Flood Forecasting model with Neural Controlled Differential Equations

Karneedi Y B Aparna

21CE31008

Why Neural CDE’s:

- Neural CDE’s handles with Irregularly sampled (uneven time gaps), Partially observed (missing values), Multivariate and noisy data.

- RNNs and LSTMs assume fixed time steps and can’t handle irregular sampling naturally.

- Neural ODEs model define dynamics from a fixed initial condition, limits it's flexibility in adapting to new observations.

Neural CDE’s :-

- Neural CDEs continuously evolve the hidden state based on a path of observations (not just time).

$$z_t = z_{t_0} + \int_{t_0}^{t} f_\theta(z_s)\mathrm{d}X_s$$

zt– Hidden State at Time t
zt0– Initial Hidden State
Xs- Input Control Path
fθ(zs) – Vector Field / Dynamics Function
Riemann–Stieltjes integral is pathwise integral over the input data path Xs, not just over time.

Objective :-

- To build an accurate and robust flood forecasting model for Hirakud Dam by using the concept of Neural Controlled Differential Equations (Neural CDEs).
- Hirakud Dam inflow estimation by using rainfall, runoff & discharge datasets
- Planning to use a 30-day sliding window to predict the next day's inflow at Hirakud.

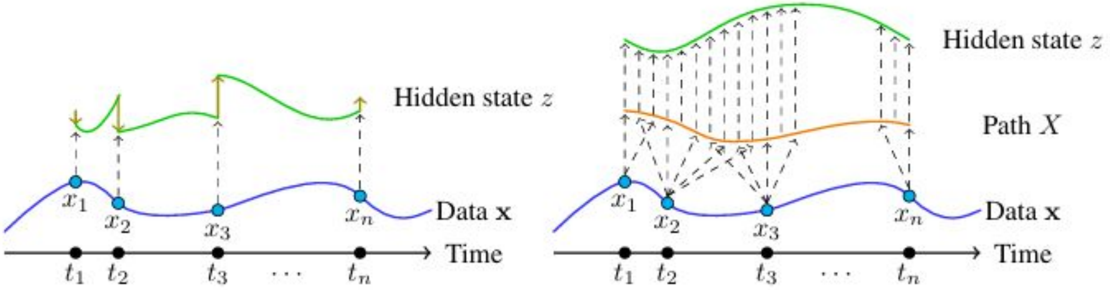


Figure 1: Some data process is observed at times $t_1, \ldots, t_n$ to give observations $x_1, \ldots, x_n$. It is otherwise unobserved. **Left:** Previous work has typically modified hidden state at each observation, and perhaps continuously evolved the hidden state between observations. **Right:** In contrast, the hidden state of the Neural CDE model has continuous dependence on the observed data.

Data :-

- We have rainfall and runoff data, each consisting of 3050 daily entries across 158 stations.

- We also have discharge data from 12 stations, and also from the Hirakud dam.

- The features used in the model are the mean values of rainfall, runoff, and discharge (excluding Hirakud).

- The target variable is the Hirakud Inflow, measured in cubic meters per second (m^3/s).

Data Preprocessing:-

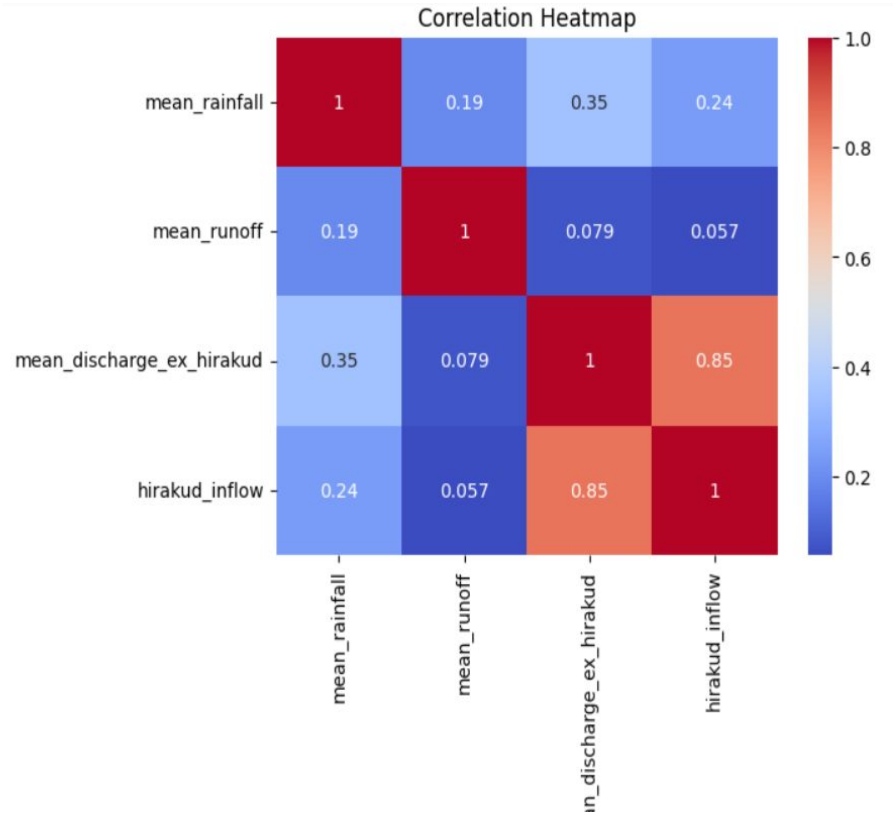
- Prepare input features(mean_rainfall, mean_runoff, and mean_discharge (excluding Hirakud)) for a Neural CDE model from hydrological time series.

- First 5 days of rainfall data were removed(because of antecedent Condition)

- Min-Max Scaling is used to normalize features between 0 and 1.

- Combined all features into a multivariate time-series tensor. Generated Hermite Cubic Spline coefficients using torchcde

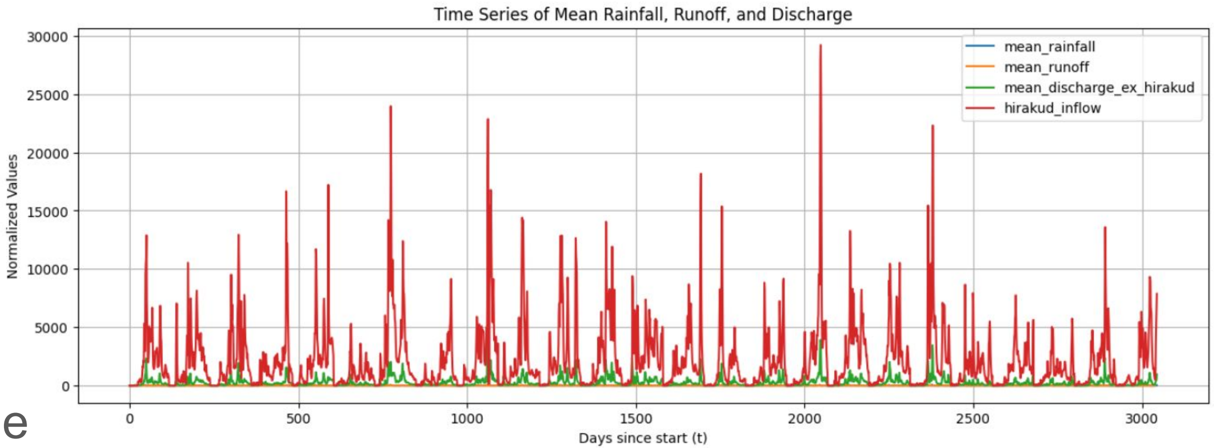
- Enables continuous-time interpolation required by CDEs.



Exploratory Data Analysis:-

- The strongest predictor of Hirakud inflow is clearly mean discharge (excluding Hirakud).
- Rainfall is a moderate predictor (correlation ≈ 0.24).
- Runoff is a weak predictor (correlation ≈ 0.06).

Time Series Analysis:-



- Lag behavior and non-Linear responses are visually evident.

- Patterns vary in magnitude and timing across the year indicating the seasonality and memory.

- Neural CDEs are used to Capture lag, memory, and evolving dependencies across variables. Interpolate smoothly between timesteps using Hermite cubic splines.

- This temporal pattern and interaction are not well captured by traditional RNNs/LSTMs.

Neural CDEs:-

3.1 Universal Approximation:-

- Neural CDEs can act as universal function approximators on path space.

Theorem (Informal): A linear map on the terminal value of a Neural CDE is a universal approximator for functionals of sequences.

3.2 Evaluating the Neural CDE:-

For differentiable paths X(s), the Neural CDE takes the form:

$$g_{\theta,X}(z,s) = f_\theta(z)\frac{\mathrm{d}X}{\mathrm{d}s}(s),$$

The hidden state is updated continuously:

$$z_t = z_{t_0} + \int_{t_0}^{t} f_\theta(z_s)\mathrm{d}X_s = z_{t_0} + \int_{t_0}^{t} f_\theta(z_s)\frac{\mathrm{d}X}{\mathrm{d}s}(s)\mathrm{d}s = z_{t_0} + \int_{t_0}^{t} g_{\theta,X}(z_s,s)\mathrm{d}s.$$

Neural CDEs Over Neural ODEs:-

Limitations of Neural ODEs:

- Models the evolution of hidden state as:

$$z_t = z_0 + \int_0^t f_\theta(z_s)\mathrm{d}s$$

Integrates over time, not the observed data and the entire trajectory depends only on the initial hidden state

- Cannot naturally adapt to new observations once started

- Requires regular sampling, struggles with missing data

Advantages of Neural CDEs:

- Hidden state evolves based on input path, not just time:

$$z_t = z_{t_0} + \int_{t_0}^t f_\theta(z_s)\mathrm{d}X_s$$

Xs: Continuous path created from the observed data

- Incorporates entire sequence of observations and naturally handles the Irregularly sampled, Partial/missing data

- Continuous updates as data arrives

Empirical Results from Neural CDE Paper:-

1. Character Trajectories Dataset:-

Task: Handwritten character recognition from time series pen strokes

Main Challenge: 70% missing data introduced artificially

Result: Neural CDE achieved ~98.6% accuracy, outperforming all baselines

- Memory usage: Only 1.3 MB, vs. 15–17 MB for GRU-based models
- So Robust even under heavy data loss

2. PhysioNet Sepsis Prediction:-

Task: Predicting risk of sepsis from irregular medical time series

Challenge: Irregular Sampling, Missing Values, Observational Intensity Bias, Long Sequences with Noise

Input: Vitals + observational intensity (how often data is recorded)

Result: Neural CDE reached 0.88 AUC

- Outperformed RNNs, GRU-ODEs, and ODE-RNNs
- 5× lower memory than recurrent alternatives

3. Speech Commands Dataset:-

Task: Audio command classification from regularly sampled sound waves

Challenge: Short but High-Frequency Sequences, Training Instability in Other Models, Need for Continuous-Time Modeling, Time-Invariance

Result: Neural CDE achieved 89.8% accuracy

- Other models struggled to converge consistently
- Demonstrates versatility for both regular and irregular time series

Architecture of Neural CDE:-

- Models hidden state evolution z(t) driven by a continuous input path X(t)
- Generalizes RNNs/ODEs by allowing the input itself to control the dynamics

Differential Equation:-

$$\frac{dz(t)}{dt} = f_\theta(z(t)) \cdot \frac{dX(t)}{dt}$$

Input Path $X(t)$: Continuous interpolation of discrete observations (e.g., rainfall, runoff, discharge)

Hidden State $z(t)$: Latent representation evolving over time via a controlled differential equation

Vector Field $f_\theta(z)$: Neural network that defines how $z(t)$ changes in response to $X(t)$

- Interpolator is used to convert discrete data to a smooth input path $X(t)$
- Neural CDE Solver is used to integrate $z(t)$ using the learned dynamics
- Readout Layer maps final state $z(T)$ to output (e.g., prediction)

Training Strategy:-

1. Input Preparation:-

Convert discrete sequential data (e.g., rainfall, runoff, discharge) into continuous path X(t) using an interpolator and construct initial hidden state z(t0) (e.g., learned or fixed vector)

2. Model Forward Pass:-

Solve the controlled differential equation:

$$\frac{dz(t)}{dt} = f_\theta(z(t)) \cdot \frac{dX(t)}{dt}$$

Use an ODE solver to integrate from t0 to T, yielding final state z(T)

3. Readout & Loss Computation:-

Pass z(T) through a readout layer to obtain prediction y^ and compute loss using appropriate criterion (e.g., MSE for regression)

4. Backpropagation:-

Use the adjoint sensitivity method or automatic differentiation to compute gradients through the ODE solver and Optimizer (e.g., Adam) updates parameters of fθ and readout

5. Iterative Training:-

Repeat over all input sequences in mini-batches and monitor metrics (loss, MAE, etc.) and adjust hyperparameters

Limitations of Neural CDEs:-

-->Computational Overhead

-->Sensitivity to Interpolation

-->Complexity of Tuning

-->Limited Support in Frameworks

-->Data Requirements

THANK YOU!!