A comparative study between different neural networks for dynamical systems

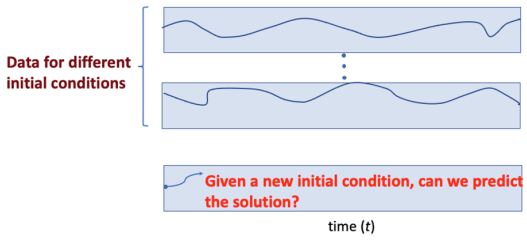
Nikhil Kadivar (nkadivar), Alan John Varghese (ajohnvar)

Introduction

Dynamical systems are seen in several fields ranging from climate modeling to biological systems.Dynamical systems are systems whose state evolves with time according to a given differential relation.

$$\frac{d}{dt}\boldsymbol{x}(t) = \boldsymbol{f}\left(\boldsymbol{x}(t)\right)$$

Nonlinear dynamical systems are widely seen in various systems, ranging from climate to biological systems .In several applications, we only have access to the observed time series, and we lack knowledge of the exact governing equations. For practical applications, we are often interested in forecasting the dynamics of these systems. We aim to leverage deep learning models to solve the above mentioned problem.



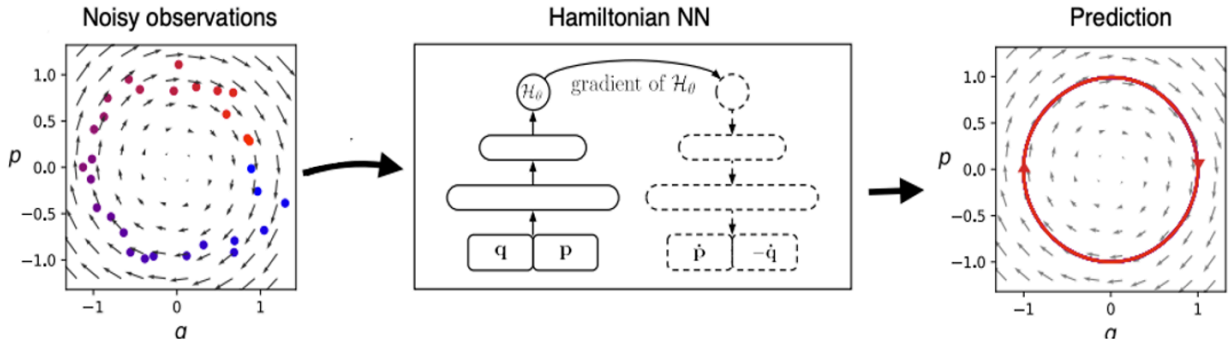
Methodology

One interesting work in this line is the Hamiltonian Neural Network [1], which is applicable to systems where energy is conserved. We apply Hamiltonian Neural Networks to the following dynamical system: i) Ideal spring mass system, ii) the three-body problem in mechanics and iii) Lorenz system (chaotic nonlinear dynamical system applicable to weather data)

In our project, we compare the performance of HNN, LSTM, multistep neural networks on all these three problems.

The training data comprises the trajectories of the dynamical system for various random initial conditions. During testing, we feed in the initial condition of the dynamical system into the neural network model (coupled with a numerical solver), and it predicts the evolution of the trajectory.

Hamiltonian Neural Networks



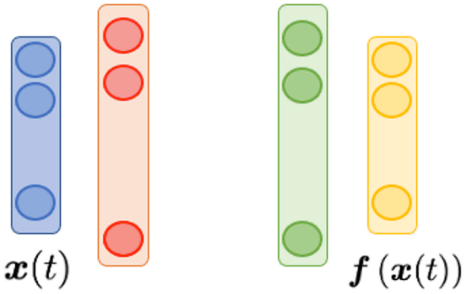
In Hamiltonian Neural Networks, the neural network learns the Hamiltonian of the dynamical system directly from the data. The derivative of the Hamiltonian with respect to the input variables are related to the time derivative of the input variables according to the following equation:

$$\frac{d\mathbf{q}}{dt} = \frac{\partial \mathcal{H}}{\partial \mathbf{p}}, \quad \frac{d\mathbf{p}}{dt} = -\frac{\partial \mathcal{H}}{\partial \mathbf{q}} \qquad \mathcal{L}_{HNN} = \left\| \frac{\partial \mathcal{H}_\theta}{\partial \mathbf{p}} - \frac{\partial \mathbf{q}}{\partial t} \right\|_2 + \left\| \frac{\partial \mathcal{H}_\theta}{\partial \mathbf{q}} + \frac{\partial \mathbf{p}}{\partial t} \right\|_2$$

During testing, the Hamiltonian Neural Network is coupled with a numericals solver. We feed in a new initial condition, and our model predicts the solution for this unseen initial condition.

Multistep Neural networks

In Multistep Neural Networks, we aim to learn the unknown function *f* directly from the data. Here, the function *f* is approximated using a feed forward neural network. The loss function is designed by using the multistep scheme from numerical analysis. In our implementation, we used the trapezoidal rule to define the loss function.



$$\boldsymbol{y}_n := \sum_{m=0}^{M} \left[\alpha_m \boldsymbol{x}_{n-m} + \Delta t \beta_m \boldsymbol{f}(\boldsymbol{x}_{n-m})\right], \quad n = M, \ldots, N$$

$$MSE := \frac{1}{N-M+1} \sum_{n=M}^{N} |\boldsymbol{y}_n|^2$$

In multistep neural networks as well, we couple the neural network model with a numerical solver. We feed in new unseen initial conditions to this model, and it predicts the time evolution of state variables.

Results

In all the plots below, dotted lines represent prediction from the neural network and solid lines represent ground truth.

Spring mass system

We simulate the spring mass system using the *solve_ivp* function in the scipy library. The Hamiltonian for the spring mass system is as follows:

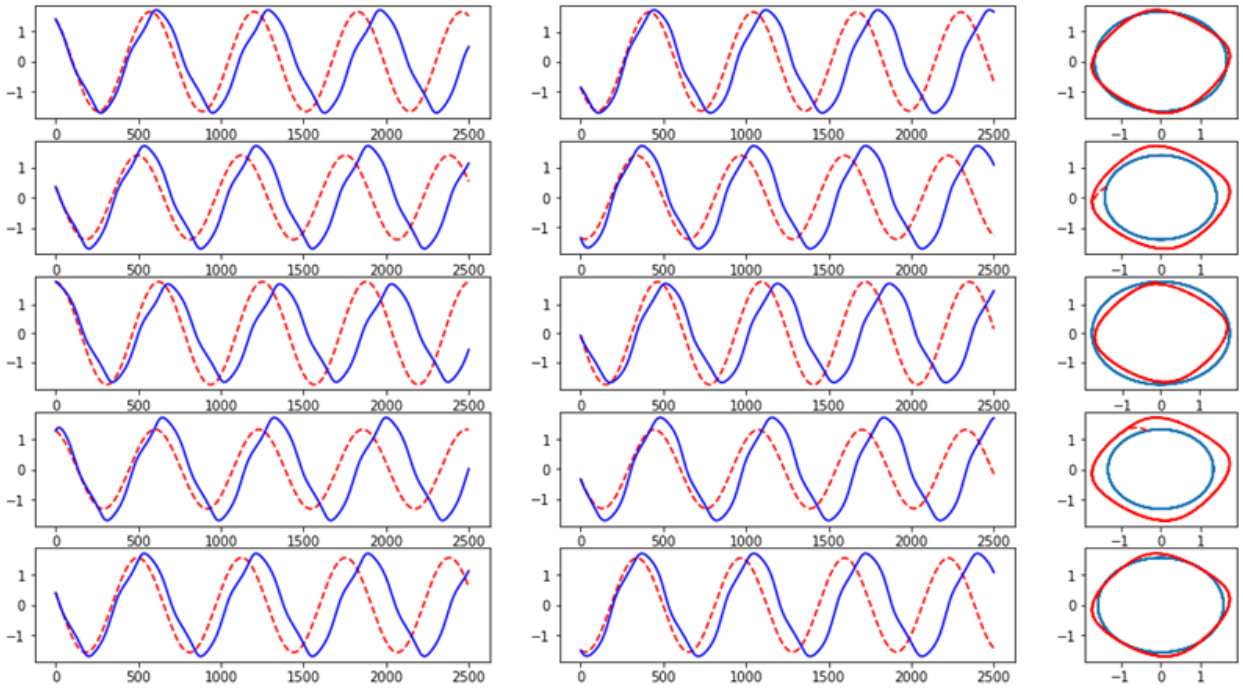
$$\mathcal{H} = \frac{1}{2}kq^2 + \frac{p^2}{2m}$$

The training data comprises 25 different trajectories of the position q and the momentum p of the simple harmonic oscillator for randomly chosen initial conditions. Each of the trajectory is 25s long and is sampled with a $\Delta t = 0.01s$.

LSTM (plots on the testing data)

The LSTM model we used had a LSTM cell with 200 units followed by a dense layer with 2 units which predicts q and p. We used a tanh activation function and the model was trained for 100 epochs.

The plots below show the results for the 5 unseen initial conditions, which comprises the test data. The first two columns show the trajectories of q and p vs t. The rightmost column shows the phase portrait.



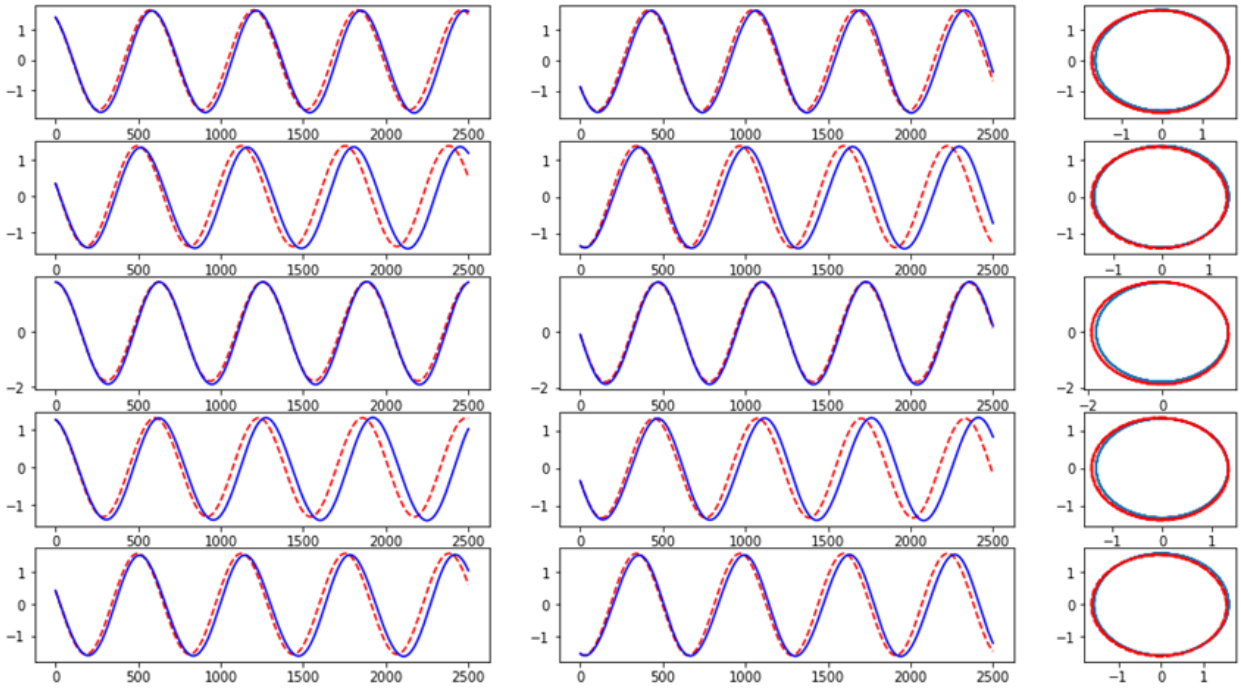
We can see that the LSTM predicts the exact trajectory accurately for a short duration, but it starts to drift apart after some time. However the predicted solution only drifts apart in terms of the phase. The overall trend of the actual solution is still learned by the LSTM model. From the rightmost column, we can see that the LSTM almost matches the true data in terms of the phase portrait, but still there are some regions where it deviates.

HNN (plots on the testing data)

In Hamiltonian Neural Networks, we use a 2 dense layer with 200 units each, and the output layer consists of one neuron. Here as well, we used the tanh activation function.

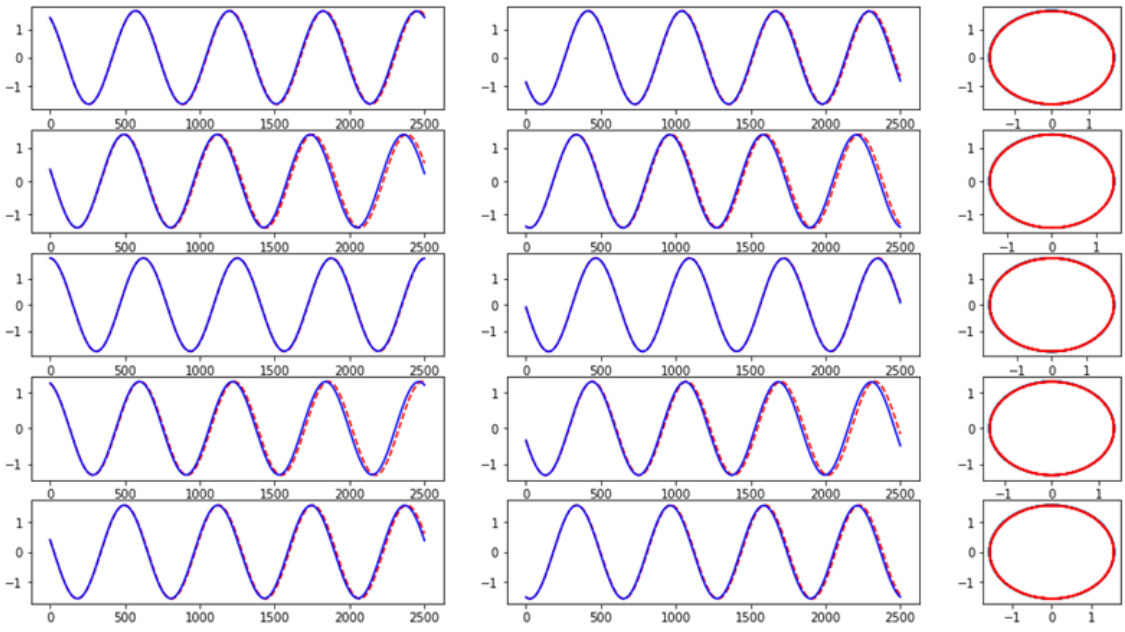
The plots below show the results for the 5 unseen initial conditions, which comprises the test data. The first two columns show the trajectories of q and p vs t. The rightmost column shows the phase portrait.

We can see that in the case of Hamiltonian neural networks, the solution drifts apart only by a very small amount with respect to the phase. In terms of the phase portrait, the solution from the Hamiltonian Neural Network matches the truth almost exactly. The prediction for the HNN is much better than that for the LSTM model.



Multistep neural networks (plots on the testing data)

In multistep neural networks, we use a dense layer with 200 units followed by a dense layer with 2 units which predicts *q_dot and p_dot*. Here as well, we have used the tanh activation function.



The plots below show the results for the 5 unseen initial conditions, which comprises the test data. The first two columns show the trajectories of q and p vs t. The rightmost column shows the phase portrait.

In the case of multistep neural networks, we can see that the solution matches the truth almost exactly. The phase portrait also matches the true phase portrait very accurately.

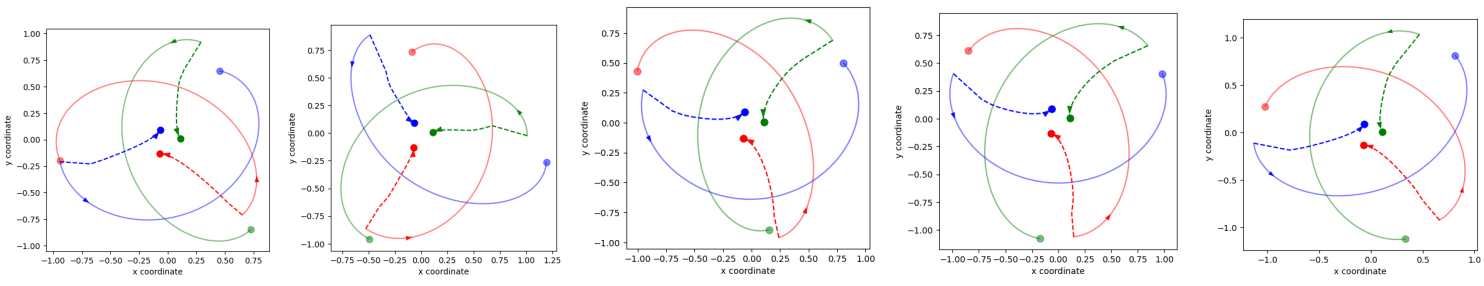
Three Body Problem

This problem consists of three masses which are positioned in space and given an initial momentum. The masses are then allowed to evolve under the influence of gravitational force. Here, the numerical simulations are performed for a duration of 3s with a timestep of 0.01s. The Hamiltonian for the three body problem is as follows:

$$\mathcal{H} = -\frac{Gm_1m_2}{|\mathbf{r_1} - \mathbf{r_2}|} - \frac{Gm_2m_3}{|\mathbf{r_3} - \mathbf{r_2}|} - \frac{Gm_3m_1}{|\mathbf{r_3} - \mathbf{r_1}|} + \frac{\mathbf{p_1}^2}{2m_1} + \frac{\mathbf{p_2}^2}{2m_2} + \frac{\mathbf{p_3}^2}{2m_3}$$

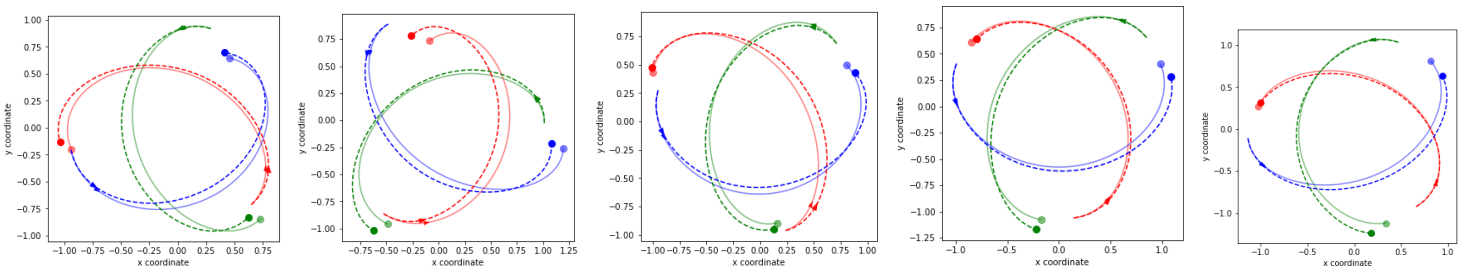
The training data consists of 25 different trajectories for randomly chosen initial conditions. The testing data consists of 5 new randomly chosen initial conditions.

LSTM (plots on the testing data)

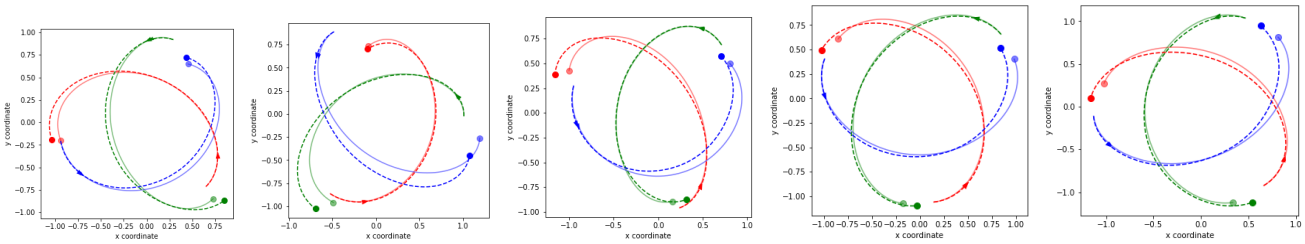


In the three body example, we used the same architecture as in the previous case except for changing the number of units in the output layer according to the dimensionality of the problem. In this benchmark problem, the LSTM fails to match the true trajectories. Both the Hamiltonian neural network and multistep neural networks do a decent job at predicting the trajectory for new unseen initial conditions.

Hamiltonian Neural Network(plots on the testing data)



Multistep neural network(plots on the testing data)



The plots shown are for the trajectories of the three bodies evolving under the influence of gravity. The plots are for the five test initial conditions which are distinct from the train initial conditions.

Lorenz system

The Lorenz system is a nonlinear dynamical system which exhibits chaos. It has applications in several fields ranging from climate modeling to fluid dynamics. It is described by the following set of differential equations:

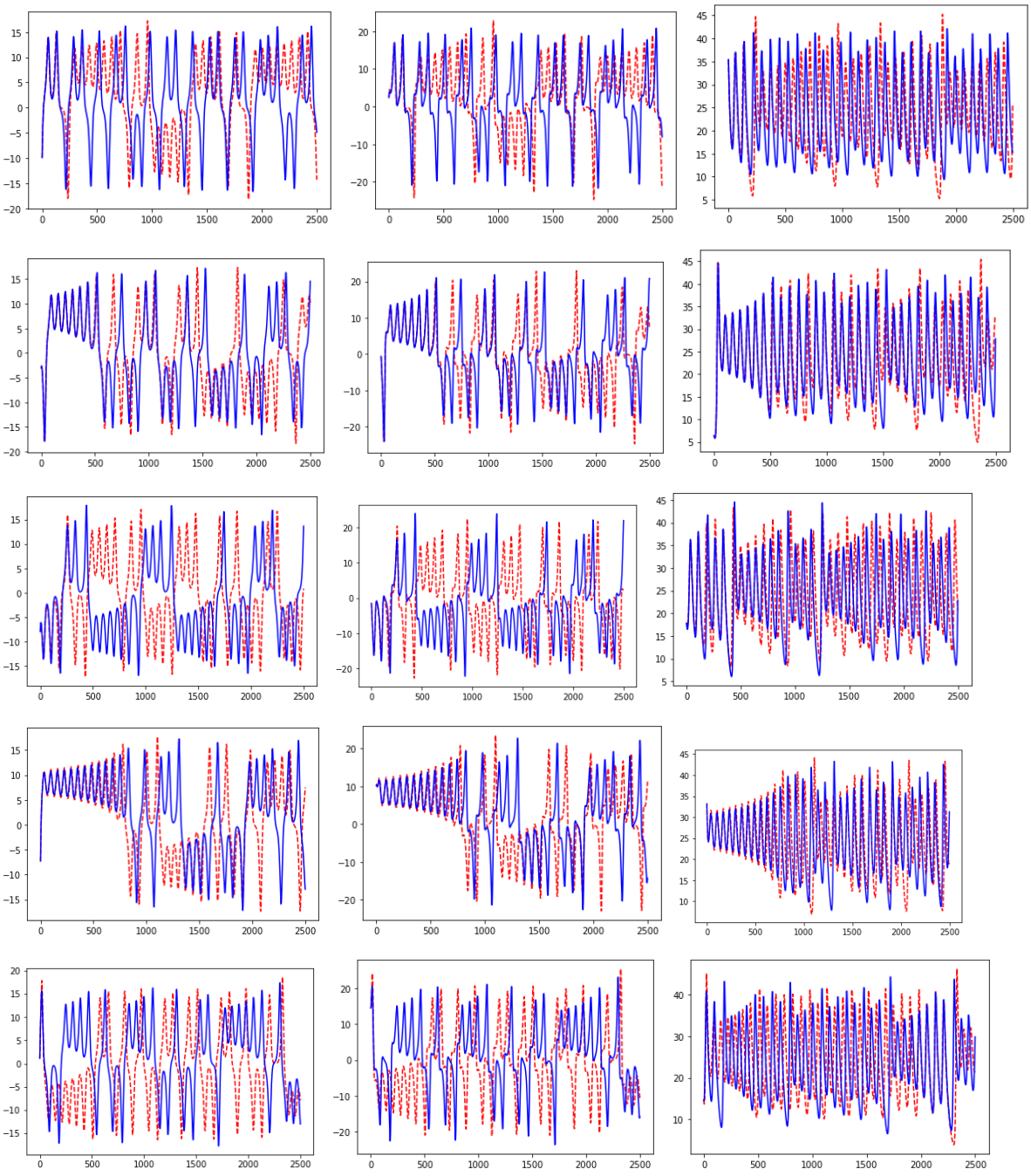
$$\frac{\mathrm{d}x}{\mathrm{d}t} = \sigma(y - x),$$

$$\frac{\mathrm{d}y}{\mathrm{d}t} = x(\rho - z) - y,$$

$$\frac{\mathrm{d}z}{\mathrm{d}t} = xy - \beta z.$$

Multistep Neural Networks (Plots on the testing data)

As, only the Multistep neural network was able to predict the dynamics of the system with descent accuracy on the testing dataset. We are not uploading the results from the LSTM and HNN. The columns indicate the x, y and z variables.



Key takeaways and learnings

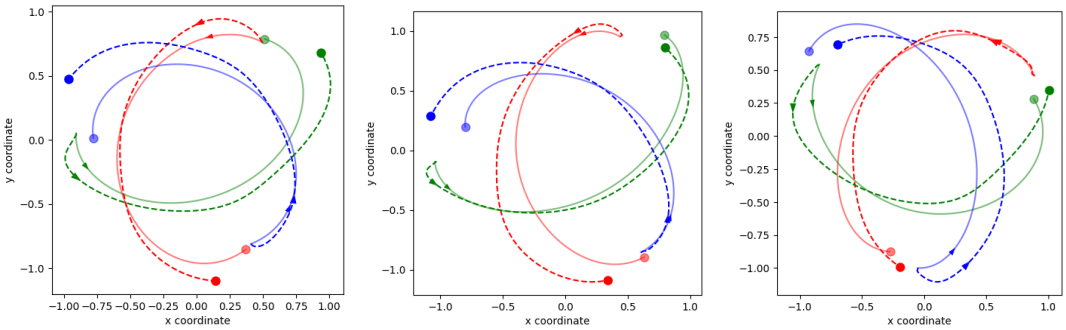
In our study we compared the Hamiltonian Neural Network, LSTM and Multistep Neural Network and learned the following ideas: The data to train HNN need not be uniformly spaced in time, contrary to LSTMs and multistep neural nets. The LSTM completely ignores physics and has less inductive bias. The Hamiltonian is not defined for all dynamical systems, hence for Lorenz, the performance was subpar.Multistep and HNN converge faster than the LSTM. Overall, the Multistep method outperformed HNN and LSTM in terms of prediction capability on all the 3 benchmark dataset.

Table of relative L2 error for the 3 models for the three benchmark datasets

	LSTM	HNN	Multistep
Simple Harmonic Oscillator	1.025	0.539	**0.253**
Three body problem	1.0744	**0.0745**	0.0854
Lorenz system	0.7818	160.277	**0.7151**

Challenges

First challenge we faced was how we implemented LSTM on this problem setup. From intuition, we were motivated as since it's a sequence of data, LSTM should be able to at least work. However, actually implementing after preprocessing the data was a bit challenging for us. Second challenge we faced was how to improve the prediction of LSTM. At the end, we concluded that it starts performing decently when we train it for a very high number of iterations. See the results below for the 3 body data set, we got after running the model with two stacked LSTM layers with 200 units each followed by the dense layer. Still we didn't reach the accuracy level of HNN and Multistep neural networks which converged way faster.



Reflection

- **How do you feel your project ultimately turned out? How did you do relative to your base/target/stretch goals?**

We compared our model with the 3 benchmark dataset on all 3 of our models which was our goal. Hence, in regards to the target goal we feel we achieved it successfully in terms of implementing the networks and testing them out. We feel we stretched ourselves when considering the effort required from our side just to test it on the 3 dataset and testing them to find if they can even perform decently? We tried LSTM with a large training data set, with more trainable parameters but ultimately, we realize it's really a tough task for LSTM to model the complex systems when it comes to predicting the dynamics of three body problems and the Lorenz system. LSTM was something new we wanted to try as this was not previously implemented by the researchers for the data sets we considered.

- **Did your model work out the way you expected it to?**

Yes, for the case of LSTM, we expected it would face issues as it has less inductive bias while the other two take advantage of physics. Hence, While having the same number of trainable parameters for all the three models, we did conclude that multistep outperforms HNN and LSTM. However, we didn't expect the amount by which LSTM lagged behind.

- **How did your approach change over time? What kind of pivots did you make, if any?**

We thought of doing the physics informed neural networks as well in the course proposal but given the time constraints, mentor TA suggested we drop it. We feel this was a good

decision. We considered the spring mass system as the toy problem for a dynamical system instead of a simple pendulum just for the sake of simplicity. Other than this, no major pivots.

- **Would you have done differently if you could do your project over again?**

We could have implemented our code in .py files rather than working with the jupyter notebooks. For each code of our network, we could have implemented them in a more modular framework and proper structure. This would have helped us in more automation when retraining and retesting the models over multiple scenarios.

- **What do you think you can further improve on if you had more time?**

We could have done hyper parameter sensitivity studies for each network model with the dataset and do some further ablation studies.

- **What are your biggest takeaways from this project/what did you learn?**
    - The data to train HNN need not be uniformly spaced in time, contrary to LSTMs and multiple neural networks.
    - LSTM completely ignores physics and has less inductive bias.
    - The Hamiltonian is not defined for all dynamical systems. Hence, for the Lorenz system dataset, the performance was subpar.
    - Last, multistep neural networks are cool!!!

References

1. Greydanus, Samuel, Misko Dzamba, and Jason Yosinski. "Hamiltonian neural networks." *Advances in neural information processing systems* 32 (2019).
2. Raissi, Maziar, Paris Perdikaris, and George Em Karniadakis. "Multistep neural networks for data-driven discovery of nonlinear dynamical systems." *arXiv preprint arXiv:1801.01236* (2018).