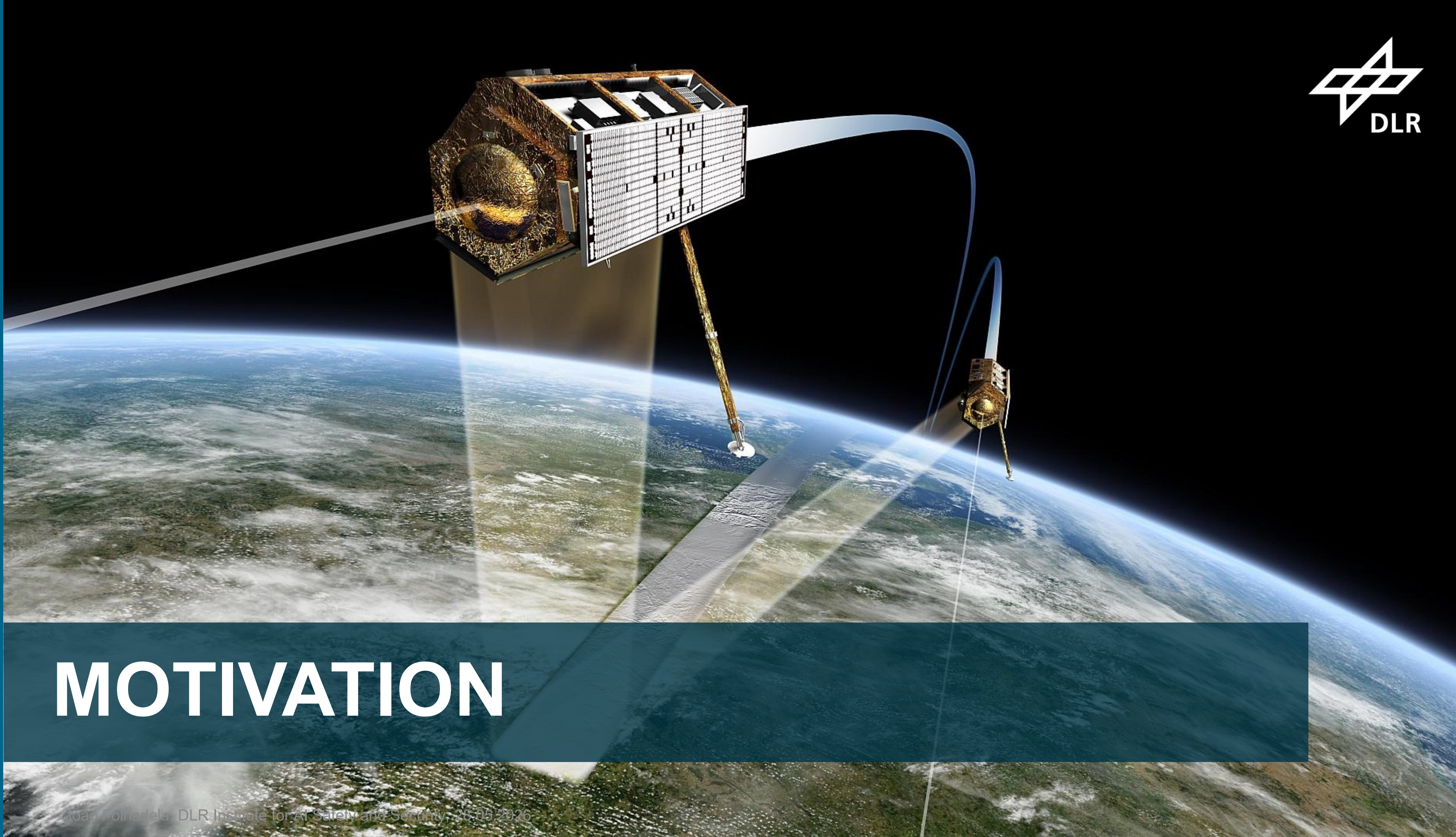
NGRC ON SOC FOR ONLINE LEARNING OF DYNAMICAL SYSTEMS

João C. L. Folhadela & Hans-Martin Rieser

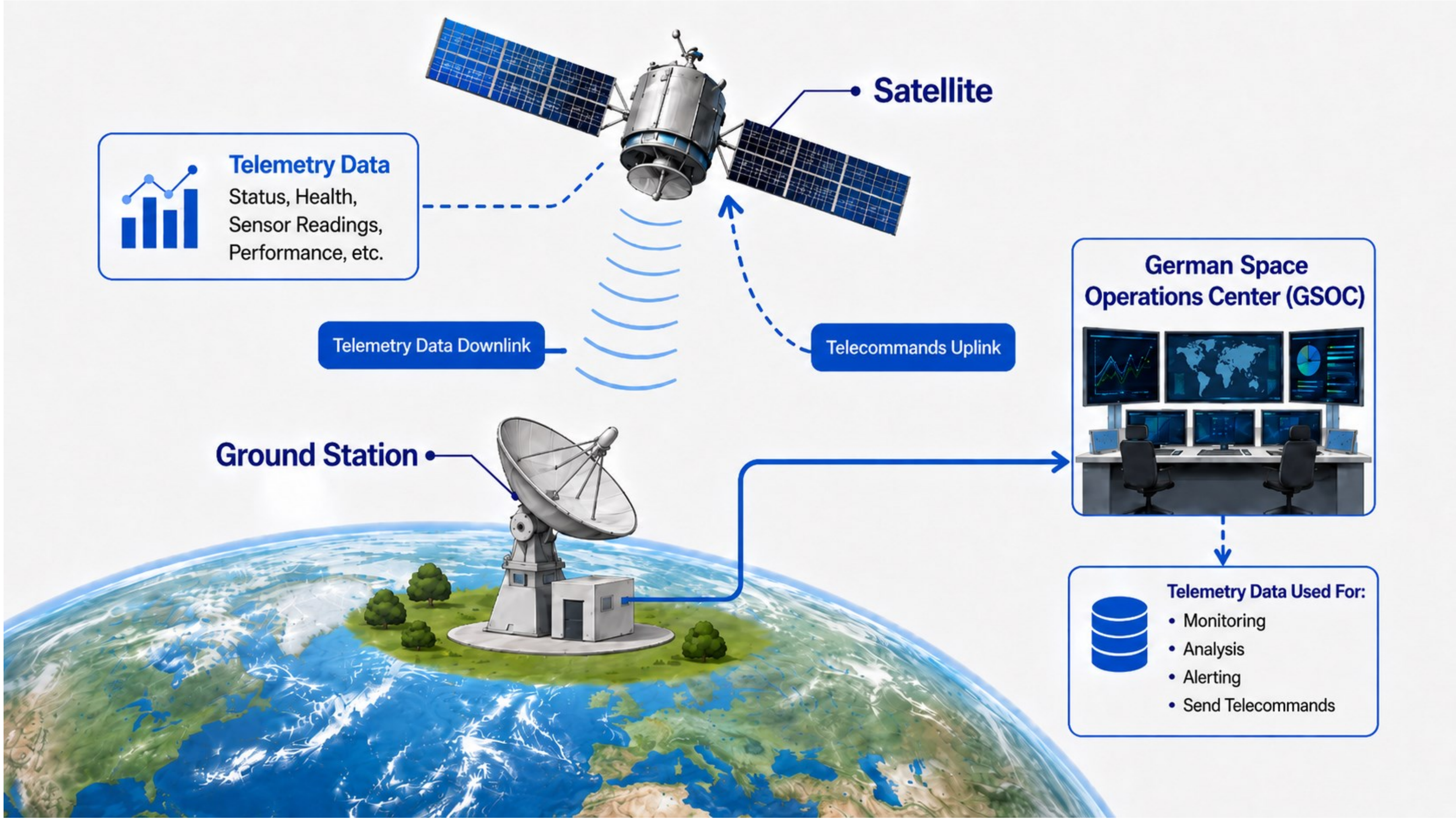
German Aerospace Center (DLR) – Institute for AI Safety and Security

3rd FPGA Developer's Forum (FDF) meeting 2026, CERN



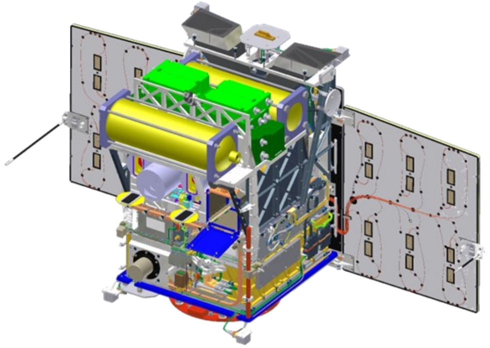
MOTIVATION

Motivation – Satellite Telemetry Data & Telecommands

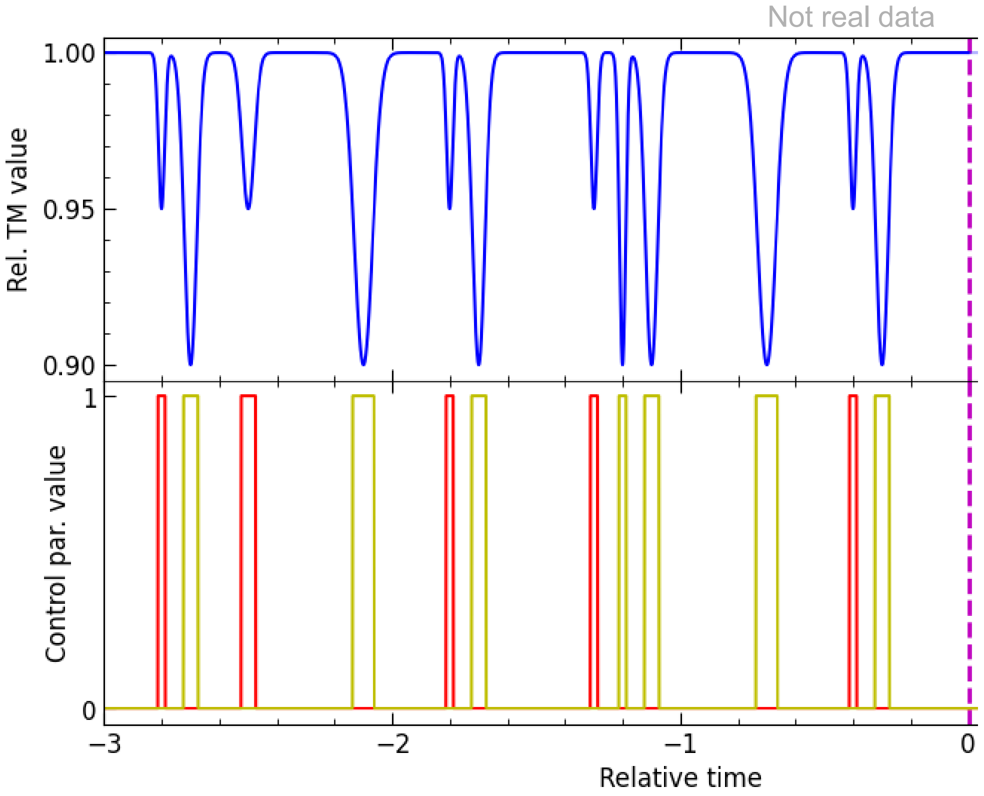


Motivation – Battery Voltage vs Telecommands

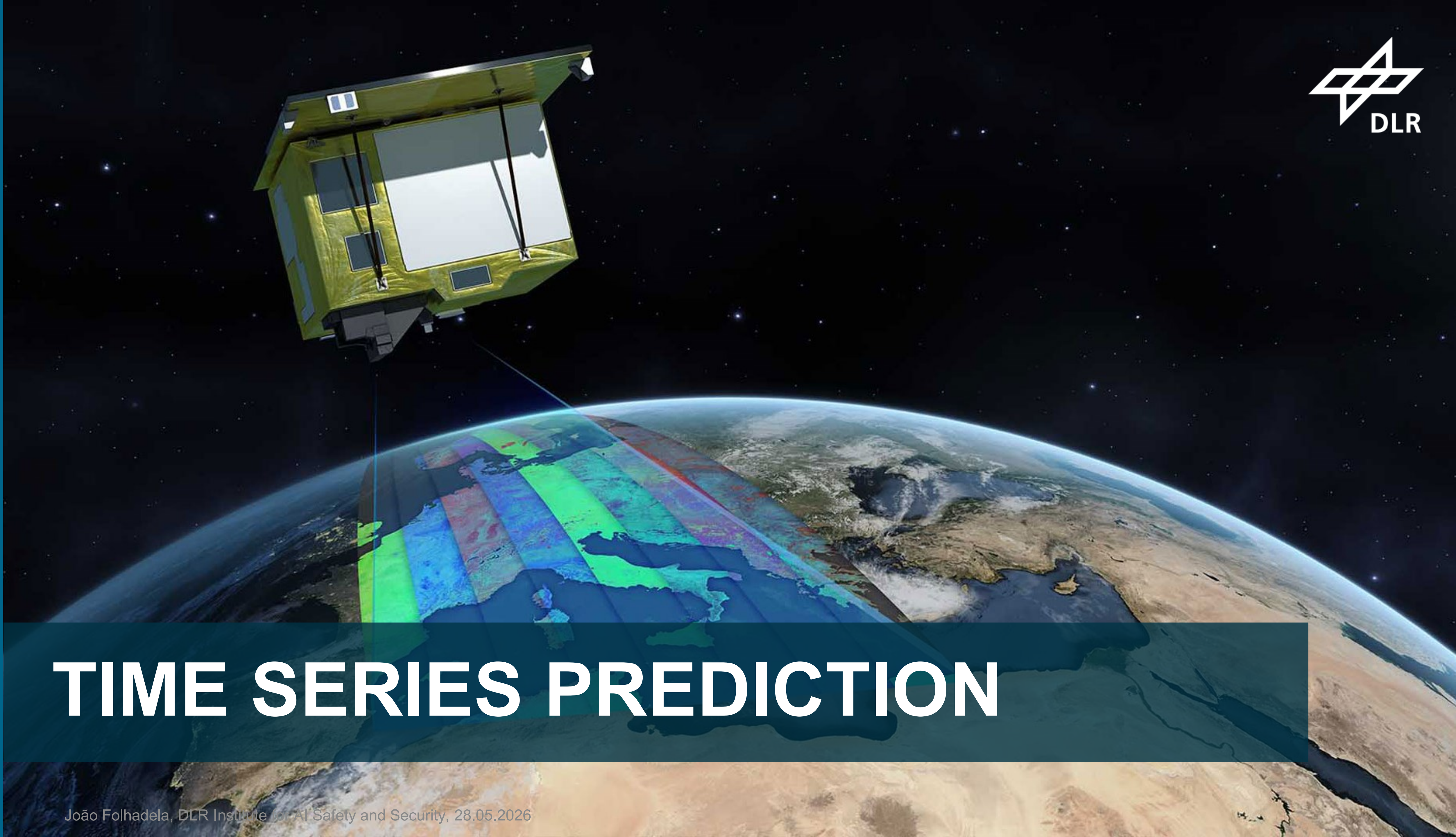
- Too low voltage imperils satellite operations
- Impacts:
  - Use of payload instruments
  - Eclipses
  - Orientation of solar panels
  - Temperature and ageing of battery



BIROS Image: DLR, CC BY-NC-ND 3.0



On-board prediction of the voltage due to planned Telecommands so impacts can be minimized.



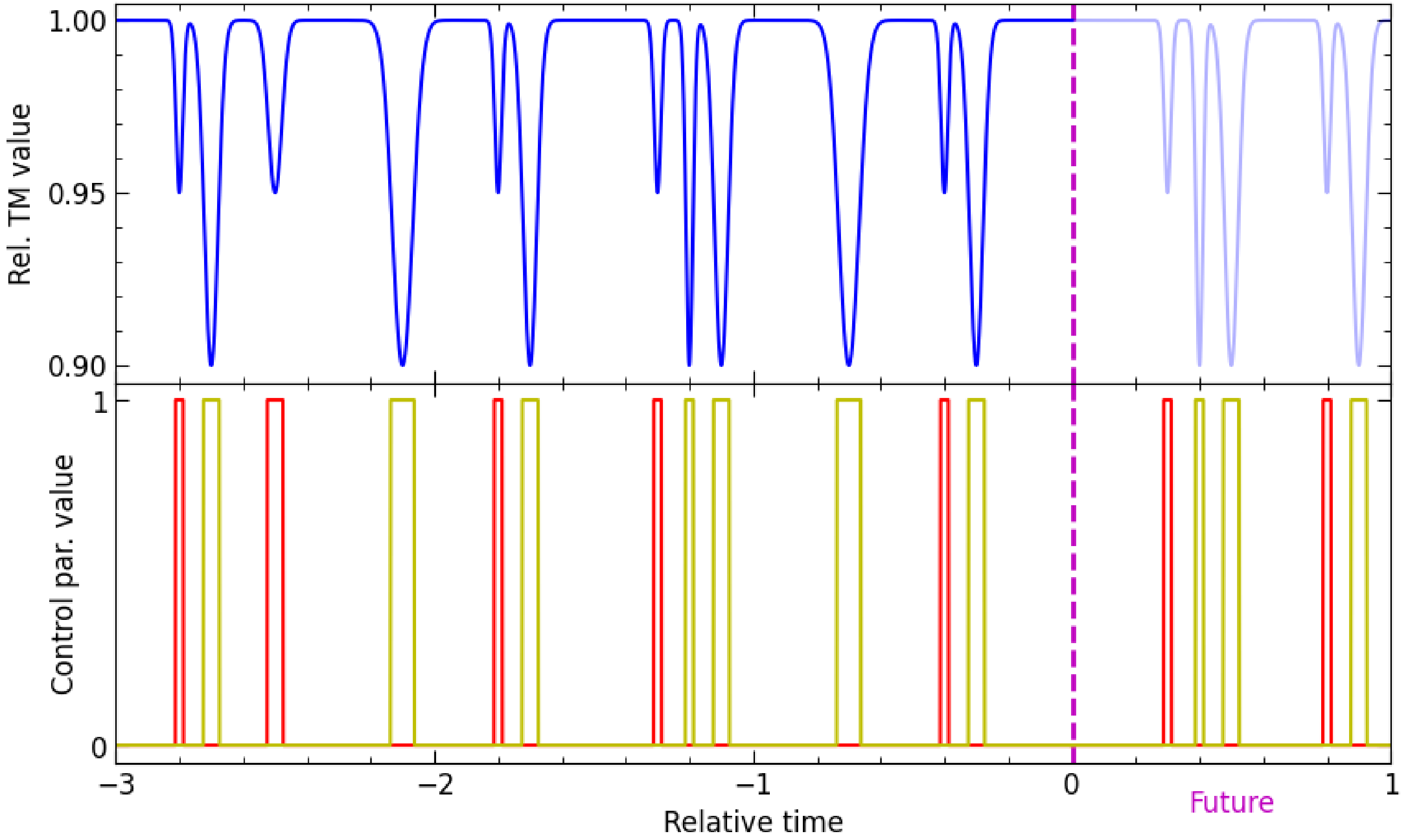
TIME SERIES PREDICTION

Time Series Prediction – Satellite battery voltage

Not real data

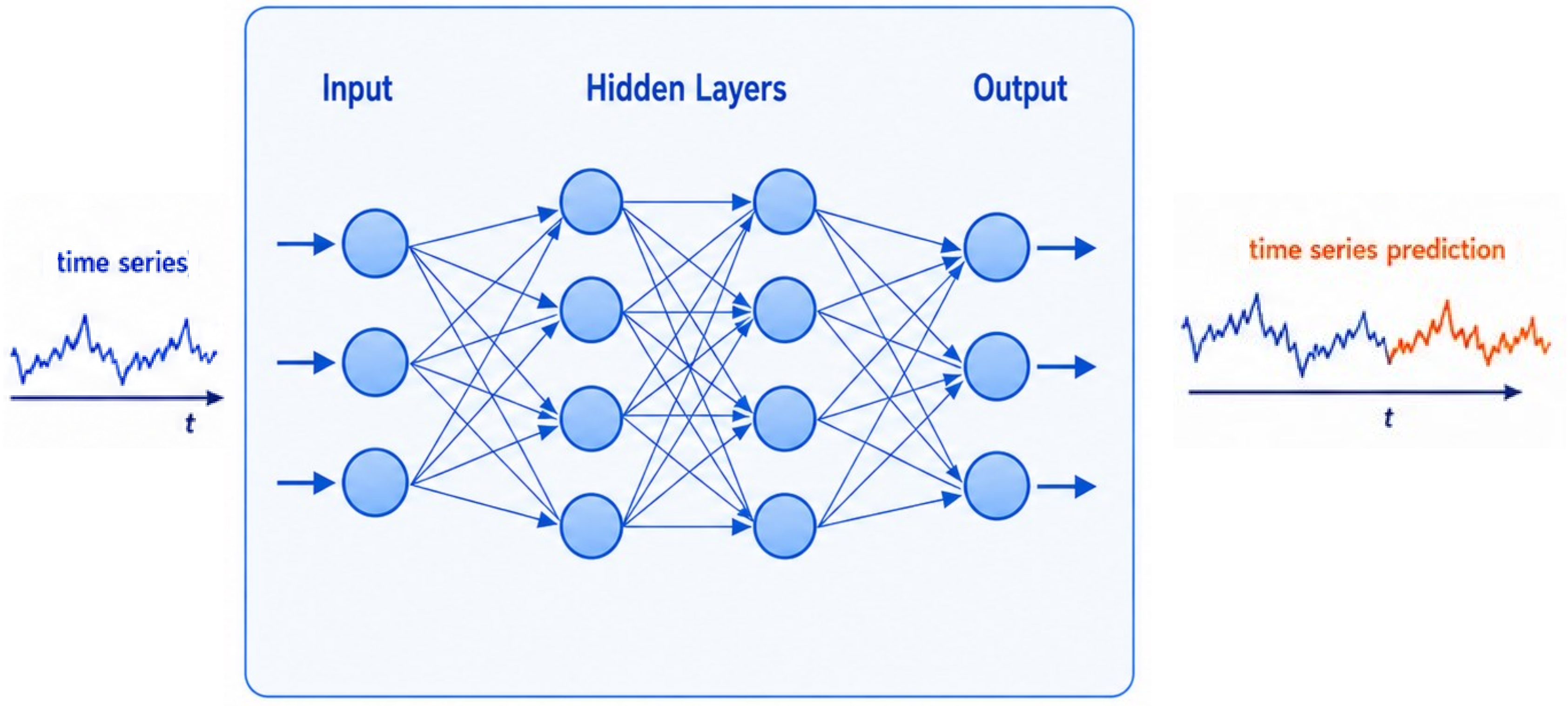
Received telemetry

Control par. based on executed and planned TC



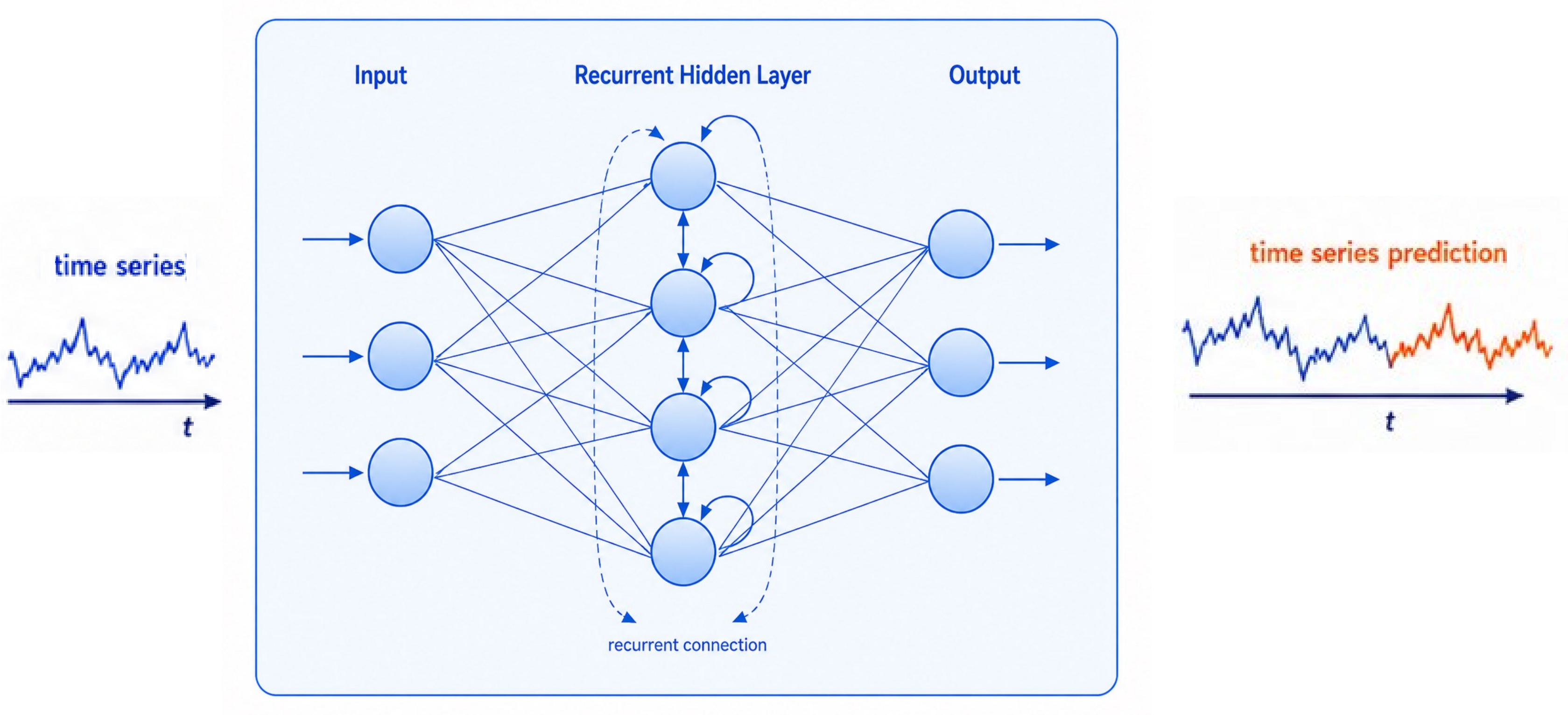
Introduction

Neural Network



Introduction

Recurrent Neural Network

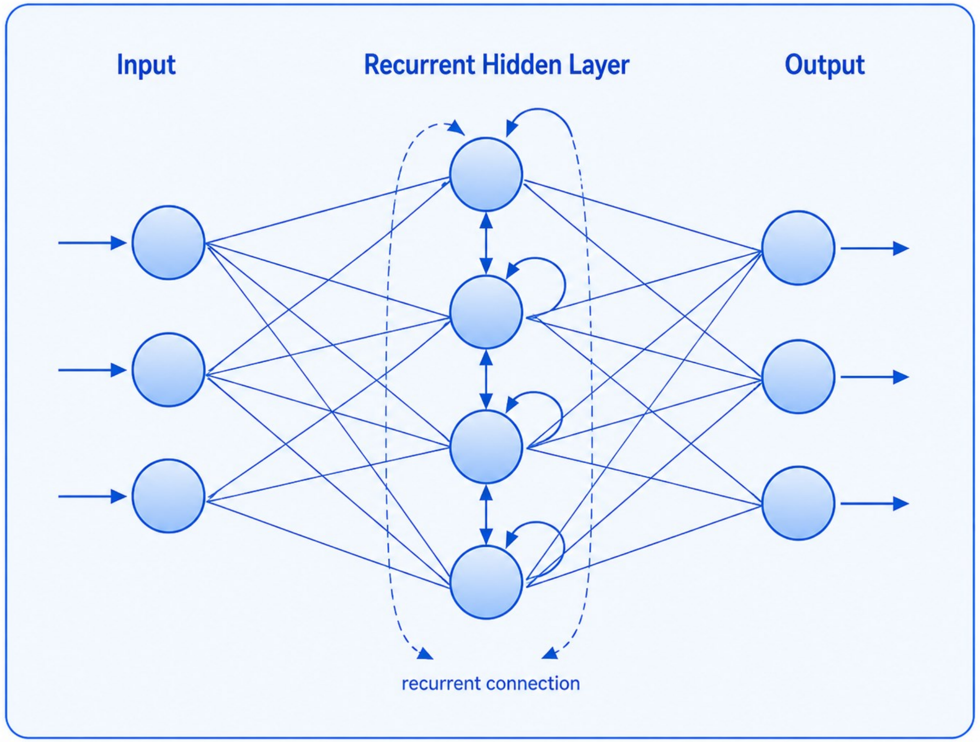


+ Increases accuracy
- Expensive train

João Folhadela, DLR Institute for AI Safety and Security, 28.05.2026

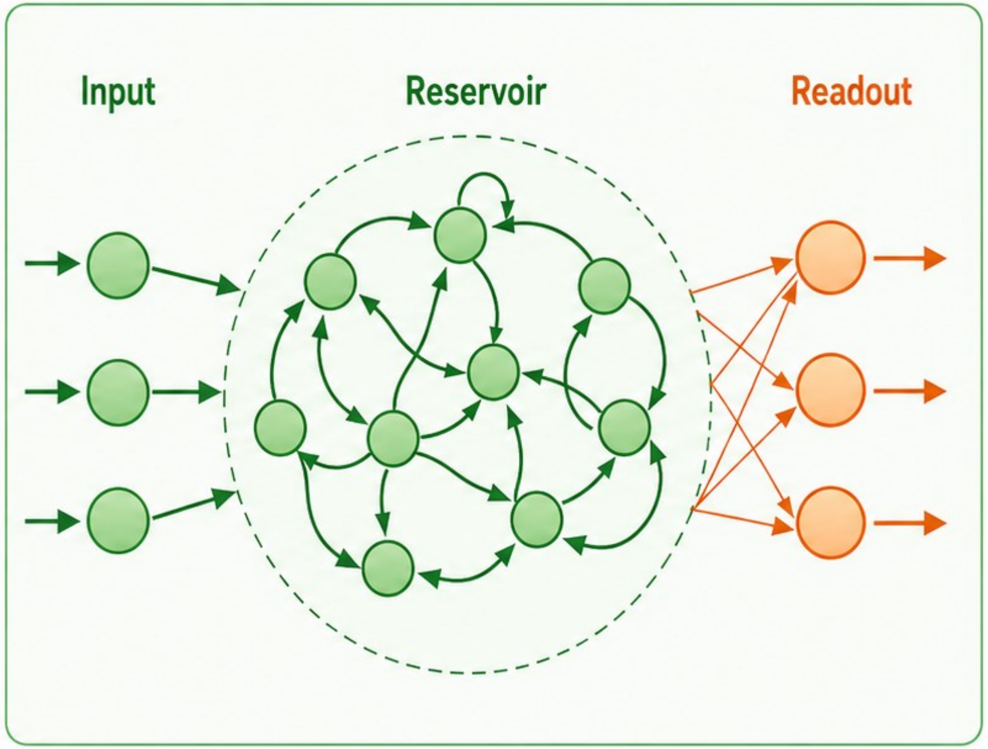
Introduction

Recurrent Neural Network



All weights (connections) are trained

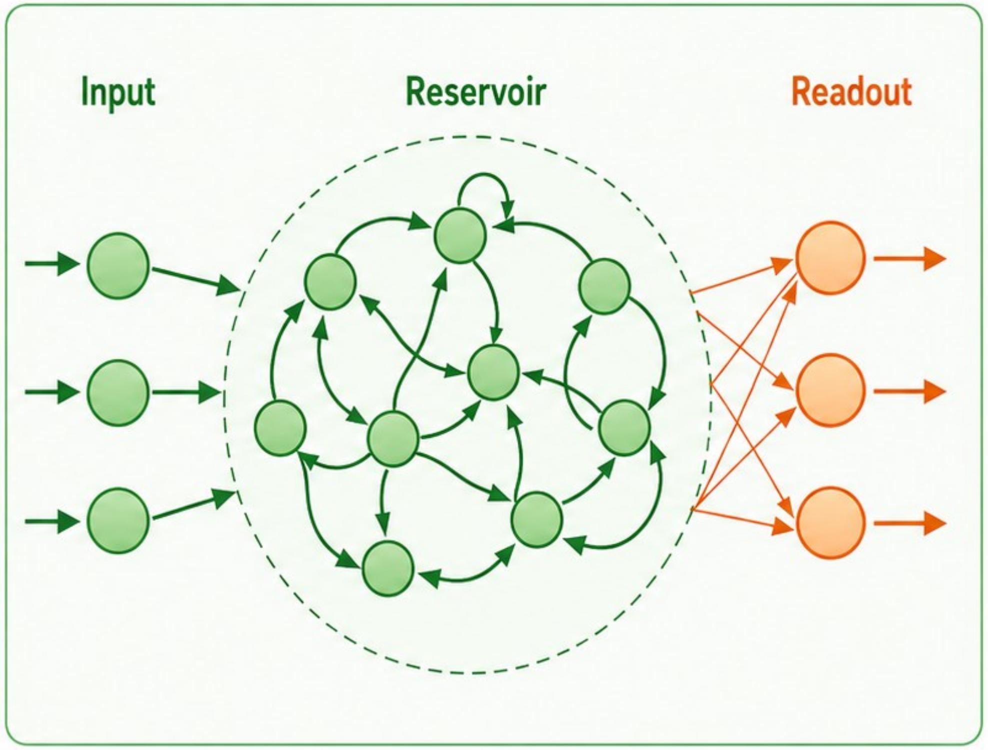
Reservoir Computing



Only readout weights are trained

Introduction

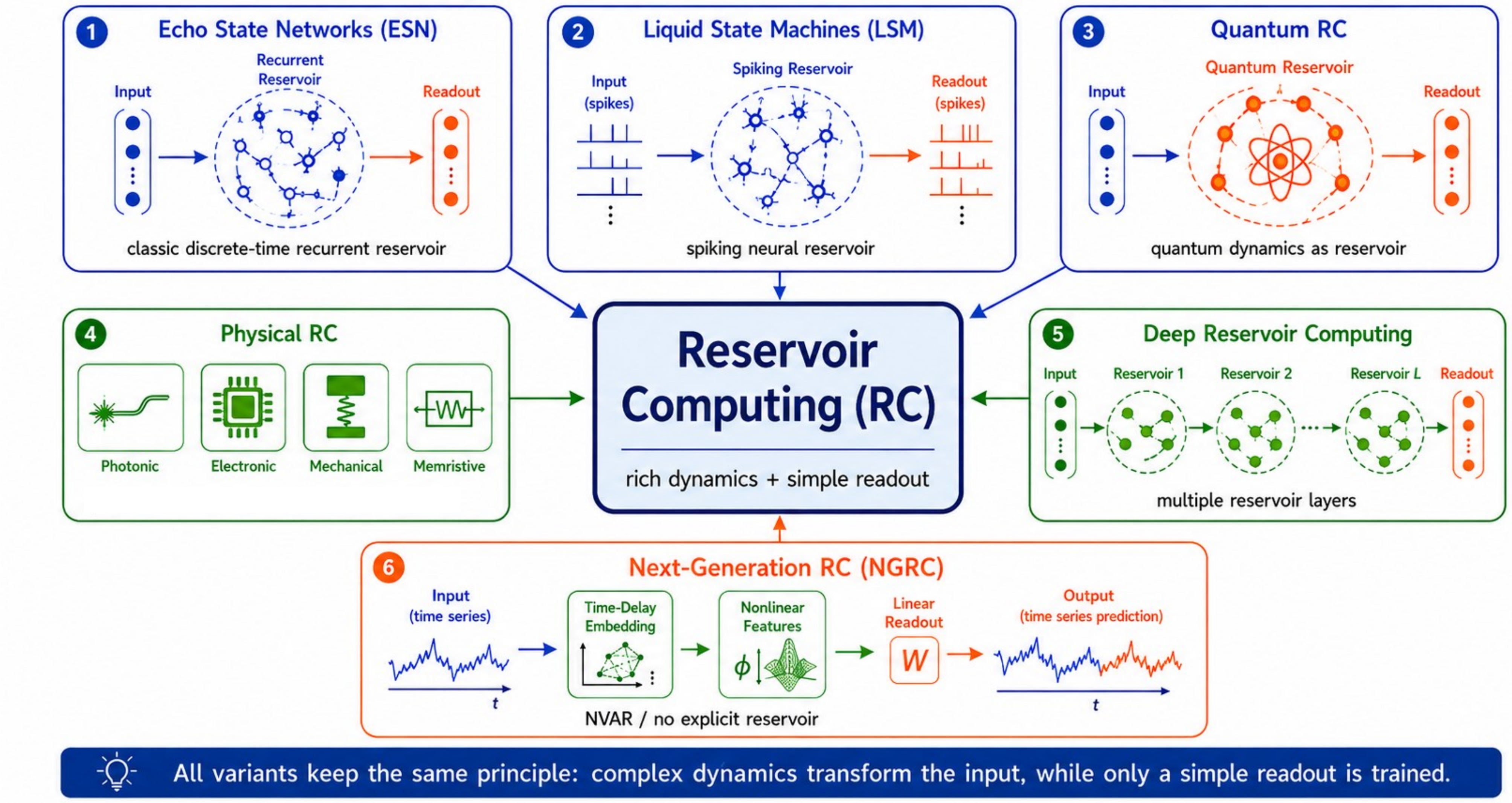
Reservoir Computing



Better fit for FPGA on-board deployment compared to classic neural networks:

- Simpler training/readout stage, often a regression
- Custom datapaths
- Simpler nonlinear operations

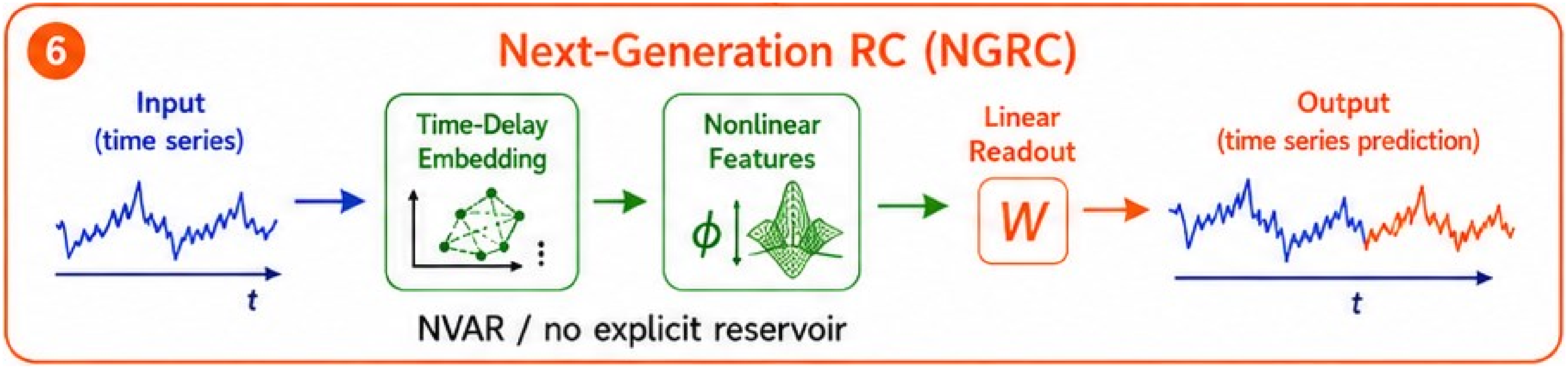
Introduction



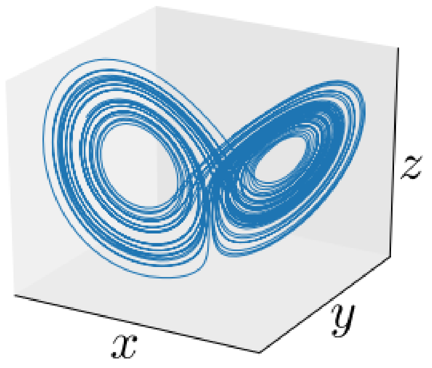
QCi NeMoQC

Next Generation Reservoir Computing

- NGRC-RC equivalence: “There exists a Nonlinear Vector Autoregression (NVAR) that can perform equally well as an optimized RC” – D. Gauthier et. al Nature Commun 2021
- Faster training and interpretable results.
- FPGA fit – no activation functions, fewer parameters, trivial implementation



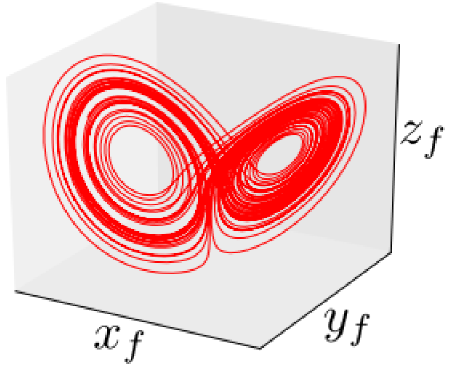
Proof of Concept – Predicting the Lorenz Attractor

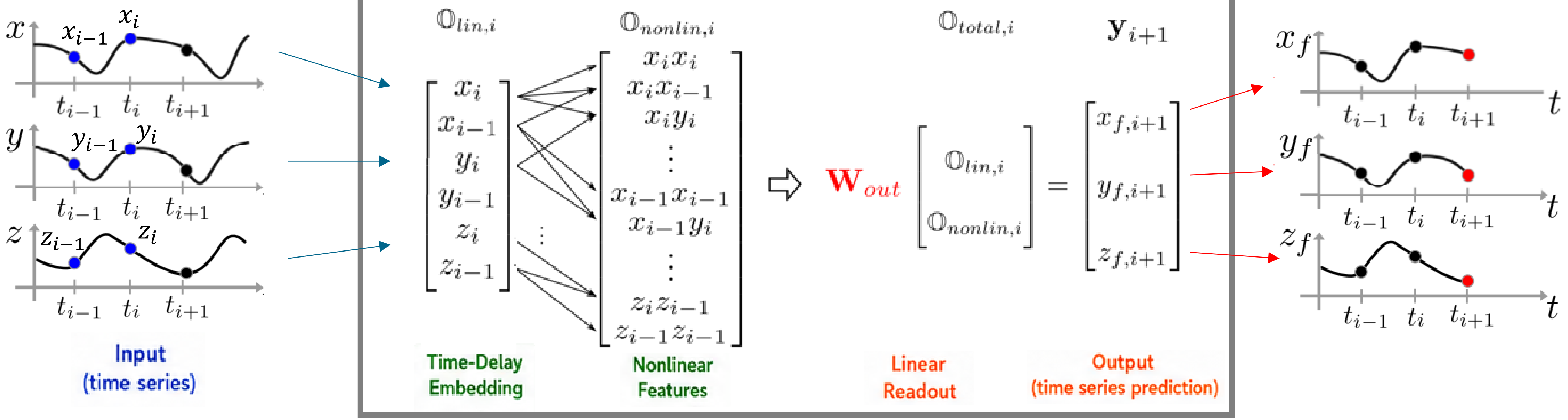


Ridge Regression Training

$$W_{out} = (r^{\intercal} r + \beta\, I)^{-1}\, r^{\intercal} Y_{train}$$

$r^{\intercal} = [\mathbb{O}^{\intercal}_{\mathrm{total},0} \;\; \dots \;\; \mathbb{O}^{\intercal}_{\mathrm{total},i}]$, $Y_{train} = [y_0 \;\; \dots \;\; y_i]$



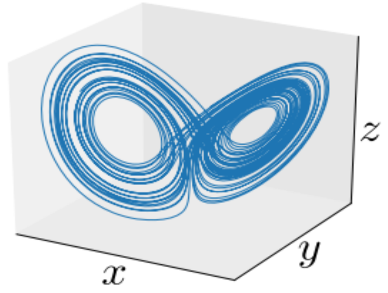


D. Gauthier et al., Nat. Comm., 2021

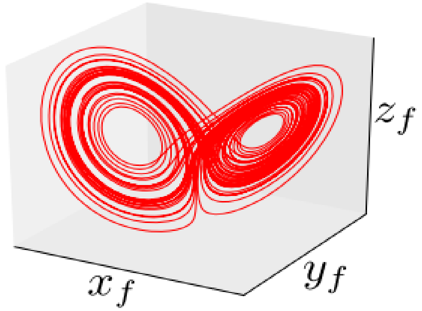
2nd FDF 2025 – NRGC Accelerator

Software

- Model with only 81 parameters
- Trained with 400 timesteps



Real



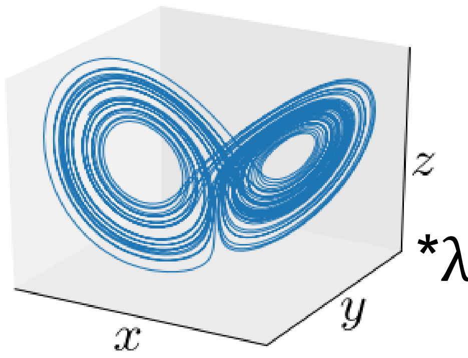
Predicted

FPGA accelerator on Zynq7000 SoC

- 2.5µs/inference @125MHz with 32-bit float precision
- On-chip estimated power 300 mW

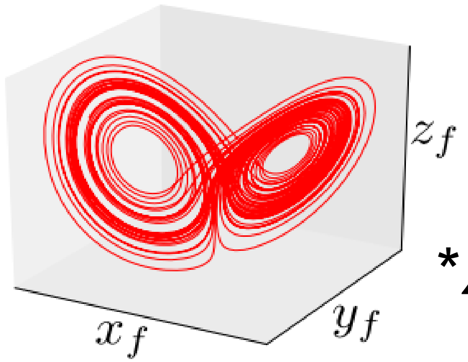
- Fixed weights, UART communication
- Accuracy…

Real vs. Predicted Lorenz



$^{*}\lambda_{max} = 0.79$

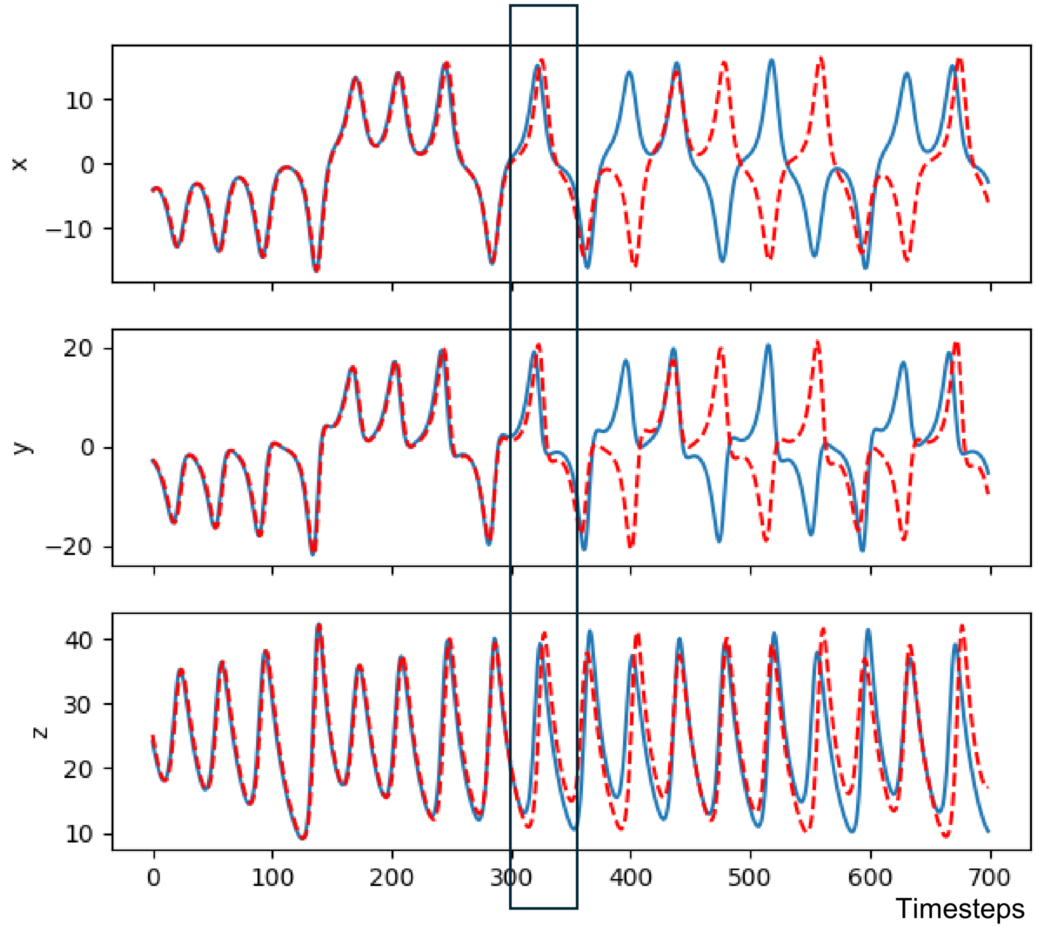
Real



$^{*}\lambda_{max} = 0.77$

Predicted

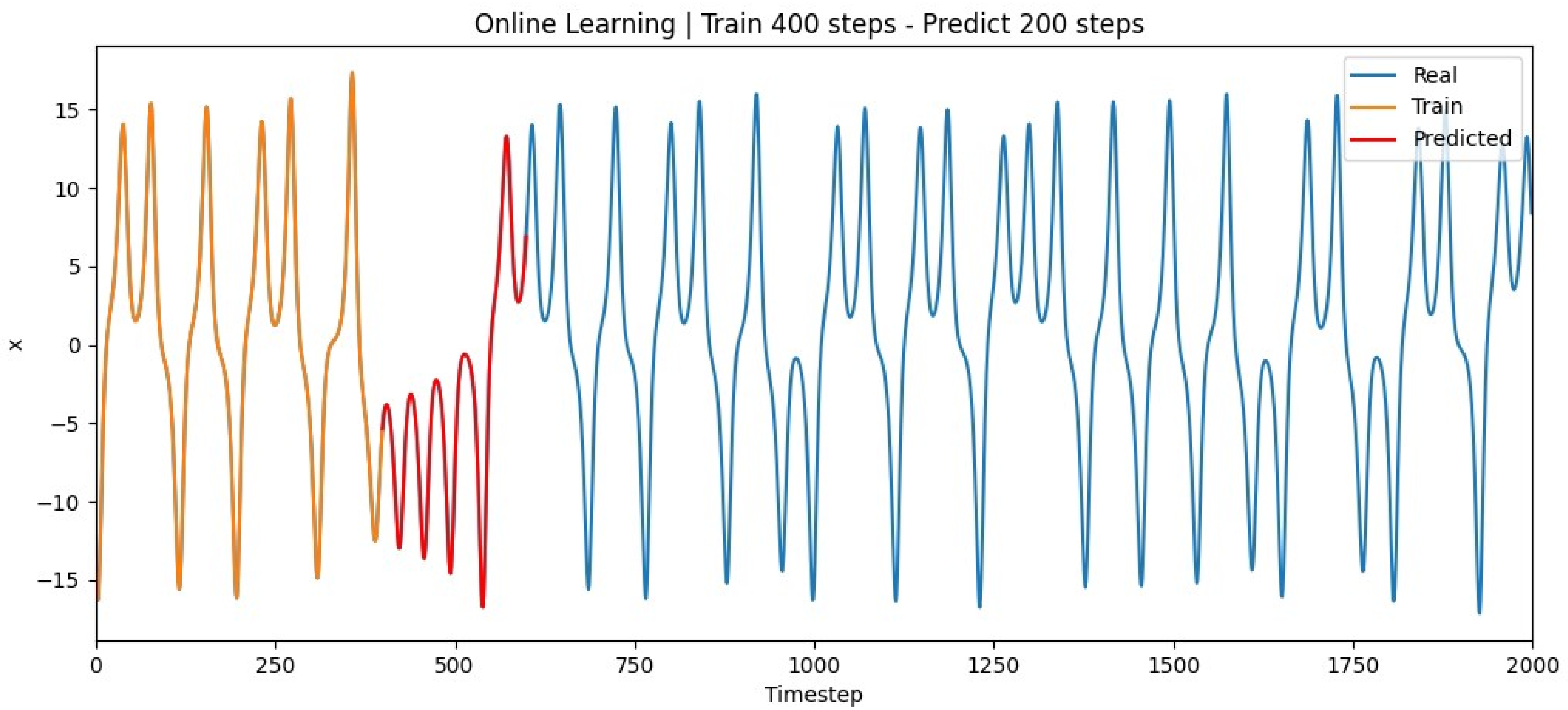
Prediction starts to collapse



***Largest Lyapunov Exponent** λ_{max} **:** measures sensitivity to initial conditions. Similar values suggest similar chaotic behaviour.

Online Learning

Since NGRC train is cheap, continuously retrain on new data



ONLINE LEARNING ON SOC

Online Learning on SoC

- Ridge regression training on FPGA fabric is non-trivial:

Lots of MACs

Memory

$$\boldsymbol{W}_{out} = (\boldsymbol{r}^{\intercal}\boldsymbol{r} + \beta\, \boldsymbol{I})^{-1}\, \boldsymbol{r}^{\intercal} Y_{train}$$

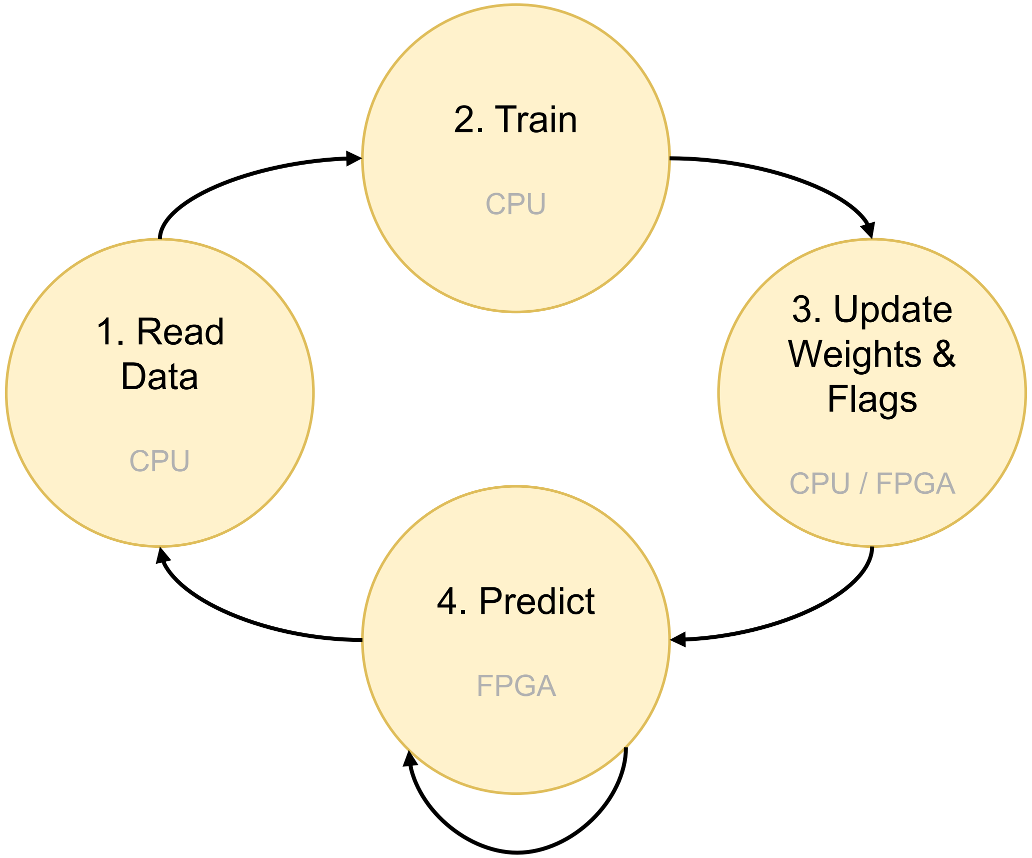
Matrix Inversion

→ Solution: use Zynq’s Processing System (PS) CPU for training…

Online Learning on SoC

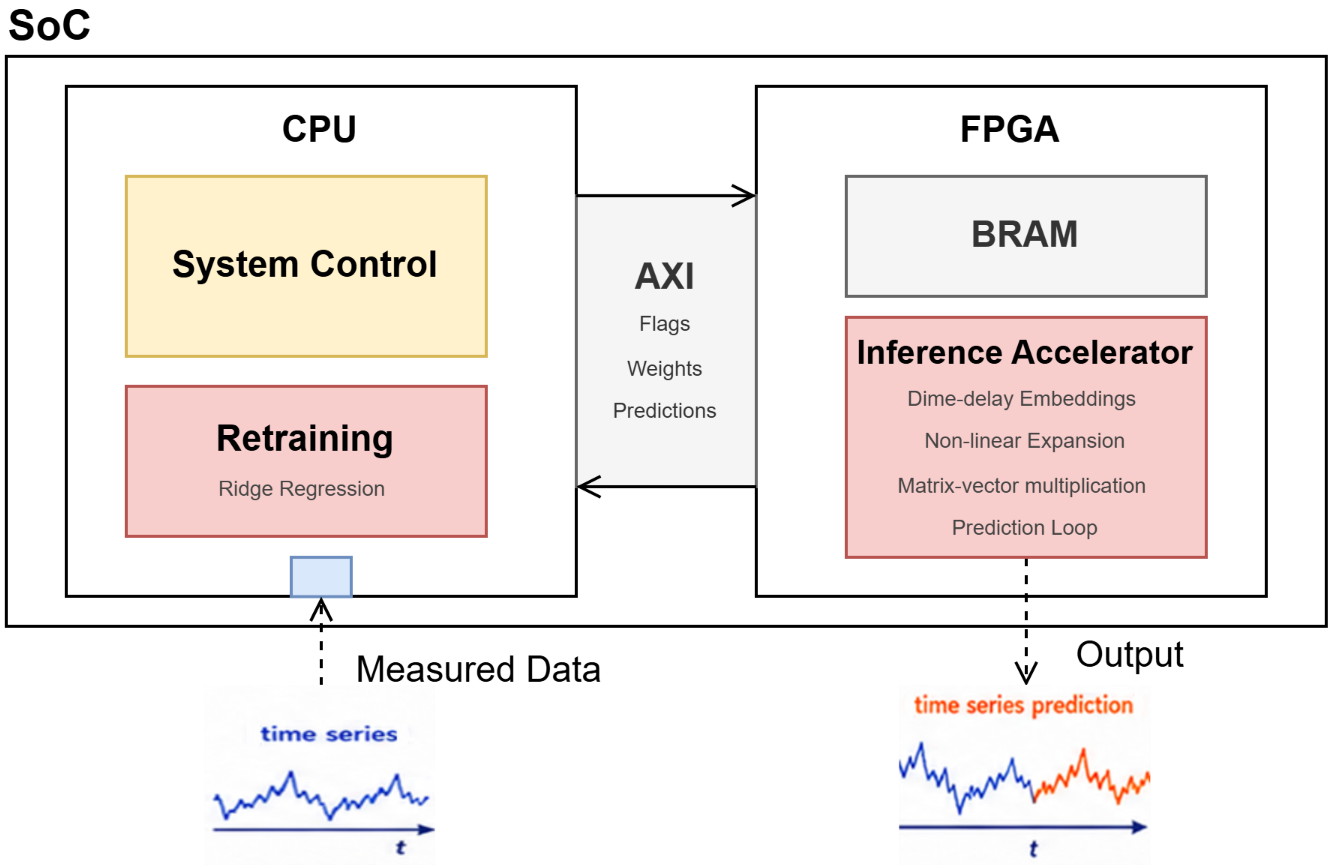
→ Solution: use Zynq's Processing System (PS) CPU for training…

→ And Programmable Logic (FPGA) to accelerate inference.



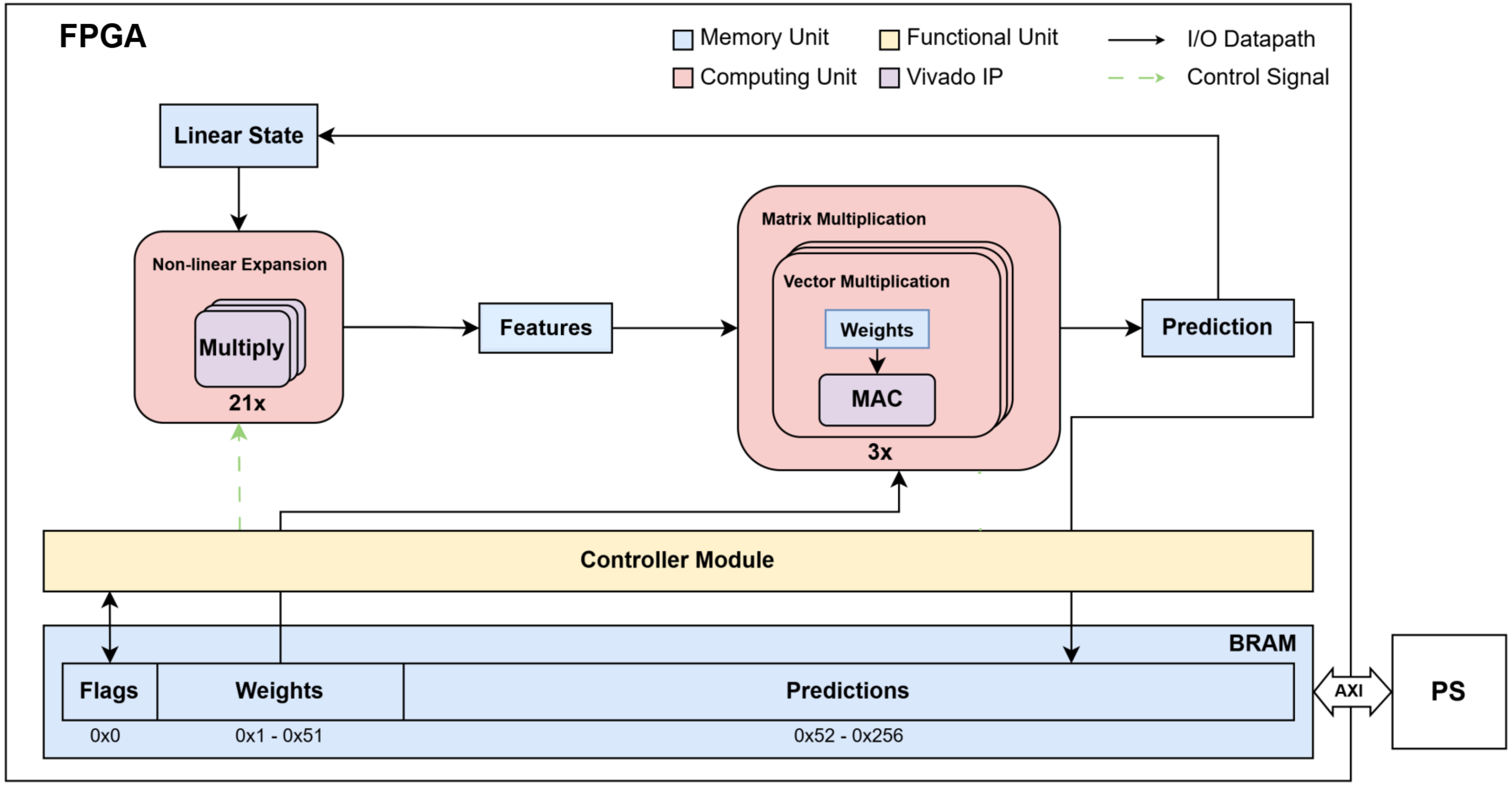
Online Learning on SoC

System Architecture



Online Learning on SoC

System Architecture - FPGA



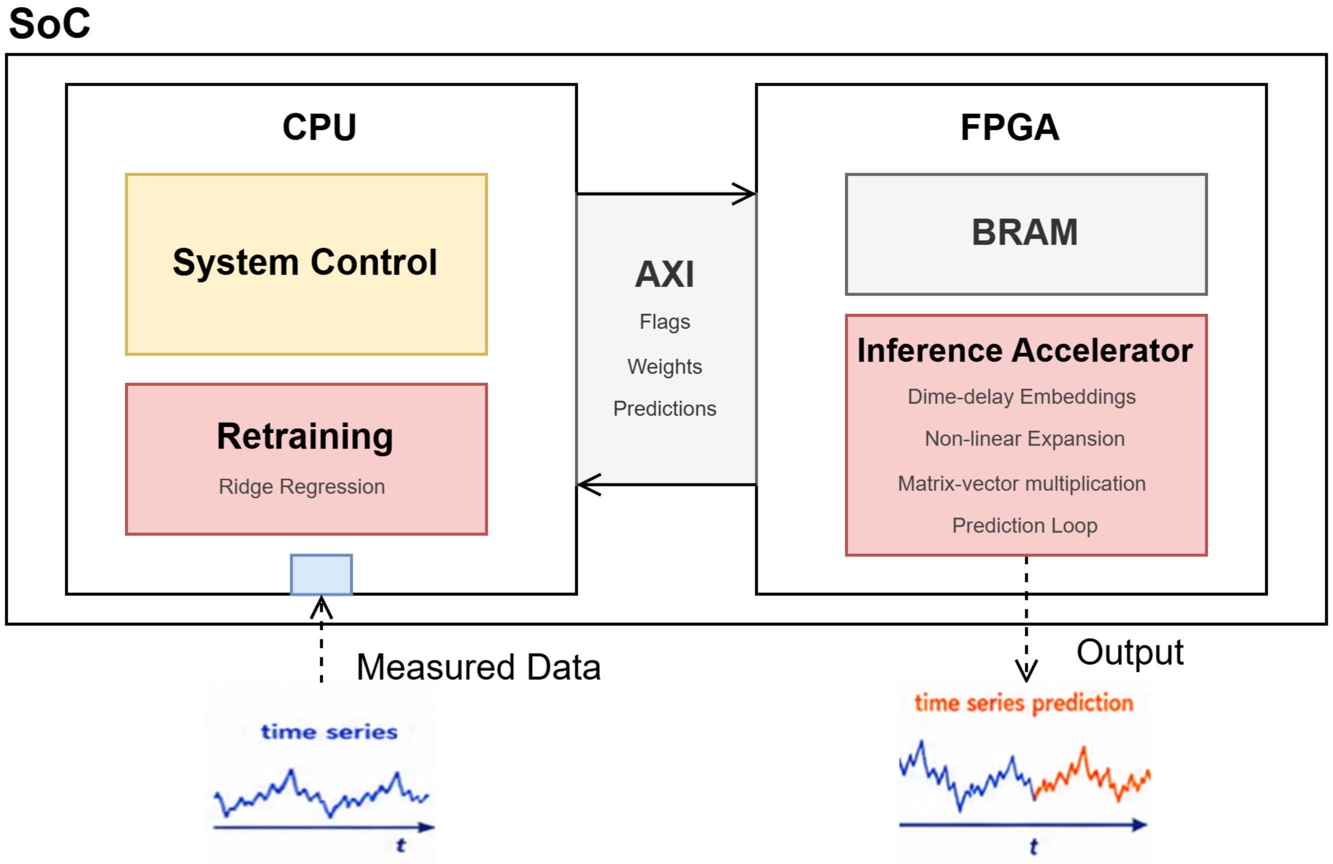
Online Learning on SoC

Resource Usage

Resource	Utilization	Available	Utilization %
LUT	8.2k	53.2k	15.5 %
LUTRAM	0.6k	17.4k	3.4 %
FF	12k	106.4k	11 %
BRAM	1	140	1.4 %
DSP	55	220	25 %

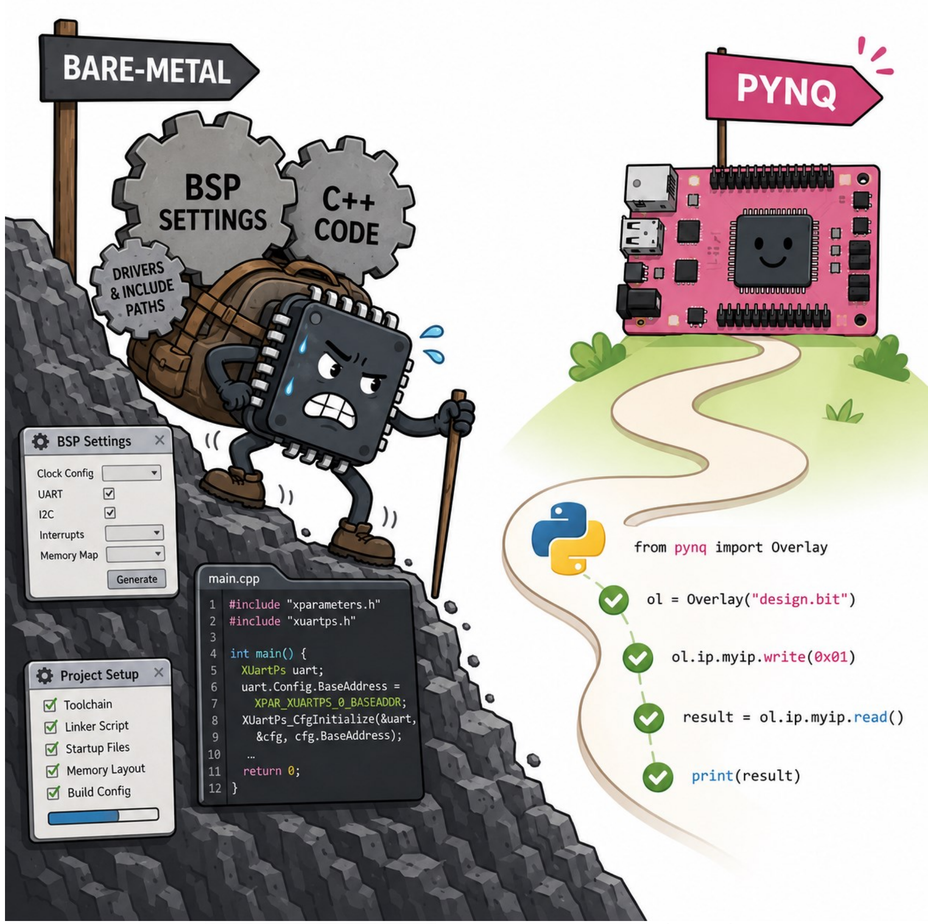
Online Learning on SoC

System Architecture



Online Learning on SoC

System Architecture - CPU



ChatGPT generated

Implementations based on SCAN package:

→ PYNQ (python)
- ease of development
- prototyping
- verification

→ Bare-Metal (C++)
- higher performance

SCAN Source-Code

https://github.com/DLR-KI/scan

Online Learning on SoC - Results

Summary

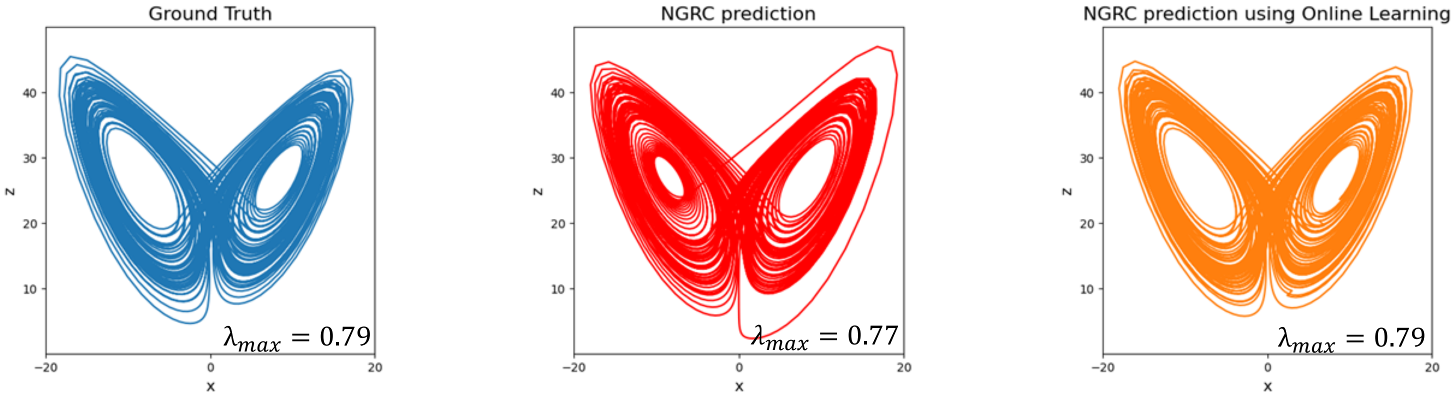
Deployment	Training Latency	Inference Loop Latency	End-to-end Latency
PYNQ (no FPGA)	257 ms	479 ms	749 ms
Bare-Metal (no FPGA)	160 ms	31 ms	192 ms
PYNQ	257 ms	0.68 ms	273 ms
Bare-Metal	**160 ms**	**0.64 ms**	**162 ms**

Train/Inference on 400/200 timesteps

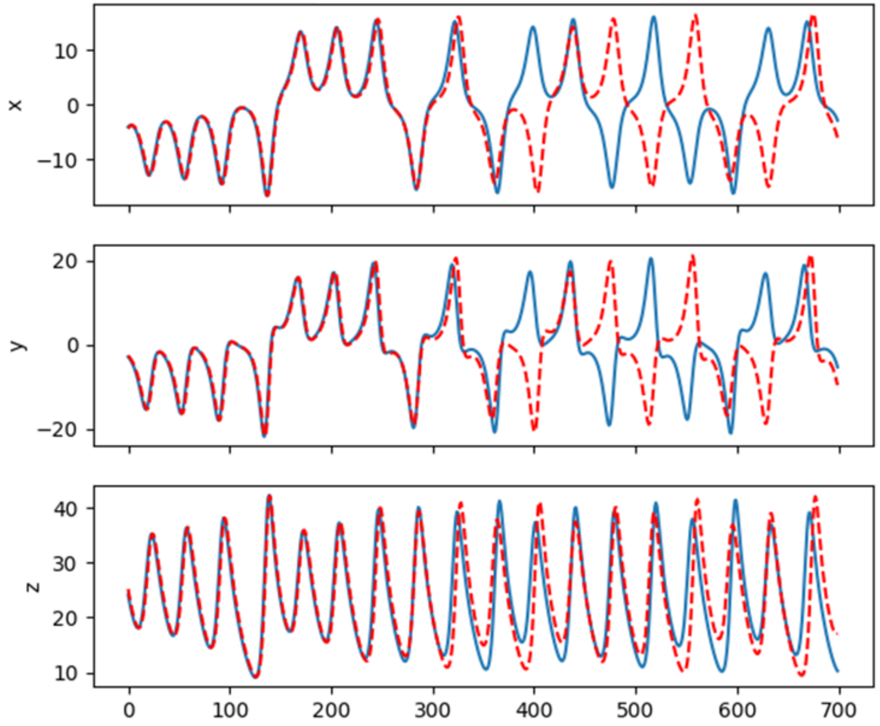
For 1 Hz Telemetry data this is more than enough

Power Estimation	Clock Frequency	Inference Latency
1.776 W	166 MHz	3.147 µs

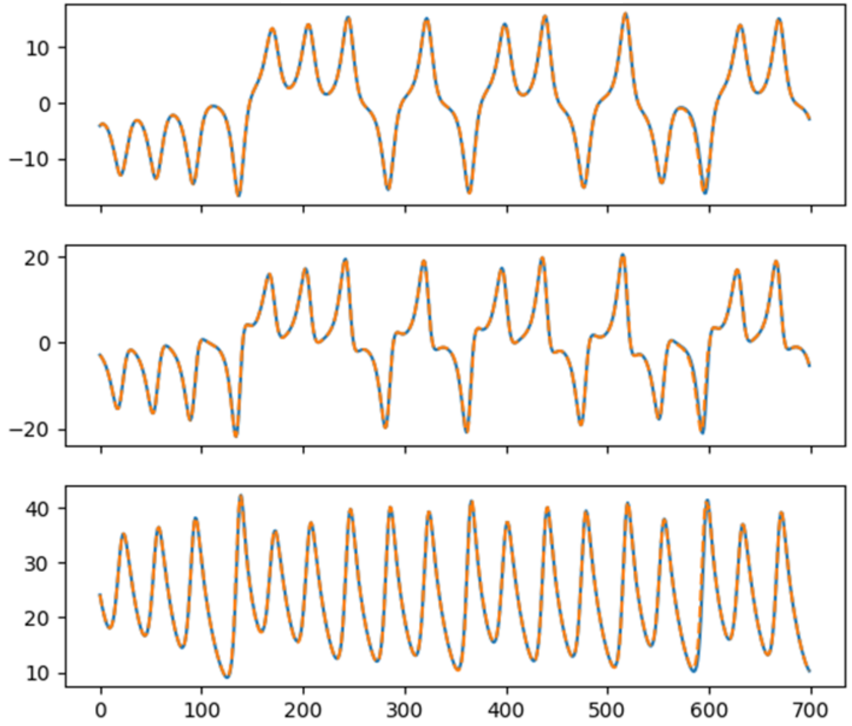
Online Learning - Results



Without Online Learning



With Online Learning



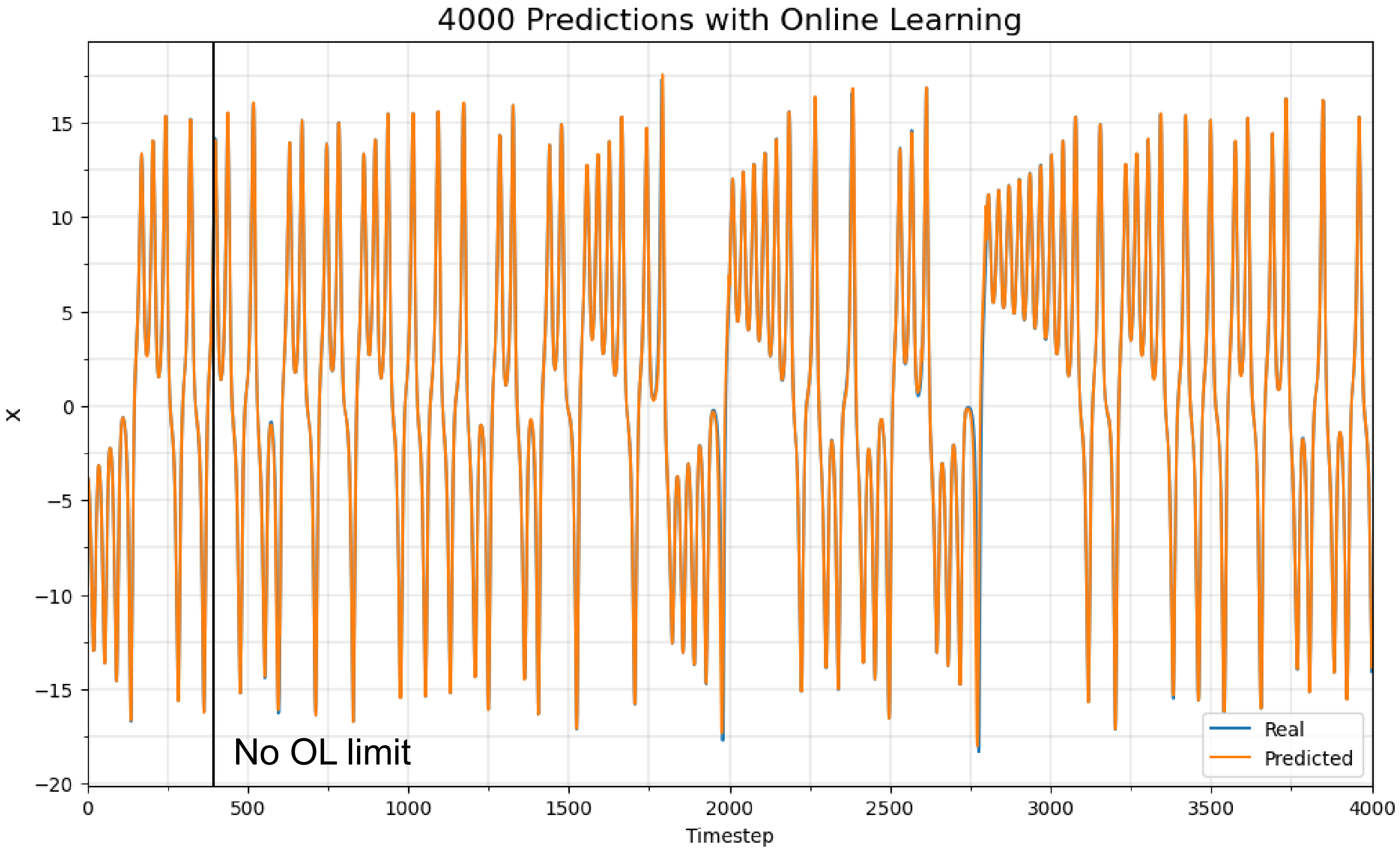
Online Learning - Results

Long-Horizon Forecast

- Previously $^*NRMSE_x = 60\%$
- With OL $NRMSE_x = 4\%$

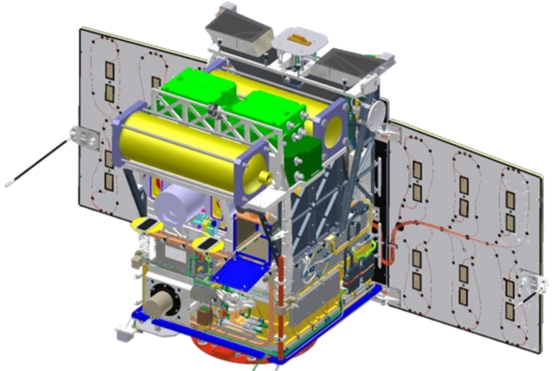
*Normalised Root Mean Squared Error

$$NRMSE = \frac{1}{\max(X)} \sqrt{\frac{\sum_1^N (\text{Predicted}_i - \text{Real}_i)^2}{N}}, \qquad N = 4000$$



Summary

- Online Learning method of NGRC that significantly extends forecast horizon
- Low-latency SoC architecture where CPU trains – FPGA predicts
- Trivial and interpretable implementation



BIROS Image: DLR, CC BY-NC-ND 3.0

Outlook

- Expand on hardware optimization (pipelining, quantization, pruning, etc.)
- Benchmark on modern devices (ex. Ultrascale+)
- Finding optimal NGRC embeddings for Telemetry Data for on-board computing

THANK YOU!

FEEDBACK, QUESTIONS?

DLR