Challenge 1: Deep Learning for Differential Equations

ML4DE Hackathon April 2025

1 Introduction

This document outlines Challenge 1 of the ML4DE Working Group hackathon, focusing on the application of deep learning techniques to predicting solutions of differential equations. The challenge involves two key equations: the Kuramoto-Sivashinsky equation and the Lorenz equation and is based on the Common Task Framework (CTF) developed by Prof. J. Nathan Kutz and his team at the AI Institute in Dynamic Systems (https://www.synapse.org/Synapse:syn52052735/wiki/622928).

2 Problem Statement

The goal of this challenge is to implement deep learning models that can accurately predict the dynamics of two time-dependent DEs with unknown parameters.

2.1 Kuramoto–Sivashinsky equation

The first challenge problem is the prediction of solution values to the Kuramoto–Sivashinsky equation with periodic boundary conditions and unknown viscosity coefficient ν:

$$\begin{cases} u_t + uu_x + \nu u_{xx} + \nu u_{xxxx} = 0, & x \in [0, 30), t \in [0, 100], \\ u(0, x) = (2 + \cos x)^{-1}, & x \in [0, 30). \end{cases}$$

The challenge dataset contains the following values in the form of arrays:

$$\begin{aligned} \text{ks\_training.npy} &= (u(j\delta t, n\delta x))_{0 \le j \le 100, 0 \le n \le 127} \\ \text{ks\_preciction.npy} \approx \text{ks\_truth.npy} &= (u(j\delta t, n\delta x))_{101 \le j \le 201, 0 \le n \le 127}, \end{aligned}$$

where $\delta t = 0.5$ and $\delta x = 30/2048$.

2.2 Lorenz equation

The second challenge problem is the prediction of solution values to the Lorenz equation

$$\begin{aligned} \frac{dx}{dt} &= \sigma(y - x) \\ \frac{dy}{dt} &= \rho x - xz - y \\ \frac{dz}{dt} &= xy - \beta z \end{aligned}$$

with initial condition $x(0) = y(0) = z(0) = 1$ and unknown coefficients σ, ρ, β. The challenge dataset contains the following values in the form of arrays:

$$\begin{aligned} \text{lorenz\_training.npy} &= (x(j\delta t), y(j\delta t), z(j\delta t))_{0 \le j \le 5000} \\ \text{lorenz\_preciction.npy} \approx \text{lorenz\_truth.npy} &= (x(j\delta t), y(j\delta t), z(j\delta t))_{5001 \le j \le 10001}, \end{aligned}$$

where $\delta t = 0.01$.

3 Rules

1. Using the information about the DEs provided above and the data in `ks_training.npy` and `lorenz_training.py` you should implement an ML approach that generates `ks_prediction.npy` and `lorenz_prediction.npy`.

2. You are not allowed to use the true values `ks_true.npy` and `lorenz_true.npy` during training. These are only on the repository to compute the final scores.

3. The predictions will be scored in real time based on the metrics described in more detail below. The current score can be found at https://ml4de-hackathon.onrender.com

4. At the end of the day, each team is asked to upload their models for each equation to their team folders. These codes should allow the generation of the same scores as obtained in the rankings and will be validated by the hackathon organisers to make sure a valid ML approach was used.

5. Please do not use classical methods, such as Runge–Kutta methods etc. The organisers have full discretion on what presents a 'classical approach' and scores obtained in this manner will be declared invalid. If in doubt please do feel free to consult us at any point.

4 Details on evaluation metrics

The following is a condensed version of a set of notes written by J. Nathan Kutz describing the idea behind the evaluation metrics for the common task framework (https://www.synapse.org/Synapse:syn53646141).

The common task framework (CTF) for dynamic systems aims to evaluate algorithms and methods on a variety of tasks that are common for engineering and science. The goals include forecasting and reconstruction of time-series and spatio-temporal data under the challenges of limited data, noise and parametric dependence.

4.1 Scoring

Each metric takes values in $[-\infty, 100]$ with a score of 100 corresponding to a perfect match. There is a short-term and a long-term forecast score for each equation.

4.2 Kuramoto-Sivashinsky

There are two scores evaluating both the short-time forecast (the "weather forecast") which is computed using root-mean square error fitting between the test set and the user's approximation, and the long-term forecast (the "climate forecast"), which is based upon the power spectral density. As such, the following two error scores are computed:

$$E_{\mathrm{ST}} = \frac{\left\| \hat{\mathbf{X}}_1[1:k,:] - \tilde{\mathbf{X}}_1[1:k,:] \right\|}{\left\| \hat{\mathbf{X}}_1[1:k,:] \right\|} \quad \text{(weather forecast)}$$

and

$$E_{\mathrm{LT}} = \frac{\left\| \hat{\mathbf{P}}_1[N-k:N,\mathbf{k}] - \tilde{\mathbf{P}}_1[N-k:N,\mathbf{k}] \right\|}{\left\| \hat{\mathbf{P}}_1[N-k:N,\mathbf{k}] \right\|} \quad \text{(climate forecast).}$$

Here $\hat{\mathbf{X}}_1[j,n]$ represents the reference solution values at $u_{true}(j\delta, n\delta x)$ and $\tilde{\mathbf{X}}_1[j,n]$ represents the corresponding ML prediction. For us $N = 201, k = 20$.

Here, E_{ST} is the short-time error evaluated using least-squares and E_{LT} is the long-time error which is computed by least-squares fitting of the power spectrum $\mathbf{P}_j[k,:,] = \ln\left(|\mathrm{FFT}\left(\mathbf{X}_j[k,:]\right)|^2\right)$, where the fftshift has been used to model the data in the wavenumber domain and $\mathbf{k} = n/2 - k_m$: $n/2 + (k_m + 1)$ with $k_m = 20$. This then looks at the first 40 wavenumbers in order to determine the decay rate of the power spectrum. It is clear that there are many ways to evaluate the long-range forecasting capabilities. However, we have chosen a simple metric fully understanding that more nuanced scoring could be used. For the challenge dynamics of interest, sensitivity of initial conditions is common so that long range forecasting to match the test set is not a reasonable task given fundamental mathematical limitations with Lyapunov times.

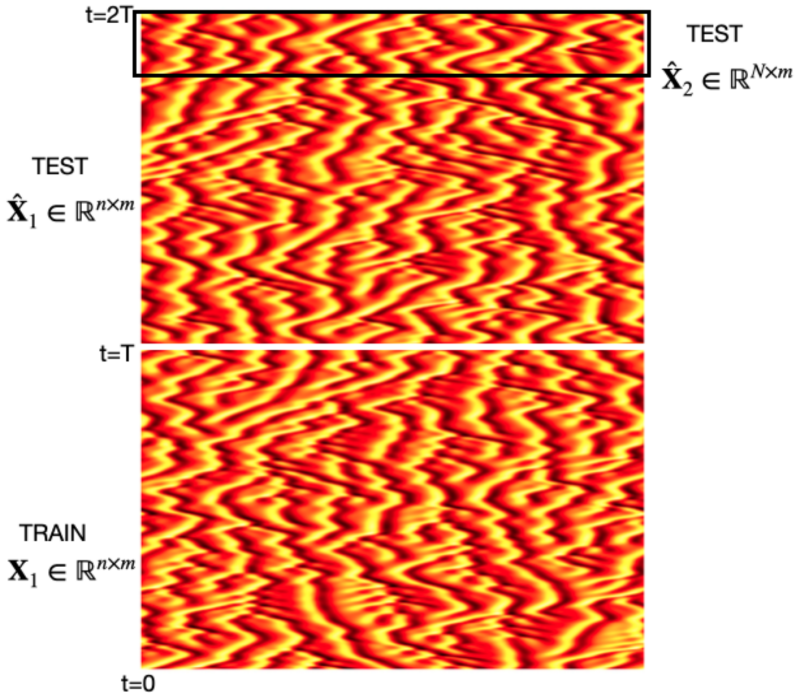

Figure 1: Example of training and test data in the KS equation.

4.3 Lorenz Dynamical System

The short term scores is identical to the above case, but the long range (climate) forecast score differs. Data matrices for testing and training are now of the form:

$$\mathbf{X} \in \mathbb{R}^{3\times 5001}$$

It is thus inappropriate to use the power spectral density of the differential equation since it is no longer a spatial system. Instead, the long-time forecasting considers the histogram of the evolution of the variables over the last k_{LT} time steps (e.g. $k_{LT} = 1000$). The histogram for a time series is computed using the histogram command with the bins specified. For the x variable in Lorenz, this would give

```
M = np.arange(-20, 21, 1)
yhistxt, xhistx = np.histogram(yt[:, 0], bins=M)
```

The difference of the histogram between the truth (x, y and z) and prediction ($\tilde{x}, \tilde{y}$ and $\tilde{z}$) and for each variable is given by

$$E_{\mathrm{LTx}} = \frac{\|Hist_x - Hist_{\tilde{x}}\|}{\|Hist_x\|}$$

This gives the long-time error score:

$$E_{\mathrm{LT}} = \left(E_{\mathrm{LTX}} + E_{\mathrm{LTY}} + E_{\mathrm{LTZ}}\right)/3 \quad \text{(climate forecast)}$$

As with the spatio-temporal system and the power spectral density, this gives a simple measure of the accuracy of the prediction from a statistical viewpoint since long-time prediction is well beyond the Lyapunov time which would not allow for a least-square match between trajectories of the truth and prediction.