NEURAL DYNAMICAL SYSTEMS

Viraj Mehta, Ian Char, Willie Neiswanger, Youngseog Chung & Jeff Schneider
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{virajm,ichar,willie,youngsec,schneide}@cs.cmu.edu

Andrew Oakleigh Nelson, Mark D Boyer & Egemen Kolemen
Princeton Plasma Physics Laboratory
Princeton, NJ 08540, USA
{anelson,mboyer,ekolemen}@pppl.gov

1 INTRODUCTION

We introduce Neural Dynamical Systems (NDS), a method of learning dynamical models which incorporates prior knowledge in the form of systems of ordinary differential equations. NDS uses neural models to estimate free parameters of the system, predicts residual terms, and numerically integrates over time to predict future states. It also natively handles irregularly sampled data and implicitly learns values of interpretable system parameters. We find that NDS learns dynamics with higher accuracy and fewer samples than a variety of deep learning methods that do not incorporate the prior knowledge. We demonstrate these advantages first on synthetic dynamical systems and then on real data captured from deuterium shots from a nuclear fusion reactor.

2 METHODS

We aim to introduce this model as a choice trading off between fidelity to our prior knowledge about the world and flexibility to allow the model to adjust for the inevitable differences between our modeling assumptions and reality.

We define a *Neural Dynamical System* (NDS) as a class of dynamical systems $\dot{x} = f_\phi(x, u, t)$ where a neural network is some part of $f_\phi(x, u, t)$, predicting the parameters ϕ or some other component of the system.

The following methods will discuss how to include ODE-structured prior knowledge in Neural Dynamical Systems: first in the ideal situation, where we know the correct system dynamics up to the parameters of the dynamical system, as given by ϕ in our definition above. As this is a highly ideal situation, we then point out two ways of relaxing to more incomplete information.

In all of these cases, our method has several advantages:

- Data Efficiency: by including prior knowledge we can learn accurate predictors with smaller amounts of data.
- Accuracy: in situations where the dynamics are difficult to learn, an approximate model can help predictions start from a baseline of accuracy that could be otherwise hard to achieve and may help the final model reach a higher end level of performance.
- Continuous time: Neural ODE models natively operate in continuous time. Data arriving at irregular intervals can be natively handled by Neural Dynamical Systems with good performance.
- Explainability: to the extent that the parameters ϕ are meaningful values like the Rayleigh constant in the Lorenz system and that our system predicts accurate values for them, we can interpret the predictions of the system through these parameters.

NDS with Full System Dynamics Consider a class of dynamical systems where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, $\phi \in \mathbb{R}^{d_p}$, $d_h, d_c \in \mathbb{X}$, and let θ, ϑ, τ be trainable neural network weights. Let $T < T'$ $h_\theta(x_{t_{1:T'}}, u_{t_{1:T}}) : \mathcal{X}^{T'} \times \mathcal{U}^T \to \mathbb{R}^{d_h + d_p}$ be a fully connected neural network which we call a

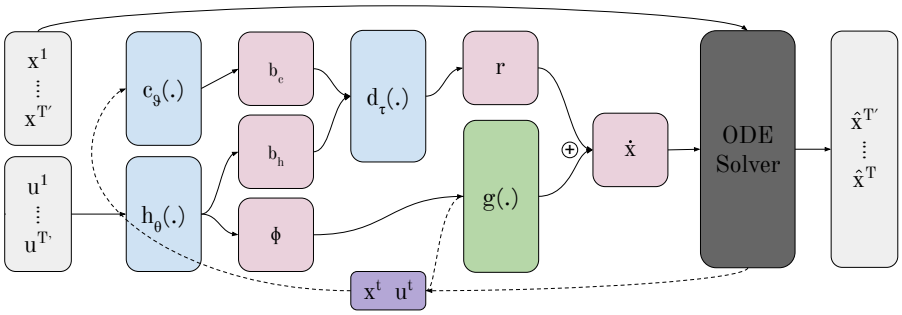

Figure 1: **A schematic Neural Dynamical System.** Blue boxes are fully connected neural networks. Grey boxes are problem data and output. Pink boxes are embeddings and intermediate quantities. The green box is the prior knowledge dynamical system. The purple box is data output by ODE solver to query derivatives. Naturally, the ODE solver is a black box.

'history encoder' that outputs the parameters of the system $\hat{\phi}$ and an embedding $b_h \in \mathbb{R}^{d_h}$. Also let $c_\vartheta(x_t, u_t) : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^{d_c}$ be a similar 'context encoder' for a single state and control that outputs an embedding $b_c \in \mathbb{R}^{d_c}$. Finally, let $d_\tau(b_h, b_c) : \mathbb{R}^{d_c+d_h} \to \mathbb{R}^n$ be a fully connected neural network, which we call a 'fusion encoder' that outputs residual terms $\hat{r}$. Then we can set up the following dynamical system, which can be seen in Figure 1:

$$\begin{aligned} \hat{\phi}, b_h = h_\theta(x_{t_{1:T'}}, u_{t_{1:T}}) \quad & b_c = c_\vartheta(x_t, u_t) \\ \hat{r} = d_\tau(b_h, b_c) \quad & \dot{x} = g_{\hat{\phi}}(x_t, u_t, t) + \hat{r} \end{aligned} \tag{1}$$

where g are domain-specific ODEs which are the input 'domain knowledge' about the system being modeled. These can be equal to the true equations f and in the 'full dynamics' setting we assume they are. But as we remove structure from the model, we will first remove equations from the system g and then remove the assumption that they correctly model the dynamics. In order to more easily explain how we evaluate and supervise these systems, we give an example.

Example 1: Lorenz system. To illustrate the full construction, we operate on the example of the the Lorenz system: a chaotic dynamical system originally defined to model atmospheric processes (Lorenz, 1963). The system has 3-dimensional state (which we'll denote by x, y, z), 3 parameters, ρ, σ, and β, and no control input. The system is given by

$$\dot{x} = \sigma(y - x) \quad \dot{y} = x(\rho - z) - y \quad \dot{z} = xy - \beta z. \tag{2}$$

For a given instantiation of the Lorenz system, we have values of $\phi = [\beta, \sigma, \rho]$ that are constant across the trajectory. So, we can instantiate a history encoder h_θ which outputs $\hat{\phi} = [\hat{\beta}, \hat{\sigma}, \hat{\rho}]$. We use the DOPRI5 method (Dormand & Prince, 1980) to integrate the full neural dynamical system in Equation 1, with g given by the system in Equation 2 using the adjoint method of Chen et al. (2018). We use the state $x_{T'}$ as the initial condition for this integration. This gives a series $\{\hat{x}_t\}_{t=T'}^{T}$, which we evaluate and supervise with a loss of the form

$$\mathcal{L}_{\theta,\vartheta,\tau}(\{\hat{x}_{t_i}\}_{i=T'+1}^{T}, \{x_{t_i}\}_{t=T'+1}^{T}) = \sum_{t=T'+1}^{T} ||x_{t_i} - \hat{x}_{t_i}||_2^2. \tag{3}$$

NDS with Partial System Dynamics Suppose we only had prior knowledge about some of the components of our system and none about others. We can easily accomodate this incomplete information by simply 'zeroing out' the function g for the components we don't know, i.e. $g_i(x, u, t) = 0$ for an unknown ith component.

NDS with Approximate System Dynamics For Neural Dynamical Systems to be useful, they must handle situations where the known model is approximate. This is transparently handled by our formulation of Neural Dynamical Systems: the parameters of the approximate model $\hat{\phi}$ are predicted by a 'history encoder' and the residuals $\hat{r}$ are predicted by a 'fusion encoder'. This is the same as in the case where we have the correct dynamics.

Example 2: Nuclear Fusion System. In this paper, we apply this technique to plasma dynamics in a tokamak. In a tokamak, two quantities of interest are the stored energy of the plasma, which we denote E and its rotational frequency, ω. The simple model used for control development in Boyer et al. (2019) is used in this work.

$$\dot{E} = P - \frac{E}{\tau_e} \qquad \dot{\omega} = \frac{T}{n_i m_i R_0} - \frac{\omega}{\tau_m} \tag{4}$$

Here, n_i is ion density (which we approximate as a constant value of 5×10^{19} deuterium ions per m^3), m_i is ion mass (which we know since our dataset contains deuterium shots and the mass of a deuterium ion is 3.3436×10^{-27} kg), and R_0 is the tokamak major radius of 1.67 m. We use the constant known values for these. τ_e and τ_m are the confinement times of the plasma energy and momentum, which we treat as variable parameters (because they are!). These are predicted by the neural network in our model.

3 EXPERIMENTS

We aim to show with the following experiments that our methods improve predictions of physical systems by including prior dynamical knowledge about the systems, even as we vary the configurations between the most structured and more fluid settings. We show that our models learn from less data and are more accurate. We first present results on a pair of synthetic physical systems where the data is generated in a noiseless and regularly spaced setting.

Afterwards, we show NDS performance on data taken from several plasma shots on tokamak reactors. This setting is extremely challenging, as the physics community is still actively trying to understand plasma and tokamak dynamics. That being said, we still see a substantial improvement in prediction even though we use a simplified model of high-level summary measures.

We use $L2$ error as our evaluation measure for predictive accuracy as given by Equation 3. For synthetic examples, we learn over trajectories $\{(x_{t_i}, u_{t_i}, t_i)\}_{i=1}^{T}$ where the x_{t_i} are generated by numerically integrating $\dot{x}_\phi(x, u, t)$ using scipy's odeint function (Virtanen et al., 2019), with x_0 and ϕ uniformly sampled from $\mathcal{X}$ and Φ, and u_{t_i} given. We evaluate the synthetic experiments on a test set of 500 trajectories that is fixed for a particular random seed generated in the same way as the training data. We use a timestep of 0.5 seconds for the synthetic trajectories, which allows us to see trajectories that don't reach their peak and those that start to fall for the Ballistic system and to get a wide spread of data on the Lorenz system (note the Lyapunov exponent of the system is less than 3 so in 16 predicted timesteps we get both predictable and unpredictable data (Frøyland & Alfsen, 1984)). We believe it is important to look at the progress of the system across this threshold to understand whether the NDS model is robust to chaotic dynamics — since the Lorenz system used for structure is itself chaotic, we want to make sure that the system doesn't blow up over reasonably long timescales. Lastly, the fusion data was measured at intervals of $0.05s$.

All of these models take 32 timesteps of state and control information as input and are trained on predictions for the following 16 timesteps. The ODE-based models are integrated from the initial conditions of the last given state, while the fully connected and LSTM models naturally handle the input and output data. The models are all trained with a learning rate of 3×10^{-3}, which was seen to work well across models. The learning rate was also reduced by a factor of 10 whenever test error plateaued for 10 evaluations.

3.1 SYNTHETIC EXPERIMENTS

Sample Complexity and Overall Accuracy As we are interested in settings with limited data, we trained the dynamical systems over $4,000$ batches of data. We generated training data on the fly for synthetic experiments, so the model only ever saw each training example once. We repeated this process with 5 different random seeds and plotted the $L2$ error of the model over the number of samples seen by the model in Figure 2. The error regions are the standard deviation of the errors over the various seeds.

As seen in Figure 2, the learning of Neural Dynamical Systems looks very different to that of the comparison models. NDS improves rapidly in comparison to the fully connected models in both the Lorenz and Ballistic cases. The NDS models with and without structure rapidly achieve good accuracy and then capture mostly incremental gains afterward. We see the NDS with partial structure

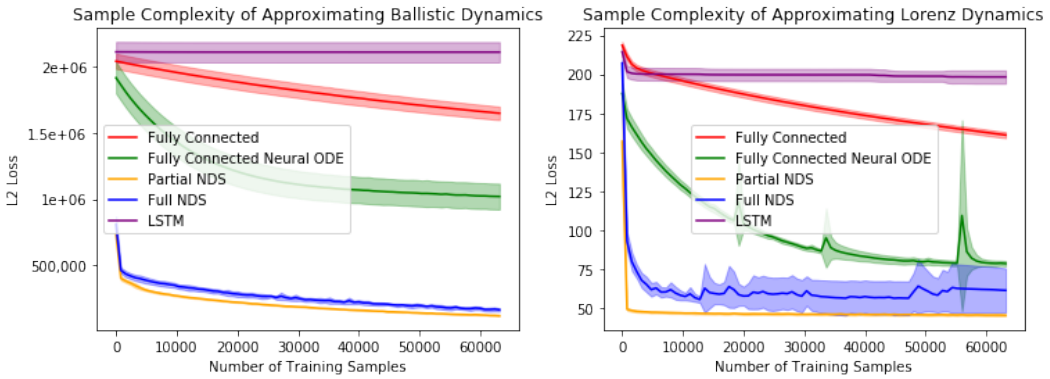

Figure 2: **L2 loss between predicted and real trajectory as we train on more samples.** The NDS models learn much more quickly and converge to much lower errors on the Ballistic and Lorenz systems.

in both cases perform slightly better than the full system. This is surprising to us as the partial systems have less information than the full ones.

A potential explanation for this is that errors propagate through the dynamical model when the parameters are wrong, while the partial systems naturally dampen errors since, for example, $\dot{z}$ only depends on the other components through a neural network. Concretely this might look like a full NDS predicting the wrong Rayleigh number σ which might give errors to y which would then propagate to x and y. Conversely, this wouldn't happen as easily in a partial NDS because there are neural networks intermediating the components of the system. This certainly bears further exploration.

The Fully Connected Neural ODE outperforms the other models besides NDS, which we posit is due to the fact that it implicitly represents that this system is a continuous time dynamical process and should change in a continuous fashion. We also conducted experiments where we added noise, looked at the accuracy of the learned parameters ϕ, and irregularly sampled data in time.

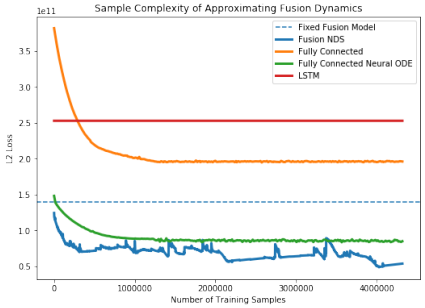

Figure 3: **L2 loss between predicted and real trajectories on the fusion problem** as we train on more samples. The Fusion NDS outperforms other models, including a fixed version of the prior knowledge model and a fully connected Neural ODE. The initial drop in loss is steep, as is characteristic of NDS models.

3.2 Fusion Experiments

We explored the concept of approximate system dynamics in a fusion system. The problem is a low-dimensional one: predicting the state of the tokamak as summarized by its stored energy and rotational frequency given the time series of control input in the form of injected power and torque. As we mentioned in Section 2, we have a simplified physical model given by Equation 4 that approximately gives the dynamics of these quantities and how they relate to one another through time.

Our full dataset consisted of 17,686 shots, which we randomly partitioned into 1000 as a test set and 16,686 as a training set. Data is loaded and partially processed within the OMFIT framework (Meneghini et al., 2015). We compare with the same models as in the previous section, but our Fusion Neural Dynamical System is as described in Equation 1 with g given by Equation 4. As we discussed above, the dynamics in this equation are approximate. To illustrate this, we have included the accuracy of the naive dynamics with no learning on our data with fixed confinement times $\tau_e = \tau_m = 0.1$s as the Fixed Fusion Model in Figure 3. We use a larger Fully Connected network with 6 layers with 512 hidden nodes to attempt to capture the added complexity of the problem.

Sample Complexity and Overall Accuracy When comparing the three points on the spectrum of added structure (i.e. fusion NDS, fully-connected neural ODE, and fully connected network), we

see that the Fusion NDS model performs best. Although the fully connected neural ODE performs competitively, it fails to reach the same performance. We speculate that the dynamical model helps with generalization whereas the fully connected network may overfit the training data and fail to reach good performance on the test set.

We also see the steep initial decrease in loss for NDS which is similar to that in the synthetic experiments. If we were more sample-constrained or wanted to learn over a specific subset of the data, it seems that these NDS models learn something useful even in the initial batches due to their system-identification-like properties.

REFERENCES

M. D. Boyer, K. G. Erickson, B. A. Grierson, D. C. Pace, J. T. Scoville, J. Rauch, B. J. Crowley, J. R. Ferron, S. R. Haskey, D. A. Humphreys, R. Johnson, R. Nazikian, and C. Pawley. Feedback control of stored energy and rotation with variable beam energy and perveance on DIII-D. *Nuclear Fusion*, 59(7):076004, May 2019. ISSN 0029-5515. doi: 10.1088/1741-4326/ab17f5. URL https://doi.org/10.1088%2F1741-4326%2Fab17f5.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural Ordinary Differential Equations. *arXiv:1806.07366 [cs, stat]*, June 2018. URL http://arxiv.org/abs/1806.07366. arXiv: 1806.07366.

J. R. Dormand and P. J. Prince. A family of embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26, March 1980. ISSN 0377-0427. doi: 10.1016/0771-050X(80)90013-3. URL http://www.sciencedirect.com/science/article/pii/0771050X80900133.

Jan Frøyland and Knut H. Alfsen. Lyapunov-exponent spectra for the Lorenz model. *Physical Review A*, 29(5):2928–2931, May 1984. doi: 10.1103/PhysRevA.29.2928. URL https://link.aps.org/doi/10.1103/PhysRevA.29.2928.

Edward N. Lorenz. Deterministic Nonperiodic Flow. *Journal of the Atmospheric Sciences*, 20 (2):130–141, March 1963. ISSN 0022-4928. doi: 10.1175/1520-0469(1963)020⟨0130:DNF⟩2.0.CO;2. URL https://journals.ametsoc.org/doi/abs/10.1175/1520-0469%281963%29020%3C0130%3ADNF%3E2.0.CO%3B2.

O. Meneghini, S. P. Smith, L. L. Lao, O. Izacard, Q. Ren, J. M. Park, J. Candy, Z. Wang, C. J. Luna, V. A. Izzo, B. A. Grierson, P. B. Snyder, C. Holland, J. Penna, G. Lu, P. Raum, A. McCubbin, D. M. Orlov, E. A. Belli, N. M. Ferraro, R. Prater, T. H. Osborne, A. D. Turnbull, and G. M. Staebler. Integrated modeling applications for tokamak experiments with OMFIT. *Nuclear Fusion*, 55(8):083008, July 2015. ISSN 0029-5515. doi: 10.1088/0029-5515/55/8/083008. URL https://doi.org/10.1088%2F0029-5515%2F55%2F8%2F083008.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C. J. Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1 0 Contributors. SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python. *arXiv:1907.10121 [physics]*, July 2019. URL http://arxiv.org/abs/1907.10121. arXiv: 1907.10121.