

Project:

A new methodology for comparing performance of prediction algorithms for chaotic dynamical systems

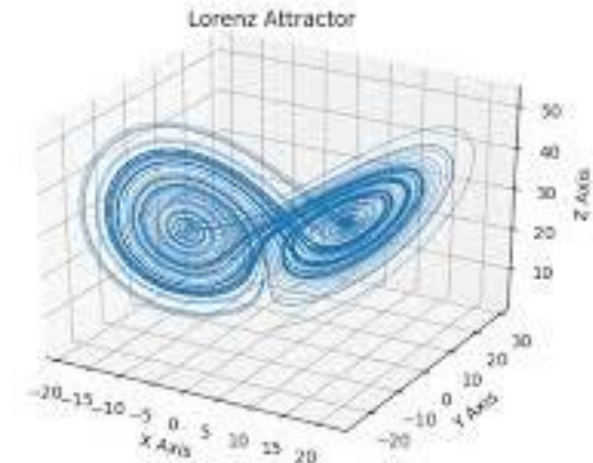
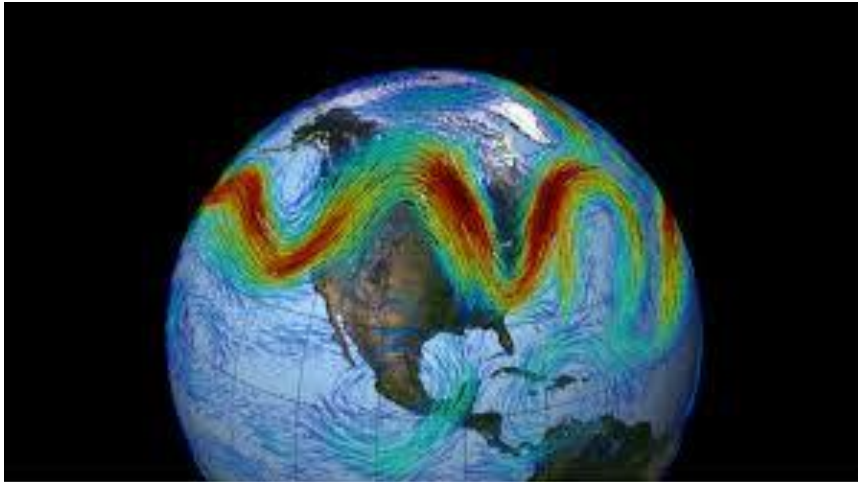
Instructors: Krishna Palem and Robert Cartwright

Project supporters: Devika Subramanian and Nikola Jovanovic

Technical participant: Adam Duracz

Problem description

- Starting with the goal of weather simulation, we first have to tackle some simpler problems that are similar in character
- These simpler problems are predicting the behavior of chaotic dynamical systems such as variants of Lorenz and KS systems (benchmark synthetic problems)



Our goals

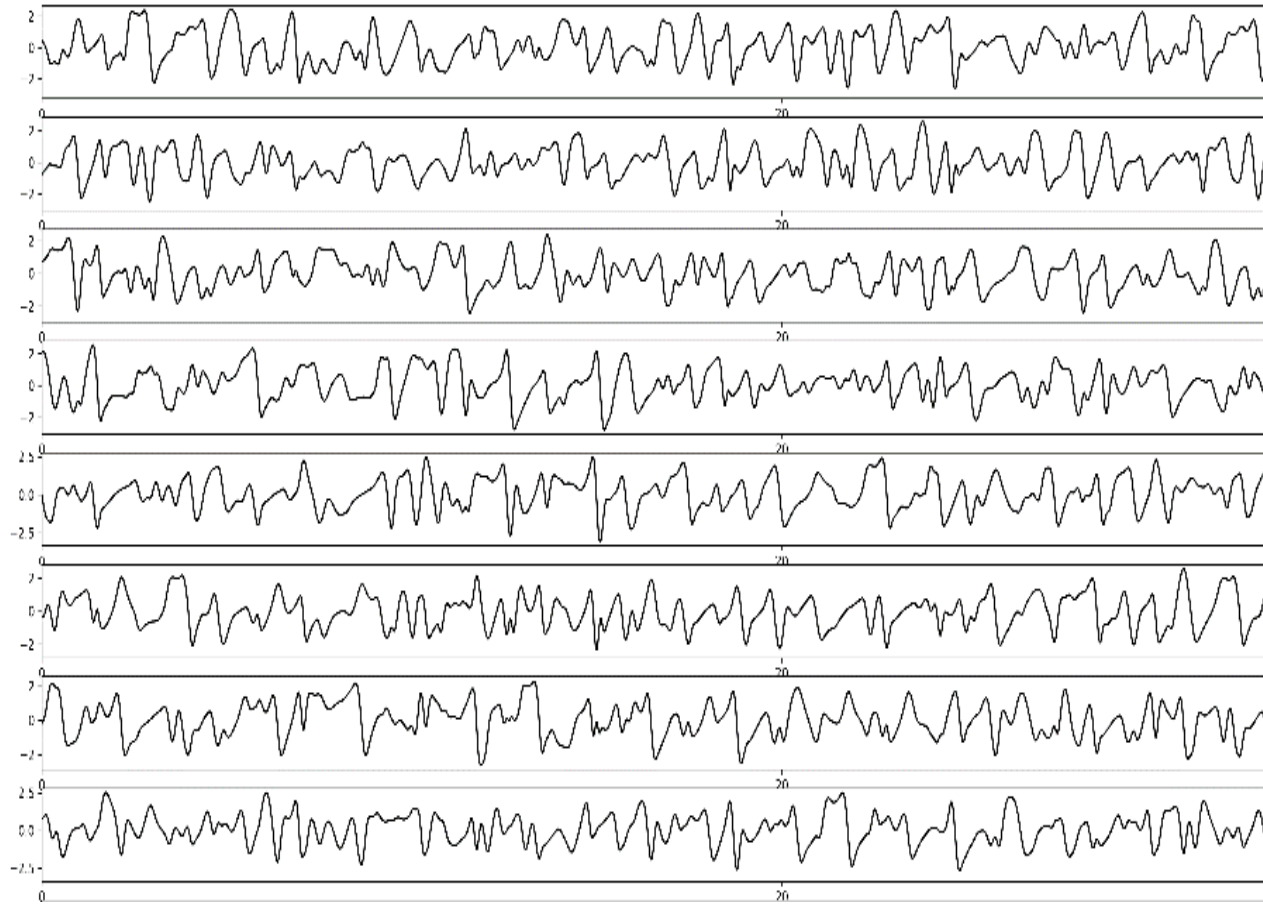
- Predicting chaotic dynamical systems time series behavior using machine learning models such as:
 - ESN (Echo State Networks),
 - LSTM (Long-Short Term Memory)
 - Ensembles of ESNs and LSTMs
- Finding the best predictive model for each of the benchmarks by developing a robust methodology for model evaluation and comparison
- Understanding why a specific model architecture performs better than others on a particular problem

Some terminology

- ***Model*** – Implementation of specific machine learning algorithm
- ***Model parameters*** – Number of input elements and hidden units, training set size, model specific constants, randomization seeds
- ***System state*** – Set of numerical values (***system variables***)
 - There are 8 system variables for Lorenz96 system
- ***IC (Initial condition)*** – System state the prediction is starting from
- ***MTU*** (Model Time Unit) – Unit of time describing chaotic system data
 - For Lorenz96 system, 1 MTU is an interval containing 200 discrete time points
 - Each time point is represented by one system state which can be used as an initial condition

Time-series data taken from Lorenz96 system, containing 8 system variables

Absolute variable values, ranging from -2 to 2



Propagation through time

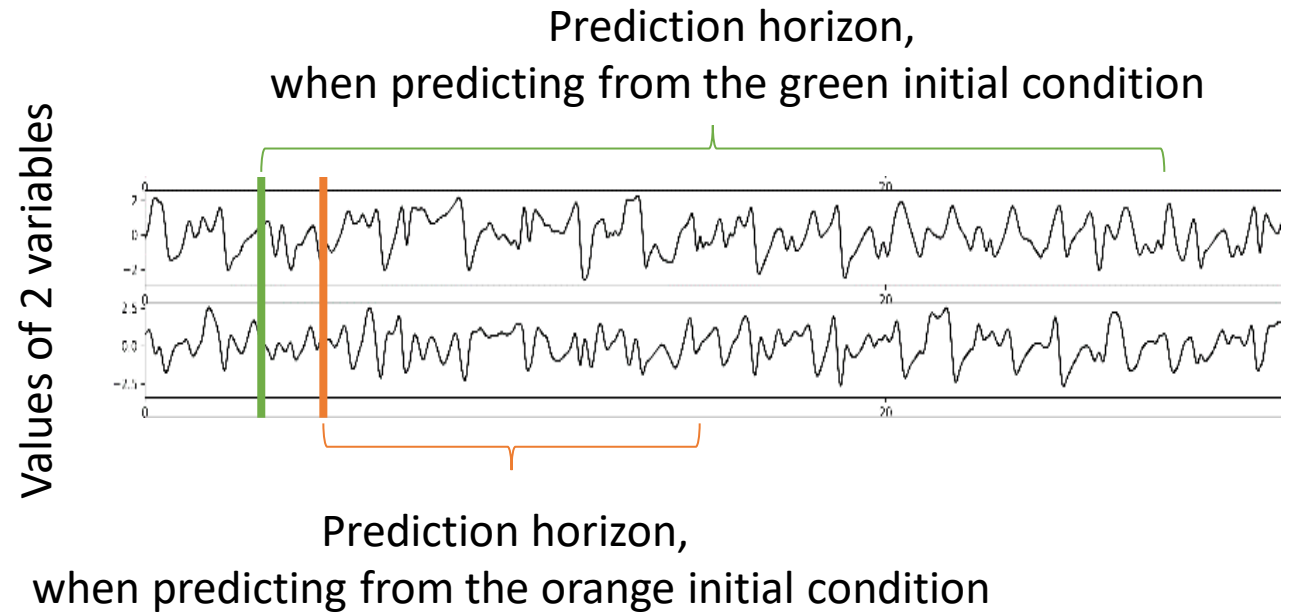
For each variable derivative is calculated using the following equation:

$$\frac{dx_i}{dt} = (x_{i+1} - x_{i-2})x_{i-1} - x_i + F$$

F Forcing factor, influencing the complexity of generated data (degree of chaotic behavior)

Prediction horizon

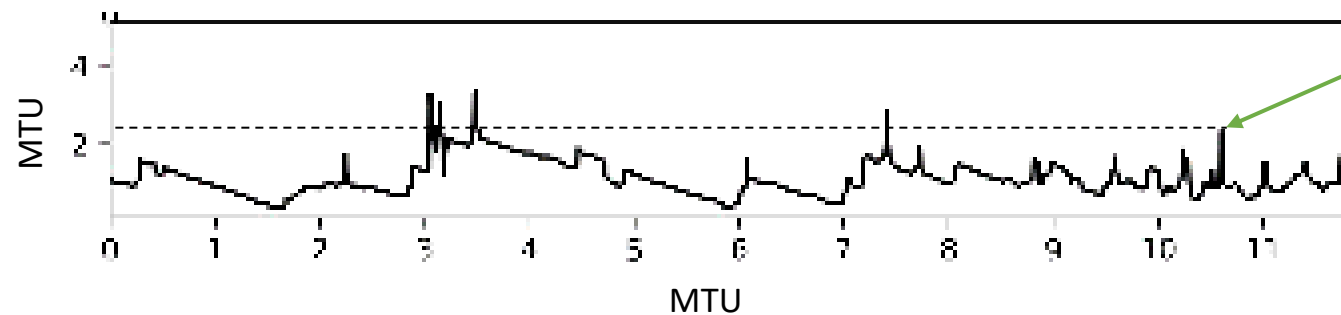
- Prediction horizon: how many predictions of the next time-step produced small enough error compared to the actual data, before the error accumulates
- As you can see from this example, two very close IC-s can produce very different result, with no apparent reason
- This makes modeling chaotic systems very difficult



Prediction horizon on consecutive IC-s

- Following graph shows prediction horizons, achieved on a set of consecutive initial conditions
- Each point on the graph represents the prediction horizon achieved on one of the consecutive initial conditions

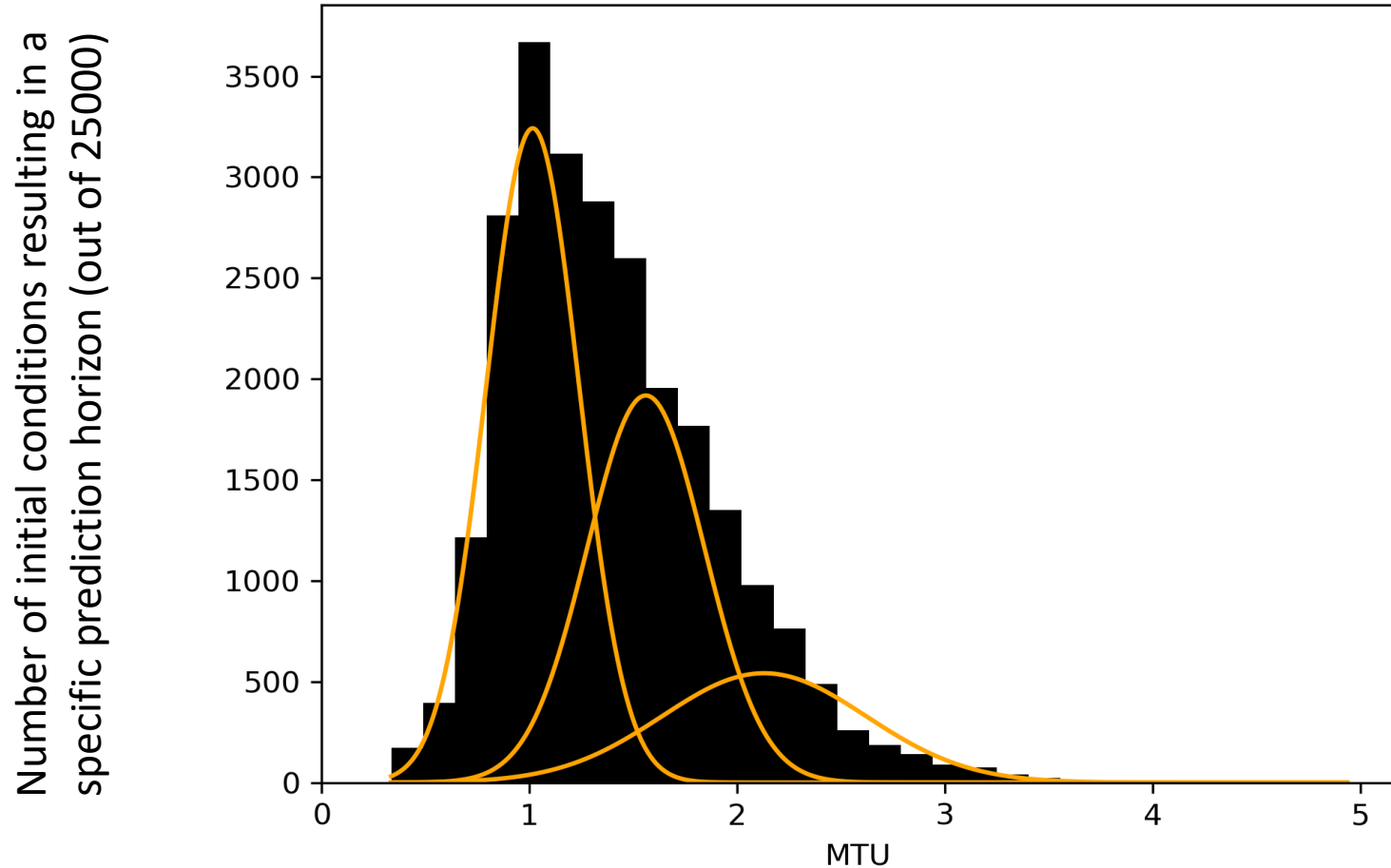
Prediction horizon value for each
densely plotted initial condition



Relative time point of each evaluated initial condition

E.g. prediction starting from the initial condition at 10.6 MTU from the first one in the set, has produced the prediction horizon of 2.2 MTU

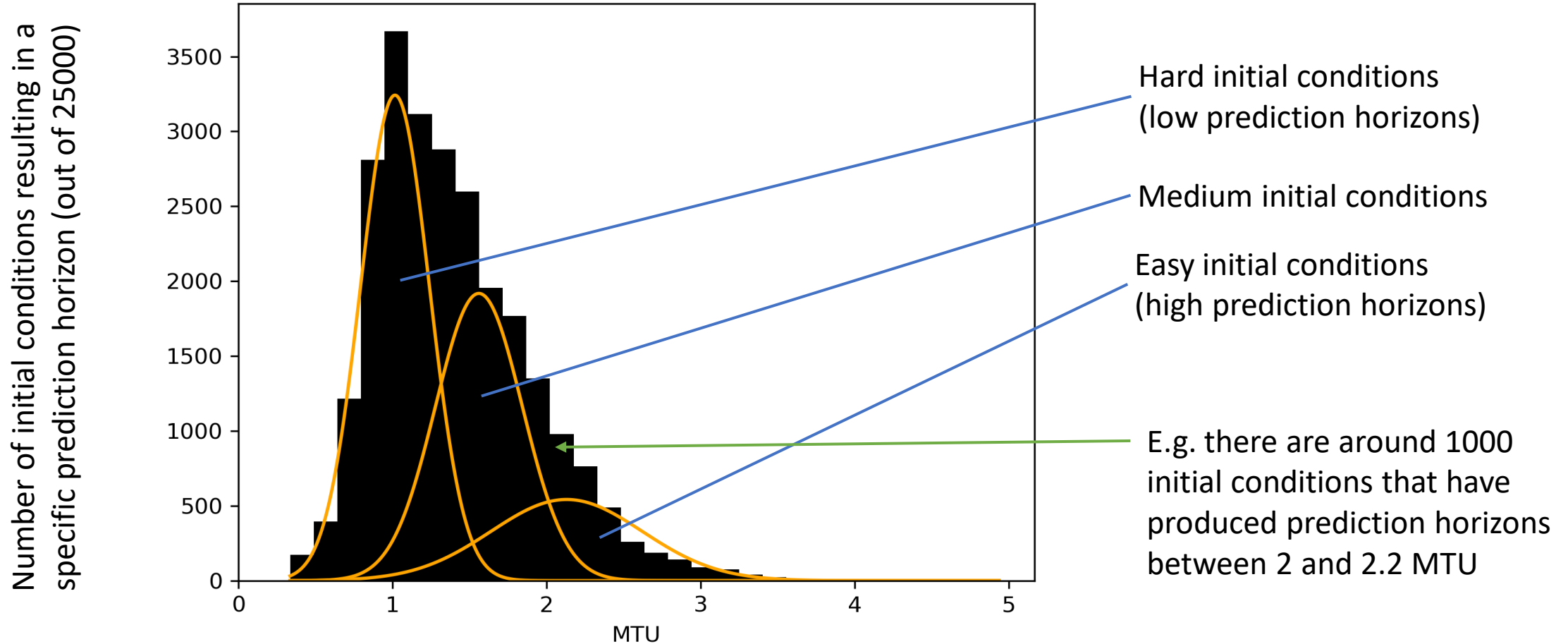
Prediction horizon histogram



Prediction horizon values achieved by evaluated initial conditions

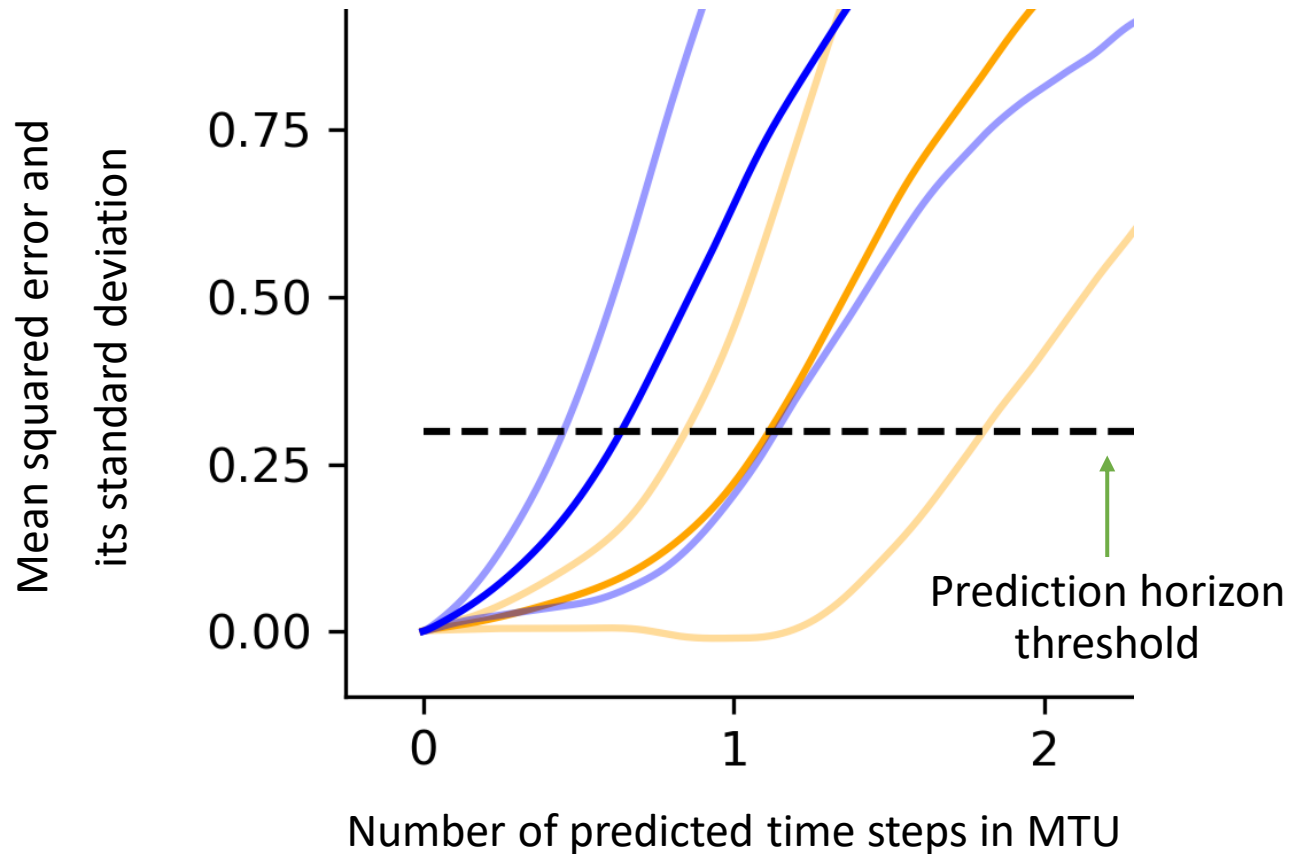
- Generated histogram does not nicely represent any common distribution, so it is not possible to correctly analyze standard deviations
- Problem is solved by dividing the histogram using Gaussian mixtures

Prediction horizon histogram



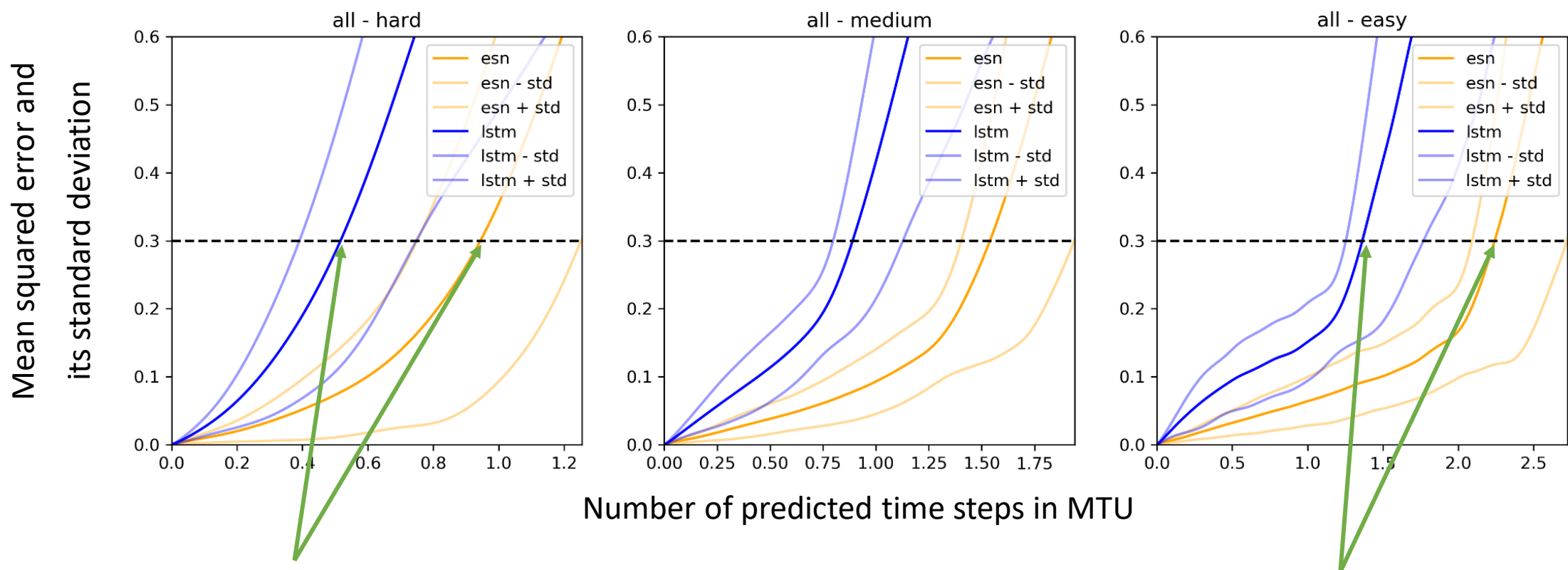
Prediction horizon values achieved by evaluated initial conditions

Average error and variation of two models



- The average error and its standard deviation is displayed by blue, and another's by orange line
- Although one model seems to cross the prediction horizon threshold sooner, the graph is still ambiguous

Average error and variation of two models on hard, medium and easy initial conditions



E.g. average LSTM error crosses the threshold at 0.5 MTU, while average ESN error crosses the threshold at 1.0 MTU, for initial conditions regarded as hard

Now the ESN model is clearly better

Our benchmark framework

- The ML models are tested on predicting chaotic dynamical systems
- Each chaotic dynamical system consists of a system of PDEs (Partial Differential Equations) which manifest highly unpredictable behavior in the very short term (tens of time-steps)
- Problems that we plan to benchmark:

Lorenz: Lorenz96 , Lorenz63 , Lorenz40	$\frac{dx_i}{dt} = (x_{i+1} - x_{i-2})x_{i-1} - x_i + F$
Kuramoto-Sivashinsky (KS)	$\frac{d}{dt}u_k = (k^2 - k^4)u_k + \sum_{k'} k' u_{k'} u_{k-k'},$
One more problem to be named later	

Data generation

- We produced Python scripts that can be found inside the project folder `data_sets` and implement basic Runge–Kutta integrator for systems we used:
 - Lorenz3 and LorenzN
 - KS
- This code can be used as template for implementation of integrators for other systems
- This produced data is later used for training and testing models, as well as “ground-truth” data for model validation and comparison

State of our work

- We have mostly completed all code implementation steps including training and evaluating models, as well as analyzing model predictions and comparison.
- We have focused on Lorenz96 variant of Lorenz system and produced analysis on its data.
- To gain enough results and support our methodology, we are aiming to apply our project pipeline to additional chaotic dynamical systems benchmark problems.

The analysis pipeline

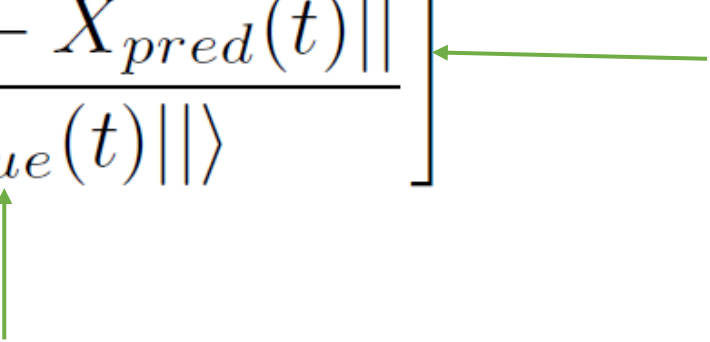
- Acquire dataset for analyzed chaotic system
- Parametrize the code to use desired:
 - Input data
 - Model type (ESN, LSTM, ensemble)
 - Model hyper-parameters (input dimension, model size, ...)
 - Execution hyper-parameters (size of training and testing sets, ...)
- Train the model and execute prediction analysis on multiple sets of execution hyper-parameters to find the best ones for the model
- Execute comparative analysis between different models

Getting started with the project

- Familiarize with the code, play with parameters and try to reconstruct existing results
- Project file structure:
 - Folder script – Implementations of ML models used in the project
 - Folder datasets – Dataset generators and generated chaotic system data
 - Folder Lorenz_96, 200, 1 – Generated data buffer, avoiding repeated calculation
 - default_esn.json, default_lstm.json – Auto-loaded model hipper parameters
 - default_run.json – Prediction hipper parameters (dataset split, ...)
 - pred_horizon_stats.py – Train the model, execute prediction and plot graphs
 - combine_graphs.py – Loads predictions of two existing trained models, executes comparative analysis and plots graphs

The error metric

- The error metric we used to estimate the quality of prediction is an averaged relative L_2 error between the true and predicted trajectories
- Following formula represents the average error on each time-step

$$e(t) = \left[\frac{||X_{true}(t) - X_{pred}(t)||}{\langle ||X_{true}(t)|| \rangle} \right]$$


Average over all predicted initial conditions (25000)

Average of absolute X_{true} over all predicted initial conditions (25000) for each time-step t

Your task

- Apply the provided code on data representing ***different chaotic dynamical systems***:
 - Adapt the code parametrization to align with dataset properties (dataset name and input path, input size)
 - Execute ***predictions*** and ***search for the best model parameters*** for the given system (using smaller and quicker testing sets), for different models (ESN, LSTM, ensemble)
 - Execute ***performance analysis*** on bigger testing set for each model (this will take the most execution time)
 - Execute ***comparative analysis*** between different models using the same predictions generated during individual performance analysis

Thank you for the attention