Guaranteed Conformance of Neurosymbolic Models to Natural Constraints

Kaustubh Sridhar[1] KSRIDHAR@SEAS.UPENN.EDU
Souradeep Dutta[1] DUTTASO@SEAS.UPENN.EDU
James Weimer[2] JAMES.WEIMER@VANDERBILT.EDU
Insup Lee[1] LEE@SEAS.UPENN.EDU
[1]*PRECISE Center, University of Pennsylvania,* [2]*Vanderbilt University*

Editors: N. Matni, M. Morari, G. J. Pappas

Abstract

Deep neural networks have emerged as the workhorse for a large section of robotics and control applications, especially as models for dynamical systems. Such data-driven models are in turn used for designing and verifying autonomous systems. They are particularly useful in modeling medical systems where data can be leveraged to individualize treatment. In safety-critical applications, it is important that the data-driven model is conformant to established knowledge from the natural sciences. Such knowledge is often available or can often be distilled into a (possibly black-box) model. For instance, an F1 racing car should conform to Newton's laws (which are encoded within a unicycle model). In this light, we consider the following problem - given a model M and a state transition dataset, we wish to best approximate the system model while being a bounded distance away from M. We propose a method to guarantee this conformance. Our first step is to distill the dataset into a few representative samples called memories, using the idea of a growing neural gas. Next, using these memories we partition the state space into disjoint subsets and compute bounds that should be respected by the neural network in each subset. This serves as a symbolic wrapper for guaranteed conformance. We argue theoretically that this only leads to a bounded increase in approximation error; which can be controlled by increasing the number of memories. We experimentally show that on three case studies (Car Model, Drones, and Artificial Pancreas), our constrained neurosymbolic models conform to specified models (each encoding various constraints) with order-of-magnitude improvements compared to the augmented Lagrangian and vanilla training methods.[1]

Keywords: Deep neural networks, prototypes, robotics, medical devices

1. Introduction

Deep neural networks (DNNs) are capable of learning highly-complex relationships between input data and the expected output. This permits training and validation of large models in robotics and medicine (Djeumou et al., 2022; Kushner et al., 2020; Shi et al., 2019), enabling designers to comfortably achieve small approximation errors. But the caveat that comes with this flexibility is the lack of generalization when pushed outside of the training distribution. We refer to the experiments in Narasimhamurthy et al. (2019) as an example. One of the instances it covers corresponds to that of Newton's first law. The neural network dynamics model of a car should predict that, given zero throttle and when at rest, the car should continue to remain at rest. The neural network model trained on real vehicle trajectory data in Goldfain et al. (2019) failed to conform to this simple property.

1. Our code can be found at: https://github.com/kaustubhsridhar/Constrained_Models

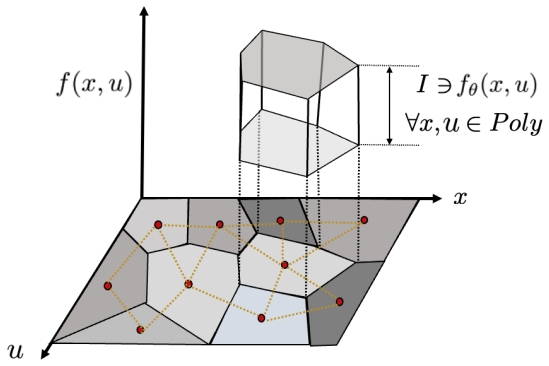

Figure 1: Depiction of our neurosymbolic algorithm. First, the input $\mathcal{X} - \mathcal{U}$ plane is partitioned into polyhedrons using a Neural-Gas. For inputs from each polyhedron, we generate sound under-approximations of the model M's output. Next, we learn the dynamics f_θ that is constrained (by construction) to respect these interval constraints.

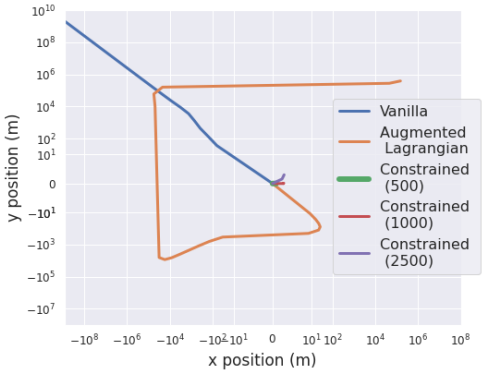

Figure 2: Trajectories generated from learned car dynamics models, starting at rest at the origin, with zero control inputs for 20 timesteps. Our neurosymbolic *constrained* models (with varying memories) respect Newton's first law of motion (and remain at rest) unlike *vanilla* and *augmented lagrangian* neural networks that drift away from the origin.

A very similar situation happens in the case of the glucose-insulin dynamics model for an artificial pancreas, a device for patients with type-1 diabetes. This property has been studied in Kushner et al. (2020), where it was found that deep neural network models could easily generate predictions that can be fatal for the patient.

However, these challenges are much less prevalent in models which are typically informed by the different scientific disciplines. Examples of this include models based on mechanical properties of robotic systems (Rajamani, 2011), aerodynamic properties of drag and lift (Mahony et al., 2012), physiological models of the human body (Man et al., 2014; Chen et al., 2015) and alike. The *advantage* of using models (rather than atomic constraints) is that they encompass a wider range of desirable properties quite naturally. In robotics, it is common to find such high-fidelity physics-engine-based simulators (Dosovitskiy et al., 2017; Coumans and Bai, 2016–2019; Todorov et al., 2012). In medical applications, examples include artificial pancreas simulators (Man et al., 2014; Chen et al., 2015). Unfortunately in practice, such models can be of black-box nature, allowing only samples to be observed. Our goal is to use such models to inviscate a deep neural network into conformal behavior.

In this work, we propose a method that guarantees the satisfaction of natural constraints by constructing a wrapper for the DNN based on symbolic information. This is achieved through a novel neural gas based partitioning technique and estimation of a model M's output ranges. Such a guarantee does not come for free, but shows up as a slightly higher approximation error (which can be attributed to the black-box nature of model M). Our contributions can be listed as: 1) A novel memory-based method to constrain neural network dynamics models with guarantees. 2) A theoretical guarantee that our memory-based constraining method guarantees conformance with only a bounded increase in approximation error. 3) Results on three case studies demonstrating that we outperform augmented Lagrangian methods for constraint satisfaction by a few orders of magnitude.

2. Related Work

Enforcing constraints on neural networks: Imposing constraints on deep neural networks has been studied from various perspectives (Djeumou et al., 2022; Finzi et al., 2020; Márquez-Neila et al., 2017; Ravi et al., 2019; Lu et al., 2021b; Dener et al., 2020; Fioretto et al., 2020; Nandwani et al., 2019; Kervadec et al., 2022). These include constraints of symmetry and contact forces for dynamical systems in Djeumou et al. (2022), suitable constraints for specific Lagrangian or Hamiltonian neural networks in Finzi et al. (2020), human pose constraints in Márquez-Neila et al. (2017), path norm constraints on resnets in Ravi et al. (2019), partial differential equation (PDE) constraints for inverse design in Lu et al. (2021b), Focker-Planck constraints for fusion in Dener et al. (2020), fairness constraints in Fioretto et al. (2020), language label constraints in Nandwani et al. (2019), and segmentation constraints in Kervadec et al. (2022). All of these methods rely on the augmented Lagrangian method to train constrained neural networks. Solving the dual problem, *i.e.* converging to a stationary point for the min-max optimization is challenging with neural networks and non-convex constraints (Márquez-Neila et al., 2017). Further, the process is data-hungry and generalizes poorly in out-of-distribution data (Narasimhamurthy et al., 2019; Márquez-Neila et al., 2017; Ravi et al., 2019). Our focus in this work is to leverage the benefits of the augmented Lagrangian approach (its flexible loss function) but constrain the neural network by design, and with a guarantee, to remain within desirable output bounds computed using models that encode all desired constraints. In the process, we obtain several orders of magnitude reduction in constraint loss and learn with very few gradient steps.

Physics informed neural networks for dynamics models: Although our focus is on enforcing constraints, we also briefly discuss related ideas in physics-informed neural networks (Raissi et al., 2019; Márquez-Neila et al., 2017; Lu et al., 2021a; Lutter et al., 2019; Cranmer et al., 2020; Greydanus et al., 2019). Physics-informed architectures for dynamical systems in particular have been explored via specific Neural ODE structures for a class of systems (Duong and Atanasov, 2021; Zhong et al., 2019; Roehrl et al., 2020; Matsubara et al., 2020; Gupta et al., 2020; Shi et al., 2019) or via a broader Neural ODE structure for a class of vector fields (Djeumou et al., 2022), all towards learning continuous-time dynamics for robotics applications. Our constraining framework can be applied around any such Neural ODE. But moreover, our constraints can include black-box models and scale quickly to any state and action space unlike NeuralODEs which are restricted to systems with rigorous mathematical models (Raissi et al., 2019). Further, to present a general solution, we make no assumption on the architecture and to extend to applications beyond dynamics models in robotics (such as medicine, computing systems, and operations research), we learn discrete-time dynamics models in our experiments rather than continuous-time dynamics models.

3. Problem Formulation

Consider a discrete time non-linear dynamical system $x_+ = f(s)$, where $\mathcal{S} := \mathcal{X} \times \mathcal{U}$, $x \in \mathcal{X}$ is the state of the system, and $u \in \mathcal{U}$ is the control input. We denote $f : \mathcal{S} \mapsto \mathcal{X}$ to be the unknown discrete-time non-linear map that captures the system dynamics. We assume access to a dataset $D = (s_0, x_0), (s_1, x_1), \ldots, (s_N, x_N)$ drawn from distribution $\mathcal{D}$, such that $x_{t+1} = f(s_t)$. Usually, the goal is to estimate f with a function f_θ, where $\theta \in \mathbb{R}^p$ is potentially the parameters of a neural network. Typically, the goal of an algorithm which estimates θ is usually to reduce approximation error on the training dataset D. In addition to this, sometimes it is desirable that the

estimated model f_θ satisfies physics-informed constraints (Cranmer et al., 2020). Next, we define a few relevant concepts.

Definition 1 (Model Constraint) *Assume a model $M : \mathcal{S} \mapsto \mathcal{X}$, and a parameter δ. Then the model constraint $\psi^{\delta}_{M,f_\theta} : \mathcal{S} \mapsto \mathbb{R}$ is* True *iff $\psi^{\delta}_{M,f_\theta}(s) > 0$ where $\psi^{\delta}_{M,f_\theta}(s) := \delta - ||M(s) - f_\theta(s)||_\infty$.*

Here we assume M to be Lipschitz continuous with constant L. We state our problem next.

Problem Statement 3.1 (Constrained Neural Network) *Find a function $f_\theta(.) : \mathcal{S} \mapsto \mathcal{X}$, which minimizes the approximation error on dataset D, while satisfying the constraints given by $\psi^{\delta}_{M,f_\theta}$. That is find $\theta^* = \underset{\theta}{\operatorname{argmin}} \ \frac{1}{N}\sum_{i=1}^{N} \|f_\theta(s_i) - x_i\|_2$, subject to, $\psi^{\delta}_{M,f_\theta}(s) > 0$.*

4. Overall Approach

To restate, we want our estimated model f_θ to approximate our training data while respecting the constraint imposed by the model M. We use the following intuition in our approach: if restricted to a small enough input region $\hat{\mathcal{S}}$ the output of the model M can be under-approximated by a set $\mathcal{X}_o$. If we can ensure that the predictions of f_θ stay within this interval then we can bound the difference between f_θ and M, as being proportional to the size of the input-region $\hat{\mathcal{S}}$, which improves with finer partitioning of the input space. Thus, to summarize our approach, we first partition an input space into small enough input regions and for each sub-region, we estimate an interval under-approximation for the values of M which can satisfy $\psi^{\delta}_{M,f_\theta}$. Next, we train our function approximator f_θ to respect these interval constraints in each such sub-region. This is accomplished using a *constraining operator* Γ on f_θ. In Section 6 we explain a method for computing these sound under-approximations of M. Then, in Section 7, we explain the constraining operator and bound the approximation error incurred due to this operator. Figure 1 displays our approach.

5. Preliminaries

We define the idea of a *neural gas* (Fritzke, 1994; Prudent and Ennaji, 2005; Martinetz et al., 1993). From a given set of points embedded in a metric space, a growing neural gas algorithm has the ability to learn important topological relations in the form of a graph of prototypical points. It uses a simple Hebb-like learning rule to construct this graph.

Definition 2 (Neural Gas) *Neural Gas $\mathcal{G} := (\mathcal{A}, \mathcal{E})$, is composed of the following two components,*

1. *A set $\mathcal{A} \subset \mathcal{S}$ of the nodes of a network. Each node $m_i \in \mathcal{A}$ is called a memory in this paper.*
2. *A set $\mathcal{E} \subset \{(m_i, m_j) \in \mathcal{M}^2, i \neq j\}$ of edges among pairs of nodes, which inform about the topological structure of the data. The edges are unweighted.*

The edges in $\mathcal{E}$ preserve the neighborhood relations among the data, and is useful in achieving a Voronoi-like partitioning of the data manifold. The graphical structure of a neural gas makes it much more appealing to algorithmically resolve neighborhood relations. For a given node m_i, let us denote $\mathcal{E}^i$ as the set of neighbors of m_i according to $\mathcal{G}$. For most practical purposes in a control setting, the spaces $\mathcal{S}$ and $\mathcal{X}$ are embedded in Euclidean spaces $\mathbb{R}^t$, and $\mathbb{R}^d$ respectively, where $t \geq d$. Where, $t - d$ is the dimension of control input. Let k be the cardinality of $\mathcal{A}$:

$\{m_1, m_2, m_3, \ldots, m_k\}$. Then, we can define the Voronoi polyhedron (Brostow et al., 1978), around a given point m_i in the following fashion.

Definition 3 (Voronoi Polyhedron) *For a point m_i , the Voronoi polyhedron $\mathcal{S}_v^i \in \mathcal{S}$ can be defined using the Euclidean distance function $d : \mathcal{S} \times \mathcal{S} \mapsto \mathbb{R}$ as,*

$$\mathcal{S}_v^i = \{s \in \mathcal{S} |\ d(s, m_i) < d(s, m_j)\ \ \forall j \in \mathcal{E}^i\}$$

In practice, constructing the Voronoi polyhedron $\mathcal{S}_v^i$ can be achieved in the following way. Given points which are neighbors m_i and m_j, it is possible to compute a line segment l_{ij} which connects them. Let us denote the perpendicular bisector of l_{ij} as the linear inequality $H_{ij}(s) > 0$. For any point s which is in the same side of H_{ij} as m_i the inequality holds. The reverse is true for the half space constraint H_{ji}. This gives us an algorithm to compute $\mathcal{S}_v^i = \bigcap_{j \in \mathcal{E}^i} H_{ij}$. Thus, given a set of k nodes the Voronoi tessellation induces a splitting of the space $\mathcal{S}$ into a set of disjoint sets $\mathcal{S}^1, \mathcal{S}^2, \ldots, \mathcal{S}^k$. We drop the subscript v for the rest of the paper. Our guarantees of constraint satisfaction is over the union of these subsets.

6. Approximating Model Constraints

Assume a (relatively small) subset $\mathcal{S}_a \subset \mathcal{S}$, and $M^j(s)$ denote the j-th output of the model at input s. We wish to compute the interval $I_a^j := [\underset{s \in S_a}{min} M^j(s), \underset{s \in \mathcal{S}_a}{max} M^j(s)]$. Assume that $\forall s \in \mathcal{S}_a$, $f_\theta^j(s) \in I'$, and $I' \subseteq I_a^j$. Where I' is the interval bound on values of f_θ^j in $\mathcal{S}_a$. Then $\underset{s \in S_a}{max} |M^j(s) - f_\theta^j(s)| \leq |I_a^j|$.

Now, in practice it is hard to precisely compute the interval I_a^j for black-box models M. Meaning that we would resort to estimating the min and max of M^j using sampling based techniques. There exists a stochastic optimization algorithm to estimate the true maxima of a Lipschitz function on a bounded domain (Mladineo, 1991). Here we follow a simple sampling based rendition to estimate I_a^j. We denote $[k]$ as the list of numbers from $0 \ldots k-1$. Next, we note the following lemma.

Lemma 4 *Let $g : \mathbb{R}^t \to \mathbb{R}$ be an L_g-Lipschitz continuous function on a closed and compact set S_a, and l and u be its estimated lower and upper bounds. Then, $\forall z \in [l, u]$, $\underset{s \in \mathcal{S}_a}{max}\ |g(s) - z| < L_g |S_a|$.*

Proof: The proof can be found in Appendix 10.1.

With $\mathcal{S}_a \subset \mathbb{R}^t$, let l and u be the estimated minima and maxima of M^j. Thus, if $\forall s \in \mathcal{S}_a$, $f_\theta^j(s) \in [l, u]$, then $\underset{s \in S_a}{max} |M^j(s) - f_\theta^j(s)| \leq L_{M^j} |S_a|$. Now, across all dimensions $j \in [d]$, let $L_M = max\ L_{M^j}$ then, $||f_\theta(s) - M(s)||_\infty < L_M |S_a|$. Assume a^*, to be the largest partition induced by the neural gas $\mathcal{G}$, then setting $\delta = L_M |S_{a^*}|$ ensures satisfaction of model constraint $\psi_{M, f_\theta}^\delta$ in Definition 1. This bound can be made much tighter in practice if the model M is known in an analytical form. Allowing tight computations of its limits possible using techniques like interval arithmetic and Taylor models (Goubault and Putot, 2022)

So, given a set $\mathcal{S}$ and using neural gas $\mathcal{G}$, we have a partitioning of $\mathcal{S} = \bigcup_{i \in [k]} \mathcal{S}^i$. Let us denote this set of partitions of $\mathcal{S}$ as $\mathcal{P}_\mathcal{S} := \{\mathcal{S}^1, \mathcal{S}^2, \ldots, \mathcal{S}^k\}$. Also, for each subset $\mathcal{S}^i$ we can compute range estimate $I_i \subset \mathcal{X}$, which respects the constraint $\psi_{M, f_\theta}^\delta$. In the following discussions, let us refer to this constraint map as $\mathsf{C}_{M,\delta} : \mathcal{P}_\mathcal{S} \mapsto \mathcal{I}^d$. Where, $\mathcal{I}^d$ is a $d-$dimensional interval in $\mathbb{R}^d$. For a subset in $\mathcal{P}_\mathcal{S}$, $\mathsf{C}_{M,\delta}$ returns the appropriate output range.

7. Function Approximation Error

In this section we define a constraining operator on a function, and analyze the error encountered in the process. The goal of a constraining operator is to threshold the values of the function to be within certain desirable limits. Assume an interval $I \subset \mathbb{R}^d$, and value $x \in \mathcal{X}$, then we define a projection in the following fashion along each dimension i, $Proj_I^i(x) := I_l^i$ when $x^i \leq I_l^i$; I_u^i when $x^i \geq I_u^i$; and x^i otherwise.

Definition 5 (Constraining Operator) *A constraining operator $\Gamma_{\mathcal{P}_\mathcal{S}} : \mathcal{X}^\mathcal{S} \to \mathcal{X}^\mathcal{S}$ parameterized by the partition set - $\mathcal{P}_\mathcal{S}$, modifies functions to respect the corresponding interval constraints. For a function $F : \mathcal{S} \mapsto \mathcal{X}$, it can be defined in the following fashion,*

$$\Gamma_{\mathcal{P}_\mathcal{S}}(F(s)) := Proj_{\mathsf{C}_{M,\delta}(\mathcal{S}_q)}(F(s)) \text{ where, } s \in \mathcal{S}_q \text{ and, } \mathcal{S}_q \in \mathcal{P}_\mathcal{S}$$

Hence, the constraining operator $\Gamma_{\mathcal{P}_\mathcal{S}}$ ensures that our estimated model f_θ which attempts to approximate the true function f, also respects the constraint ψ^δ_{M,f_θ}. Even though we assume that $f \models \psi^\delta_{f_\theta,M}$, our approximation error in building the map $\mathsf{C}_{M,\delta}$ can affect the model approximation error $|f - f_\theta|$. This however as we show only leads to a bounded cost in approximation error. Which can be reduced by adopting finer partitions in $\mathcal{P}_\mathcal{S}$, that is increasing the nodes $\mathcal{A}$ in the neural gas $\mathcal{G}$.

Theorem 6 (Approximation Error) *Assume real and continuous functions $f, f_\theta : \mathcal{S} \to \mathcal{X}$, $\forall s \in \mathcal{S}$, if $||f_\theta(s) - f(s)||_\infty < \epsilon$, then $||\Gamma_{\mathcal{P}_\mathcal{S}}(f_\theta)(s) - f(s)||_\infty < 2\epsilon + \alpha \max\limits_{\mathcal{S}^k \in \mathcal{P}_\mathcal{S}} |\mathcal{S}^k|$, where α is some constant.*

Proof: Assume a generic input $s \in \mathcal{S}$, and $s \in \mathcal{S}^q$ for some $q \in [|\mathcal{P}_\mathcal{S}|]$. Additionally, let I^q be the interval constraint imposed by $\Gamma_{\mathcal{P}_\mathcal{S}}$ on f_θ using the map $\mathsf{C}_{M,\delta}$. Since the sets $\mathcal{S}$ and $\mathcal{X}$ are embedded in the real spaces $\mathbb{R}^t$ and $\mathbb{R}^d$ respectively, we can analyze the error incurred along each dimension. Also, we drop the subscript and denote the constraining operator as simply Γ since the partition remains fixed for the remainder of the results.
x_j refers to the j^{th} element of x. Let us pick a dimension $w \in [t]$, we define the lower correction set $\gamma|_{w,l} : \{s \mid \Gamma(f_\theta)(s)_w \geq f_\theta(s)_w \text{ and } s \in \mathcal{S}^q\}$. Intuitively, this is the set of points in $\mathcal{S}^q$, which need a correction due to underflow. Let us denote the difference function as $\Delta_{w,l}$,

$$\Delta_{w,l}(s) := \begin{cases} \Gamma(f_\theta)(s)_w - f_\theta(s)_w & \text{when } s \in \gamma_{w,l} \cap \mathcal{S}^q \\ 0 & \text{when } s \in \mathcal{S}_q \setminus \gamma_{w,l} \end{cases}. \tag{1}$$

We can similarly define the upper correction set $\gamma_{w,u} \subseteq \mathcal{S}^q$ and the difference function as $\Delta_{w,u}(s) = f_\theta(x)_w - \Gamma(f_\theta)(s)_w$ for $s \in \gamma_{w,u} \cap \mathcal{S}^q$ and 0 for anywhere in $\mathcal{S}^q \setminus \gamma_{w,u}$.
Now the following is true, for $s \in \gamma_{w,l}$: $0 \leq \Delta_{w,l}(s) \leq I^q_{w,l} - \min\limits_{x \in \mathcal{S}^q} f_\theta(s)_w$. This is simply due to the bound respected by $\Gamma(f_\theta)(s)_w$. Due to very similar reasons the following is true as well : $0 \leq \Delta_{w,u}(s) \leq \max\limits_{s \in \mathcal{S}^q} f_\theta(s)_w - I^q_{w,u}$. Next, we wish to bound the following quantity: $|\Gamma(f_\theta)(s)_w - f(s)_w|$. The difference between the constrained function and ground truth. Then,

$$\begin{aligned} \Gamma(f_\theta)(s)_w - f(s)_w &= f_\theta(s)_w + \Delta_{w,l}(s) - \Delta_{w,u}(s) - f(s)_w \\ &= (f_\theta(s)_w - f(s)_w) + (\Delta_{w,l}(s) - \Delta_{w,u}(s)) \end{aligned}$$

The first equality is simply because $\mathcal{S}_q$ can be expressed as a union of the following disjoint sets $\{\gamma|_{w,l} \cap \mathcal{S}_q, \gamma|_{w,u} \cap \mathcal{S}^q, \mathcal{S}^q \setminus (\gamma|_{w,u} \cup \gamma|_{w,l})\}$ .Therefore, we can write the following,

$$\Gamma(f_\theta)(s)_w - f(s)_w \leq \epsilon + (I^q_{w,l} - \min_{s \in \mathcal{S}^q} f_\theta(s)_w)$$
$$\Gamma(f_\theta)(s)_w - f(s)_w \geq -\epsilon - (\max_{s \in \mathcal{S}^q} f_\theta(s)_w - I^q_{w,u})$$

Note, the R.H.S of the above equation is negative. Then using the bound on the upper limit of absolute values, we get the following,

$$\begin{aligned} |\Gamma(f_\theta)(s)_w - f(s)_w| &\leq \epsilon + \underbrace{(I^q_{w,l} - \min_{s \in \mathcal{S}^q} f_\theta(s)_w)}_{\geq 0} + \epsilon + \underbrace{(\max_{s \in \mathcal{S}^q} f_\theta(s)_w - I^q_{w,u})}_{\geq 0} \\ &= 2\epsilon + \left(\max_{s \in \mathcal{S}^q} f_\theta(s)_w - \min_{s \in \mathcal{S}^q} f_\theta(s)_w\right) - \underbrace{(I^q_{w,u} - I^q_{w,l})}_{\text{constraining width}} \end{aligned} \tag{2}$$

Thus, we can bound $||\Gamma(f_\theta)(s) - f(s)||_\infty$ in the following fashion, for $s \in \mathcal{S}^q$:

$$\begin{aligned} ||\Gamma(f_\theta)(s) - f(s)||_\infty &\leq 2\epsilon + \max_{w \in [d]} \left(\left(\max_{s \in \mathcal{S}^q} f_\theta(s)_w - \min_{s \in \mathcal{S}^q} f_\theta(s)_w\right) - \underbrace{(I^q_{w,u} - I^q_{w,l})}_{=|I^q|_w \geq 0} \right) \\ &\leq 2\epsilon + \max_{w \in [d]} \big(L_{\theta,w} |\mathcal{S}^q|\big) = 2\epsilon + |\mathcal{S}^q| \max_{w \in [d]} \big(L_{\theta,w}\big) \end{aligned}$$

Now, setting $\alpha = L_\theta$, where L_θ is the global Lipschitz constant of f_θ, we can write,

$$||\Gamma(f_\theta)(s) - f(s)||_\infty \leq 2\epsilon + \alpha \max_{\mathcal{S}^q \in \mathcal{P}_\mathcal{S}} |\mathcal{S}^q|, \ \forall s \in \mathcal{S} \quad \square$$

We draw the reader's attention to the following terms in inequality 2: $(I^q_{w,l} - \min_{s \in \mathcal{S}^q} f_\theta(s)_w)$ and $(\max_{s \in \mathcal{S}^q} f_\theta(s)_w - I^q_{w,u})$. Similar to Lemma 4 it can be shown that these two terms decrease with the size of the set $\mathcal{S}^q$. In other words, having finer Voronoi partitions decreases the approximation error.

8. Training a Constrained Neural Network Dynamics Model

We detail our constraining operator used in practice and our overall algorithm below. The inputs to the algorithm are a state transitions dataset D containing ((state, control), (next state)) pairs (s, x), model M, and architecture $f_\theta : \mathcal{S} \to \mathcal{X}$.

Algorithm 1. First (line 1), we generate the unlabelled Ω dataset which consists of only inputs to the model $s' \in \mathcal{S}$ by sampling throughout the input space but with particular emphasis on relevant regions in $\mathcal{S}$. Next (lines 2, 3), we use the unsupervised neural gas algorithm (Martinetz et al., 1993; Fritzke, 1994) to obtain the memories. We partition the input space into voronoi cells around each memory (line 4). With model M, we obtain the upper and lower limits along each dimension of the output space (line 5). Finally, we can train the constrained neural network given below,

$$\Gamma(f_\theta)(s) = \texttt{Lo}(s) + \sigma\left(f_\theta(s)\right) \left(\texttt{Up}(s) - \texttt{Lo}(s)\right) \tag{3}$$

Algorithm 1 `Training a Constrained Neurosymbolic Dynamics Model`

Input: Dataset $\mathcal{D} = \{(s, x)_i\}_{i \in [N_{\mathcal{D}}]}$, model M, DNN architecture $f_\theta(.)$
Output: Constrained neurosymbolic dynamics model $\Gamma(f_\theta)(.)$
Parameters: Number of memories n_{memories}

1: Generate input samples $\Omega = \{(s')_i\}_{i \in [N_\Omega]}$ // Generating unlabelled dataset for conformance.
2: $D|_{inputs} = \{(s)_i\}_{i \in [N_{\mathcal{D}}]} \cup \{(s')_i\}_{i \in [N_\Omega]}$ // combine inputs in both datasets
3: Memories $\mathcal{A}$, edges $\mathcal{E} \leftarrow$ NeuralGas($D|_{inputs}$, n_{memories}) // topology of input space
4: $\mathcal{S}^1, ..., \mathcal{S}^j, ... \leftarrow$ VoronoiCells($\mathcal{A}, \mathcal{E}$) // partitions in input space
5: Compute lower and upper bounds I^j_{low}, I^j_{up} for each partition $\mathcal{S}^j$ using M. These are the limits $\texttt{Lo}(s), \texttt{Up}(s)$ for each sample s belonging to a partition $\mathcal{S}^j$.
6: Train model $\Gamma(f_\theta)(.)$ in Equation 3 with wrapper limits above and self-supervised loss in Equation 4.

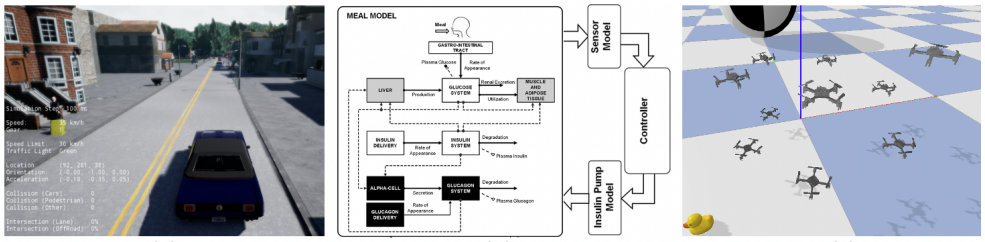

Figure 3: Depictions of high-fidelity simulators used in experiments: (a) CARLA (Dosovitskiy et al., 2017), (b) UVA/Padova Artifical Pancreas (Man et al., 2014), (c) Pybullet Drones (Panerati et al., 2021).

where $\sigma(.)$ is sigmoid. Our loss function is the augmented Lagrangian (Lu et al., 2021b) with label and constraint supervison on $\mathcal{D}$ but only constraint supervison on Ω (where $\psi^\delta_{M,\Gamma(f_\theta)}$ is from Def. 1, $\lambda_i, \mu_i \in \mathbb{R}$). Additional details of the algorithm can be found in Appendix 10.2.

$$Loss(\theta, \lambda_1, \mu_2, \lambda_2, \mu_2) = \mathop{\mathbb{E}}_{\substack{s \sim \mathcal{D} \\ s' \sim \Omega}} \Big[L\big(\Gamma(f_\theta)(s), x\big) + \Big(\lambda_1 \psi^\delta_{M,\Gamma(f_\theta)}(s) + \lambda_2 \psi^\delta_{M,\Gamma(f_\theta)}(s') + \mu_1 \mathbb{1}_{(\lambda_1 > 0 \vee \psi > 0)} \big(\psi^\delta_{M,\Gamma(f_\theta)}(s)\big)^2 + \mu_2 \mathbb{1}_{(\lambda_2 > 0 \vee \psi > 0)} \big(\psi^\delta_{M,\Gamma(f_\theta)}(s')\big)^2 \Big) \Big] \quad (4)$$

9. Experiments

Overview and baseline: We perform simulated experiments on three case studies. We create a dataset D from high-fidelity simulators that can closely represent reality in each case study. These are depicted in Figure 3. Our baseline is the augmented Lagrangian method which utilizes the loss function in equation 4 but uses a standard parameterization $f_\theta(.)$ rather than the constrained model given in equation 3. The augmented Lagrangian method lacks guarantees on constraint satisfaction with deep neural networks and non-convex constraints. We observe that augmented Lagrangian in fact fails to achieve conformance on in-distribution transitions in the test set.

Case Study 1: CARLA – Conformance of a vehicle model to unicycle dynamics with emphasis on at-rest condition. In the first case study, we collect trajectories of x position, y position, heading, velocity, yaw rate from the CARLA simulator (Dosovitskiy et al., 2017; Kaur et al., 2022) on a variety of terrains and environments (See Figure 3(a)) for our $\mathcal{D}$ dataset. With previous work Narasimhamurthy et al. (2019) having demonstrated the difficulty of learning a dynamics model

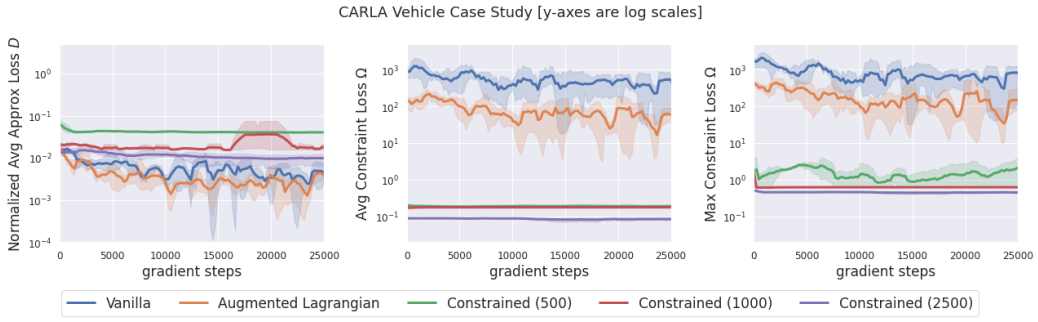

Figure 4: Plots of approximation loss on $\mathcal{D}$, average constraint loss on Ω, and maximum constraint loss on Ω (for 3 random seeds) against steps for the CARLA Vehicle case study.

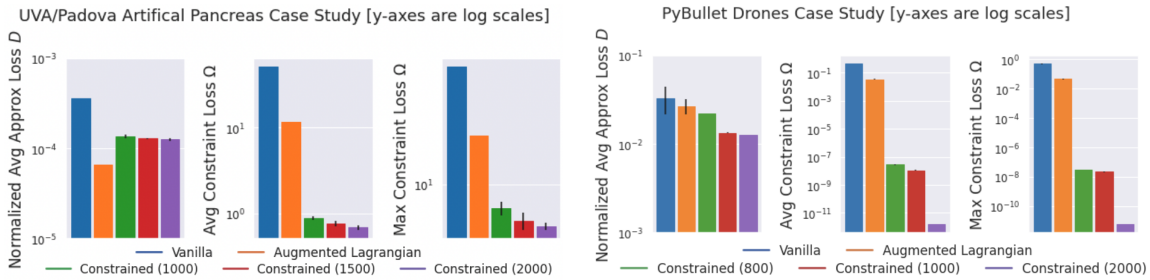

Figure 5: Bar charts of approximation loss on $\mathcal{D}$, average constraint loss on Ω, and maximum constraint loss on Ω (for 3 random seeds) after training completes for AP *(left)*, and Drones *(right)*. Plots of these metrics vs gradient steps for both case studies are in Appendix 10.4.

that predicts no change in state when a vehicle is at rest, we uniformly sample at-rest data for the augmenting dataset Ω. Unicycle dynamics (Rajamani, 2011; Sridhar et al., 2022) are chosen as the model M. This implicitly encodes the at-rest condition. We have 15,000 training points, 2000 test points in each of $\mathcal{D}$ and Ω. We select 500, 1000 and 2500 memories to observe the performance with increasing partitions in the training distribution.

We observe, in Figure 4, that the approximation loss for constrained methods is either similar to or slightly higher than the Vanilla and augmented Lagrangian. This is expected in light of Theorem 6. The average constrained loss and max constrained loss on the augmenting dataset Ω are significantly improved, *by 4 and 3 orders of magnitude respectively* for our method in comparison to Vanilla and augmented Lagrangian. Moreover, with increasing memories, the constraint loss, both average and maximum on Ω, improve consistently. We also notice that constrained training is highly data-efficient, learning in less than 300 gradient steps unlike the 12000 required by the Augmented Lagrangian. In Figure 2, we analyze each of the models' predictions starting from the origin at rest, and given zero control inputs for 20 timesteps. We clearly observe that both Vanilla and augmented Lagrangian models predict large drift to the top-left Constrained models, on the other hand, accurately predict little to no movement. This is also observed at a different random seed in 6.

Case Study 2: Artificial Pancreas (AP) – Conformance of AP models to ARMAX model that encodes glucose-insulin constraints. We collect traces of glucose, insulin and meal quantities for a patient with the UVA/Padova simulator (See Figure 3(c)) (Man et al., 2014) to create the $\mathcal{D}$

dataset. The states consist of a 30 elements– 10 historical values of glucose, insulin and meals respectively. The model is expected to predict the glucose 5 steps in the future. Each timestep spans 5 minutes. The intial value of glucose and carbohydrates are randomly chosen in $[150, 190]$, $[50, 150]$ respectively.

We also uniformly sample the state space with emphasis on low glucose initial values in $[120, 150]$ and low carbohydrates to create the Ω dataset. We have 18,750 training points, 2500 test points in each of $\mathcal{D}$ and Ω datasets. Moreover, for our model M, we train a constrained ARMAX model such that any increase in insulin, will reduce glucose. This is accomplished by constraining insulin weights to be negative in the ARMAX model. In Figure 5, we observe that approximation loss on $\mathcal{D}$ is similar across all methods with a slight advantage in the favour of our constrained training. Yet, constrained neural networks outperform vanilla and Lagrangian by an order of magnitude in conforming to the ARMAX model on the Ω and $\mathcal{D}$ datasets.

Method	Max. violation	Avg. violation
Vanilla	3.8356	1.315
Aug. Lagrangian	3.8072	1.245
Constrained (1k)	0.9157	0.0092
Constrained (1.5k)	0.2047	0.0027
Constrained (2k)	0.1775	0.0026

Table 1: Delta-monotonicity analysis of "increasing insulin, decreases glucose" violation in AP models on subsets of test data.

The delta-monotonicity property of such models in (Kushner et al., 2020), refers to the following - everything else remaining fixed, increasing insulin should lead to reduction in blood glucose prediction. In order to test this property we increase the insulin value in each input trace of test set by a random amount in $[0.6, 1.0]$ and observe the prediction. We report this in Table 1. We observe that vanilla and Lagrangian models violate the constraint by a large margin, whereas constrained models increase the prediction by nearly zero amount.

Case Study 3: PyBullet Drones – Conformance of drone models to quadrotor dynamics with emphasis on hover. We collect circular flight trajectories of 6 drones (See Figure 3(b)) with aerodynamics effects (drag, downwash, ground effect) included in the Pybullet Drones environment (Panerati et al., 2021) to create the $\mathcal{D}$ dataset. The states consist of 20 items – x, y, z positions and velocities; roll, pitch, yaw and their rates; quaternions, and rpms of each of the four motors. The controls consist of 4 rpm commands. Our model M is given by the quadrotor dynamics (Mahony et al., 2012; Sridhar and Sukumar, 2019). For emphasis on hover, we uniformly sample states across the state distribution and uniformly sample controls for balancing gravity (and hence hovering in-place) to create the Ω dataset. We have 15,000 training points, 2000 test points in each of $\mathcal{D}$ and Ω. We vary the number of memories from 800, 1000, to 2000. Similar to CARLA, we see (in Figure 5) that approximation loss on $\mathcal{D}$ is similar across all methods but there is *upto a 6 order-of-magnitude decrease* in the average and maximum constraint loss on Ω with our constrained training algorithm. We also observe a rather large increase in performance from 1000 to 2000 memories. We also plot the average constraint loss on $\mathcal{D}$ for all case studies in Appendix 10.4.

10. Conclusion

We demonstrate how DNN training can be constrained using symbolic information which enforces adherence to natural laws. We report experiments on three case studies where our method achieves many-fold reductions in constraint loss when compared to the augmented Lagrangian. In future work, we plan to create safety-constrained neurosymbolic policies.

Acknowledgements This work was supported in part by ARO W911NF-20-1-0080 and AFRL and DARPA FA8750-18-C-0090. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Air Force Research Laboratory (AFRL), the Army Research Office (ARO), the Defense Advanced Research Projects Agency (DARPA), the Department of Defense, or the United States Government. Additionally, we would like to thank Prof Eric Eaton from the University of Pennsylvania for valuable discussions on a closely related idea.

References

Witold Brostow, Jean-Pierre Dussault, and Bennett L Fox. Construction of voronoi polyhedra. *Journal of Computational Physics*, 29(1):81–92, 1978. ISSN 0021-9991. doi: https://doi.org/10.1016/0021-9991(78)90110-9. URL https://www.sciencedirect.com/science/article/pii/0021999178901109.

Sanjian Chen, James Weimer, Michael R Rickels, Amy Peleckis, and Insup Lee. Towards a model-based meal detector for type i diabetics. 2015.

Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2019.

Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. *arXiv preprint arXiv:2003.04630*, 2020.

Alp Dener, Marco Andres Miller, Randy Michael Churchill, Todd Munson, and Choong-Seock Chang. Training neural networks under physical constraints using a stochastic augmented lagrangian approach. *arXiv preprint arXiv:2009.07330*, 2020.

Franck Djeumou, Cyrus Neary, Eric Goubault, Sylvie Putot, and Ufuk Topcu. Neural networks with physics-informed architectures and constraints for dynamical systems modeling. In *Learning for Dynamics and Control Conference*, pages 263–277. PMLR, 2022.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.

Thai Duong and Nikolay Atanasov. Hamiltonian-based neural ode networks on the se (3) manifold for dynamics learning and control. *arXiv preprint arXiv:2106.12782*, 2021.

Marc Finzi, Ke Alexander Wang, and Andrew G Wilson. Simplifying hamiltonian and lagrangian neural networks via explicit constraints. *Advances in neural information processing systems*, 33: 13880–13889, 2020.

Ferdinando Fioretto, Pascal Van Hentenryck, Terrence WK Mak, Cuong Tran, Federico Baldo, and Michele Lombardi. Lagrangian duality for constrained deep learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 118–135. Springer, 2020.

Bernd Fritzke. A growing neural gas network learns topologies. In *Proceedings of the 7th International Conference on Neural Information Processing Systems*, NIPS'94, page 625–632, Cambridge, MA, USA, 1994. MIT Press.

Brian Goldfain, Paul Drews, Changxi You, Matthew Barulic, Orlin Velev, Panagiotis Tsiotras, and James M. Rehg. Autorally: An open platform for aggressive autonomous driving. *IEEE Control Systems Magazine*, 39(1):26–55, 2019. doi: 10.1109/MCS.2018.2876958.

Eric Goubault and Sylvie Putot. Rino: Robust inner and outer approximated reachability of neural networks controlled systems. In *Computer Aided Verification: 34th International Conference, CAV 2022, Haifa, Israel, August 7–10, 2022, Proceedings, Part I*, page 511–523, Berlin, Heidelberg, 2022. Springer-Verlag. ISBN 978-3-031-13184-4. doi: 10.1007/978-3-031-13185-1_25. URL https://doi.org/10.1007/978-3-031-13185-1_25.

Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. *Advances in neural information processing systems*, 32, 2019.

Jayesh K Gupta, Kunal Menda, Zachary Manchester, and Mykel Kochenderfer. Structured mechanical models for robot learning and control. In *Learning for Dynamics and Control*, pages 328–337. PMLR, 2020.

Ramneet Kaur, Kaustubh Sridhar, Sangdon Park, Susmit Jha, Anirban Roy, Oleg Sokolsky, and Insup Lee. Codit: Conformal out-of-distribution detection in time-series data. *arXiv preprint arXiv:2207.11769*, 2022.

Hoel Kervadec, Jose Dolz, Jing Yuan, Christian Desrosiers, Eric Granger, and Ismail Ben Ayed. Constrained deep networks: Lagrangian optimization via log-barrier extensions. In *2022 30th European Signal Processing Conference (EUSIPCO)*, pages 962–966. IEEE, 2022.

Taisa Kushner, Sriram Sankaranarayanan, and Marc Breton. Conformance verification for neural network models of glucose-insulin dynamics. In *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, pages 1–12, 2020.

Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021a.

Lu Lu, Raphael Pestourie, Wenjie Yao, Zhicheng Wang, Francesc Verdugo, and Steven G Johnson. Physics-informed neural networks with hard constraints for inverse design. *SIAM Journal on Scientific Computing*, 43(6):B1105–B1132, 2021b.

Michael Lutter, Christian Ritter, and Jan Peters. Deep lagrangian networks: Using physics as model prior for deep learning. *arXiv preprint arXiv:1907.04490*, 2019.

Robert Mahony, Vijay Kumar, and Peter Corke. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE Robotics and Automation magazine*, 19(3):20–32, 2012.

Chiara Dalla Man, Francesco Micheletto, Dayu Lv, Marc Breton, Boris Kovatchev, and Claudio Cobelli. The uva/padova type 1 diabetes simulator: new features. *Journal of diabetes science and technology*, 8(1):26–34, 2014.

Pablo Márquez-Neila, Mathieu Salzmann, and Pascal Fua. Imposing hard constraints on deep networks: Promises and limitations. *arXiv preprint arXiv:1706.02025*, 2017.

T.M. Martinetz, S.G. Berkovich, and K.J. Schulten. 'neural-gas' network for vector quantization and its application to time-series prediction. *IEEE Transactions on Neural Networks*, 4(4):558–569, 1993. doi: 10.1109/72.238311.

Takashi Matsubara, Ai Ishikawa, and Takaharu Yaguchi. Deep energy-based modeling of discrete-time physics. *Advances in Neural Information Processing Systems*, 33:13100–13111, 2020.

Regina Hunter Mladineo. Stochastic minimization of lipschitz functions. 1991.

Yatin Nandwani, Abhishek Pathak, and Parag Singla. A primal dual formulation for deep learning with constraints. *Advances in Neural Information Processing Systems*, 32, 2019.

Monal Narasimhamurthy, Taisa Kushner, Souradeep Dutta, and Sriram Sankaranarayanan. Verifying conformance of neural network models: Invited paper. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8, 2019. doi: 10.1109/ICCAD45719.2019.8942151.

Jacopo Panerati, Hehui Zheng, SiQi Zhou, James Xu, Amanda Prorok, and Angela P. Schoellig. Learning to fly—a gym environment with pybullet physics for reinforcement learning of multi-agent quadcopter control. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

Y. Prudent and A. Ennaji. An incremental growing neural gas learns topologies. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 1211–1216 vol. 2, 2005. doi: 10.1109/IJCNN.2005.1556026.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Rajesh Rajamani. *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

Sathya N Ravi, Tuan Dinh, Vishnu Suresh Lokhande, and Vikas Singh. Explicitly imposing constraints in deep networks via conditional gradients gives improved generalization and faster convergence. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4772–4779, 2019.

Manuel A Roehrl, Thomas A Runkler, Veronika Brandtstetter, Michel Tokic, and Stefan Obermayer. Modeling system dynamics with physics-informed neural networks based on lagrangian mechanics. *IFAC-PapersOnLine*, 53(2):9195–9200, 2020.

Walter Rudin. *Principles of mathematical analysis*. McGraw-Hill Book Company, Inc., New York-Toronto-London, 1953.

Guanya Shi, Xichen Shi, Michael O'Connell, Rose Yu, Kamyar Azizzadenesheli, Animashree Anandkumar, Yisong Yue, and Soon-Jo Chung. Neural lander: Stable drone landing control using learned dynamics. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9784–9790. IEEE, 2019.

Kaustubh Sridhar and Srikant Sukumar. Finite-time, event-triggered tracking control of quadrotors. In *5th CEAS Specialist Conference on Guidance, Navigation & Control (EurGNC 19) Milano, Italy*, 2019.

Kaustubh Sridhar, Radoslav Ivanov, Vuk Lesi, Marcio Juliato, Manoj Sastry, Lily Yang, James Weimer, Oleg Sokolsky, and Insup Lee. A framework for checkpointing and recovery of hierarchical cyber-physical systems. *arXiv preprint arXiv:2205.08650*, 2022.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Symplectic ode-net: Learning hamiltonian dynamics with control. *arXiv preprint arXiv:1909.12077*, 2019.

Appendix

10.1. Proof of Lemma 4

Lemma 7 *Let $g : \mathbb{R}^t \to \mathbb{R}$ be an L_g-Lipschitz continuous function on a closed and compact set S_a, and l and u be its estimated lower and upper bounds. Then, $\forall z \in [l, u]$, $\max_{s \in \mathcal{S}_a} |g(s) - z| < L_g |S_a|$.*

Proof : Note that g is a real and continuous function on the connected set $\mathcal{S}_a$ in the metric space $\mathbb{R}^t$. Since, there exists points s_l and s_u which map to l and u respectively, then by Theorem 4.22 Rudin (1953), for any $z \in [l, u]$ there exists $s_z \in \mathcal{S}_a$ such that $z = g(s_z)$. Then we can write the following : $\max_{s \in S_a} |g(s) - g(s_z)| \leq L_g |s - s_z| \leq L_g |S_a|$. This completes the proof.

10.2. Detailed Algorithm for Training a Constrained Neural Network Dynamics Model

Algorithm 2 `Training a Constrained Neural Network Dynamics Model`

Input: Dataset $\mathcal{D} = \{(s, x)_i\}_{i \in [N_\mathcal{D}]}$, Knowledge M, DNN architecture $f_\theta(.)$
Output: Constrained neural network dynamics model $\Gamma(f_\theta)(.)$
Parameters: Number of memories n_{memories}, batch sizes $N_{\mathcal{D}_{\text{batch}}}$, $N_{\Omega_{\text{batch}}}$, $0 \leq \gamma < 1$, N_Steps, update_freq

1: Generate input samples $\Omega = \{(s')_i\}_{i \in [N_\Omega]}$ // unlabelled dataset
2: $D|_{inputs} = \{(s)_i\}_{i \in [N_\mathcal{D}]} \cup \{(s')_i\}_{i \in [N_\Omega]}$ // combine inputs in both datasets
3: Memories $\mathcal{A}$, edges $\mathcal{E} \leftarrow$ NeuralGas($D|_{inputs}$, n_{memories}) // topology of input space
4: $\mathcal{S}^1, ..., \mathcal{S}^j, ... \leftarrow$ VoronoiCells($\mathcal{A}, \mathcal{E}$) // partitions in input space
5: **for** each voronoi cell S^j **do**
6: Sample points inside the cell, propagate through model M, and compute lower and upper bounds $I^j_{low} = \min_{s \sim \mathcal{S}^j} M(s)$ and $I^j_{up} = \max_{s \sim \mathcal{S}_j} M(s)$
7: **end for**
8: **for** s in D, Ω **do**
9: $S_j \leftarrow$ FindVoronoiCell(s, $\mathcal{A}$)
10: Set $\texttt{Lo}(s), \texttt{Up}(s) \leftarrow I^j_{low}, I^j_{up}$ // output bounds for datasets
11: **end for**
12: **for** step in N_Steps **do**
13: Sample batches $\mathcal{D}_{\text{batch}} = \text{Sample}(D, N_{\mathcal{D}_{\text{batch}}})$, $\Omega_{\text{batch}} = \text{Sample}(D, N_{\Omega_{\text{batch}}})$
14: Set $\texttt{Lo}(s,\ \text{step}) = \texttt{Lo}(s) - \gamma^{\text{step}}\ (\texttt{Up}(s) - \texttt{Lo}(s))$ and $\texttt{Up}(s,\ \text{step}) = \texttt{Lo}(s) + \gamma^{\text{step}}\ (\texttt{Up}(s) - \texttt{Lo}(s))$
15: Compute $\Gamma(f_\theta)(.)$ for $\mathcal{D}_{\text{batch}}$ and Ω_{batch} using $\texttt{Lo}(s,\ \text{step})$ and $\texttt{Up}(s,\ \text{step})$ // constrained DNN (5)
16: Compute $Loss(\theta, \lambda, \mu)$ // augmented Lagrangian loss (6) or vanilla approximation loss
17: $\theta \leftarrow Optimization\_Step(Loss, \theta, \mathcal{D}_{\text{batch}}, \Omega_{\text{batch}})$
18: **if** step % update_freq == 0 **then**
19: $\lambda_1, \lambda_2, \mu_1, \mu_2 \leftarrow Update\_Step(\psi^\delta_{M, \Gamma(f_\theta)}, \lambda, \mu)$
20: **end if**
21: **end for**
22: **return** $\Gamma(f_\theta)(.)$

We detail our Algorithm in this section. The inputs to the algorithm are a state transitions dataset D containing ((state, control), (next state)) pairs (s, x), model M, and architecture $f_\theta : \mathcal{S} \to \mathcal{X}$.

Algorithm 2 First (line 1), we generate the unlabelled Ω dataset which consists of only inputs to the model $s' \in \mathcal{S}$ by sampling throughout the input space but with particular emphasis on relevant regions in $\mathcal{S}$. Then, (lines $2 - 4$), we use the unsupervised neural gas algorithm (Martinetz et al.,

1993; Fritzke, 1994) to obtain the neural gas graph $\mathcal{G} = (\mathcal{A}, \mathcal{E})$. We utilize these memories and edges, to create partitions of the input space as voronoi cells with memories at their center. In each voronoi cell, we sample points, propagate them through the model M and obtain the upper and lower limits along each dimension of the output space $\mathcal{X}$ (lines 5-7). This creates the constraint map C. Using this, we can find the lower and upper bounds of each point in $\mathcal{D}$ and Ω (lines 8-12). First, we locate the corresponding voronoi cell, and then use the bounds computed in Line 6. Finally, we can train the constrained neural network (denoted $\Gamma(f_\theta)(.)$) as follows,

$$\Gamma(f_\theta)(s) = \texttt{Lo}(s) + \sigma\left(f_\theta(s)\right) \left(\texttt{Up}(s) - \texttt{Lo}(s)\right) \tag{5}$$

where $f_\theta : \mathcal{S} \to \mathcal{X}$ is a parameterized function which maps from the input space to output space, and $\sigma(x) : \mathcal{X} \to [0, 1]$ is the sigmoid function. Equation 5 is but one realization of the constraining operator discussed in Definition 5. Our loss function is the augmented Lagrangian loss (Lu et al., 2021b) itself and is given below $\big($where $\psi^{\delta}_{M,\Gamma(f_\theta)}(s) = \delta - ||M(s) - \Gamma(f_\theta)(s)||\big)$.

$$\begin{aligned} Loss(\theta, \lambda_1, \mu_2, \lambda_2, \mu_2) = \mathop{\mathbb{E}}_{\substack{s\sim\mathcal{D} \\ s'\sim\Omega}} \Bigg[L\big(\Gamma(f_\theta)(s), x\big) + \Bigg(\lambda_1 \psi^{\delta}_{M,\Gamma(f_\theta)}(s) + \lambda_2 \psi^{\delta}_{M,\Gamma(f_\theta)}(s') \\ + \mu_1 \, \mathbb{1}_{(\lambda_1>0 \vee \psi>0)} \left(\psi^{\delta}_{M,\Gamma(f_\theta)}(s)\right)^2 + \mu_2 \, \mathbb{1}_{(\lambda_2>0 \vee \psi>0)} \left(\psi^{\delta}_{M,\Gamma(f_\theta)}(s')\right)^2 \Bigg) \Bigg] \end{aligned} \tag{6}$$

We can then train the neural network by back-propagating through the constrained neural network (lines 12-16). We enhance gradient feedback under constrained outputs with an exponential schedule on the lower and upper bounds (line 13). We also intermittently update the slack variables through a schedule or as a gradient ascent step on the value of the constraint $\psi^{\delta}_{M,\Gamma(f_\theta)}$ (lines 17-19).

10.3. Additional Details of Experiments

For the CARLA vehicle and PyBullet Drones models, we use a two layer MLP with 1024 neurons in each layer. In the UVA/Padova Artificial Pancreas case study, we use a three layer neural network with 20 neurons in each layer. We utilize the Adam optimizer in all case studies and choose a learning rate with grid search in $[0.001, 0.1]$. We also utilize training batch sizes of 64 for both $\mathcal{D}$ and Ω datasets. Further, for the CARLA and Drones case studies, we set γ to 0. For Artificial Pancreas, we used $\gamma = 0.99$.

10.4. Additional Plots

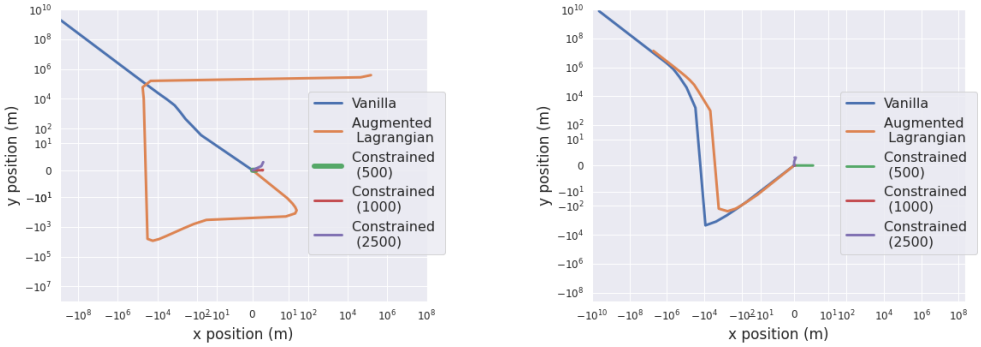

Figure 6: Analysis of CARLA model prediction drift starting from origin at rest when given zero control inputs for 20 timesteps for random seed of 0 *(left)* and random seed of 1 *(right)*.

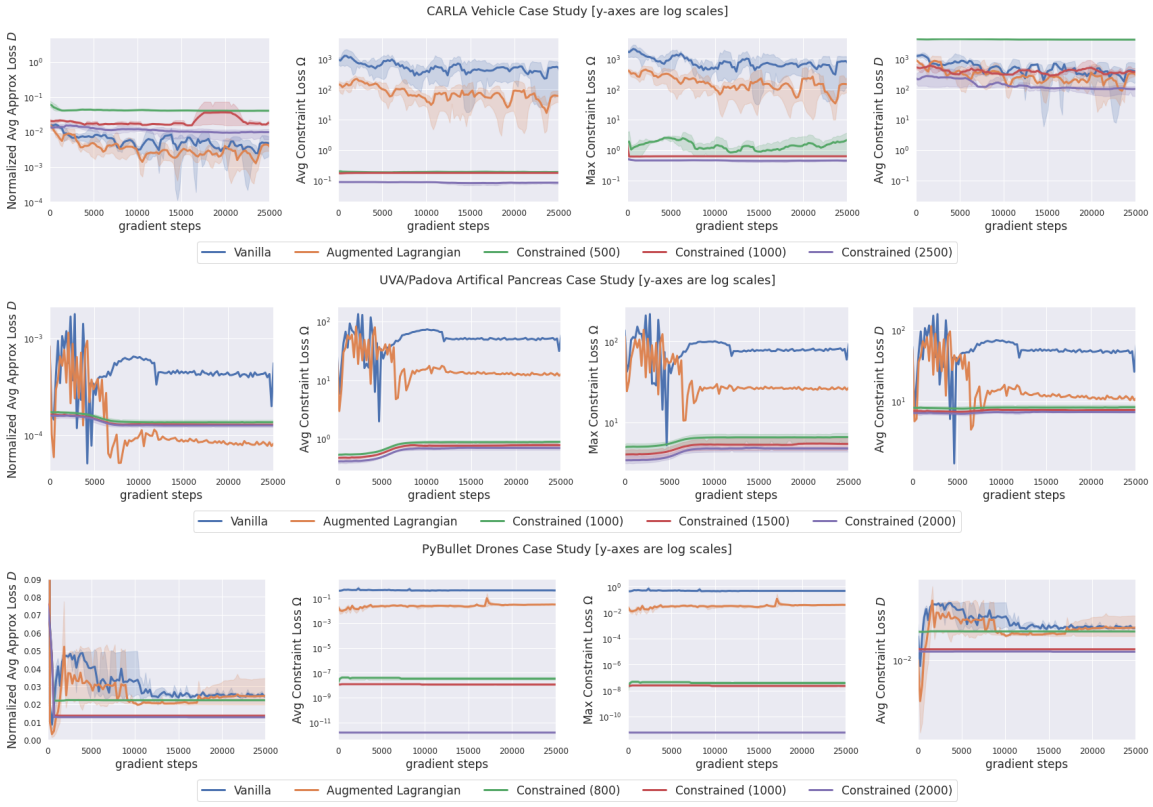

Figure 7: Plots of approximation loss on $\mathcal{D}$, average constraint loss on Ω, maximum constraint loss on Ω, and average constraint loss on $\mathcal{D}$ (for 3 random seeds) against gradient steps for the CARLA Vehicle *(top row)*, Artifical Pancreas *(second row)*, and PyBullet Drones *(third row)* case studies.