Noordhoff Limited, 1964. [Online]. Available: https://books.google.com/books?id=DGCEzQEACAAJ

[5] W. Hahn, *Stability of Motion*, ser. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, 1967. [Online]. Available: https://books.google.com/books?id=svhQAAAAMAAJ

[6] G. Chesi, "Estimating the domain of attraction for non-polynomial systems via lmi optimizations," *Automatica*, vol. 45, no. 6, pp. 1536–1541, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109809000971

[7] P. Giesl and S. Hafstein, "Review on computational methods for lyapunov functions," pp. 2291–2331, 2015. [Online]. Available: /article/id/ee7e4111-9a13-4e44-95c8-14c5557278fc

[8] E. Najafi, R. Babuska, and G. Lopes, "A fast sampling method for estimating the domain of attraction," *Nonlinear Dynamics*, vol. 86, 10 2016.

[9] H.-D. Chiang and J. Thorp, "Stability regions of nonlinear dynamical systems: a constructive methodology," *IEEE Transactions on Automatic Control*, vol. 34, no. 12, pp. 1229–1241, 1989.

[10] I. Mitchell, A. Bayen, and C. Tomlin, "A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games," *Automatic Control, IEEE Transactions on*, vol. 50, pp. 947 – 957, 08 2005.

[11] G. Yuan and Y. Li, "Estimation of the regions of attraction for autonomous nonlinear systems," *Transactions of the Institute of Measurement and Control*, vol. 41, pp. 106 – 97, 2019.

[12] H. D. Chiang, M. Hirsch, and F. F. Wu, "Stability regions of nonlinear autonomous dynamical systems," *IEEE Transactions on Automatic Control*, vol. 33, pp. 16–27, 1988.

[13] J. Zaborszky, G. Huang, B. Zheng, and T.-C. Leung, "On the phase portrait of a class of large nonlinear dynamic systems such as the power system," *IEEE Transactions on Automatic Control*, vol. 33, no. 1, pp. 4–15, 1988.

[14] P. A. Parrilo, "Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization," Ph.D. dissertation, California Institute of Technology, 2000. [Online]. Available: https://thesis.library.caltech.edu/1647/

[15] W. Tan and A. Packard, "Stability region analysis using polynomial and composite polynomial lyapunov functions and sum-of-squares programming," *IEEE Transactions on Automatic Control*, vol. 53, pp. 565–571, 2008.

[16] L. Khodadadi, B. Samadi, and H. Khaloozadeh, "Estimation of region of attraction for polynomial nonlinear systems: A numerical method," *ISA transactions*, vol. 53, 09 2013.

[17] F. Berkenkamp, R. Moriconi, A. Schoellig, and A. Krause, "Safe learning of regions of attraction for uncertain, nonlinear systems with gaussian processes," 12 2016, pp. 4661–4666.

[18] S. Chen, M. Fazlyab, M. Morari, G. J. Pappas, and V. M. Preciado, "Learning region of attraction for nonlinear systems," in *2021 60th IEEE Conference on Decision and Control (CDC)*, 2021, pp. 6477–6484.

[19] Y. Shen, M. Bichuch, and E. Mallada, "Model-free learning of regions of attraction via recurrent sets," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, 2022, pp. 4714–4719.

[20] W. Ubellacker, N. Csomay-Shanklin, and A. D. Ames, "Approximating regions of attraction via flow-control barrier functions and constrained polytope expansion," in *Proceedings of the American Control Conference (ACC)*, 2024, accepted.

[21] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0893608089900208

[22] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.

[23] P. L. Bartlett, D. J. Foster, and M. J. Telgarsky, "Spectrally-normalized margin bounds for neural networks," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017.

[24] J. Carpentier, F. Valenza, N. Mansard *et al.*, "Pinocchio: fast forward and inverse dynamics for poly-articulated systems," https://stack-of-tasks.github.io/pinocchio, 2015–2021.

[25] Boost, "Boost C++ Libraries," http://www.boost.org/, 2021, last accessed 2021-02-28.

[26] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[27] Supplemental Video, https://youtu.be/7Gu2ej5FmEI.

[28] Better Hockey, "Synthetic ice tiles," 2024, accessed: 2024-01-05. [Online]. Available: https://www.betterhockey.com/shop/flooring-tiles